Package 'archivist'

December 31, 2016

Version 2.1.2

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 (>= 3.2.0)

Imports RCurl, digest, httr, DBI, lubridate, RSQLite, magrittr

Suggests shiny, dplyr, testthat, ggplot2, devtools, knitr

Enhances archivist.github

URL https://pbiecek.github.io/archivist/

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

RoxygenNote 5.0.1

NeedsCompilation no

Author Przemyslaw Biecek [aut, cre],
Marcin Kosinski [aut],
Witold Chodor [ctb]

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

Date/Publication 2016-12-31 18:45:14

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 on, 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 along 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 artifacts by using its class, name, date of creation or other properties. It also facilitates restoring such artifacts. Archivist allows you 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.

Extensions of **archivist** are

- Tools for Archiving, Managing and Sharing R Objects via GitHub: archive marcinkosinski.github.io/archivist.github/

Details

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

Blogging

We have prepared history of blog posts and conference talks about **archivist** under this link http://pbiecek.github.io/archivist/P

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: `Repository`, `Tags`, `%a%`, `addHooksToPrint`, `addTagsRepo`, `aformat`, `ahistory`, `alink`, `aoptions`, `aread`, `asearch`, `asession`, `atrace`, `cache`, `copyLocalRepo`, `createLocalRepo`, `createMDGallery`, `deleteLocalRepo`, `getRemoteHook`, `getTagsLocal`, `loadFromLocalRepo`, `md5hash`, `restoreLibs`, `rmFromLocalRepo`, `saveToLocalRepo`, `searchInLocalRepo`, `setLocalRepo`, `shinySearchInLocalRepo`, `showLocalRepo`, `splitTagsLocal`, `summaryLocalRepo`, `zipLocalRepo`

addHooksToPrint *Add* **archivist** *Hooks to* **rmarkdown** *Reports*

Description

addHooksToPrint adds an overloaded version of the print function for objects of selected class. The overloaded function will add all objects of selected class to the Repository and then will add hooks (to the Remote or Local Repository) to the HTML report (generated in **rmarkdown**) for these objects (artifacts - archivist-package). The great example can be seen in this blogpost http://www.r-bloggers.com/why-should-you-backup-your-r-objects/.

Usage

```
addHooksToPrint(class = "ggplot", repoDir = aoptions("repoDir"),
  repo = aoptions("repo"), user = aoptions("user"), branch = "master",
  subdir = aoptions("subdir"), format = "markdown")
```

Arguments

class	A character with a name of class (one or more) that should be archived.
repoDir	A character containing a name of a Local Repository.
repo	A character with a name of a Remote repository on which the Repository is archived. If repo = NULL then hooks will be added to files in local directories.
user	A character with a name of a Remote-repository user on whose account the repo is created.
branch	A character with a name of Remote-repository's branch on which the Repository is archived. Default branch is master.
subdir	A character with a name of a subdirectory on a Remote repository on which the Repository is stored. If the Repository is stored in main folder on a Remote repository, this should be set to subdir = "/" as default.
format	A character denoting format as in alink.

Note

One can specify userTags as in saveToLocalRepo for artifacts by adding "tags" attribute. See note secion about that in saveToLocalRepo.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# only in Rmd report, links to github repository
addHooksToPrint(class="ggplot", repoDir = "arepo",
repo="graphGallery", user="pbiecek")
# only in Rmd report, links to local files
addHooksToPrint(class="ggplot", repoDir = "arepo",
repo=NULL)

## End(Not run)
```

---

addTagsRepo *Add new Tags to the Existing Repository*

Description

addTagsRepo adds new Tags to the existing Repository.

Usage

```
addTagsRepo(md5hashes, repoDir = NULL, FUN = NULL, tags = NULL, ...)
```

Arguments

md5hashes	a character vector of md5hashes specifying to which corresponding artifacts new Tags should be added. See Note to get to know about the length of tags and md5hashes parameters.
repoDir	A character that specifies the directory of the Repository to which new Tags will be added. If it is set to NULL (by default), it uses the repoDir specified in setLocalRepo.
FUN	A function which is evaluated on the artifacts for which md5hashes are given. The result of this function evaluation are Tags which will be added to the local Repository.
tags	A character vector which specifies what kind of Tags should be added to artifacts corresponding to given md5hashes. See Note to get to know about the length of tags and md5hashes parameters. One can specify either FUN or tags.
...	Other arguments that will be passed to FUN.

Details

addTagsRepo function adds new Tags to artifacts that are already stored in the repository. One can add new Tags either explicitly with tags parameter or by passing a function which extracts Tags from selected artifacts corresponding to md5hashes. To learn more about artifacts visit archivist-package.

Note

One should remember that length(tags) modulo length(md5hashes) must be equal to 0 or length(md5hashes) modulo length(tags) must be equal to 0.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:

## We Take all artifacts of lm class from repository,
## extract R^2 for them and store as R^2:number Tags

# Creating empty repository
exampleRepoDir <- tempfile()
createLocalRepo(exampleRepoDir, force=TRUE)

# Saving lm artifacts into repository
m1 <- lm(Sepal.Length~Species, iris)
saveToLocalRepo(m1, exampleRepoDir)
m2 <- lm(Sepal.Width~Species, iris)
saveToLocalRepo(m2, exampleRepoDir)

# We may see what kind of Tags are related to "m1" artifact corresponding to
# "9e66edd297c2f291446f3503c01d443a" md5hash
getTagsLocal("9e66edd297c2f291446f3503c01d443a", exampleRepoDir, "")

# We may see what kind of Tags are related to "m2" artifact corresponding to
# "da1bcaf68752c146903f700c1a458438" md5hash
getTagsLocal("da1bcaf68752c146903f700c1a458438", exampleRepoDir, "")

# We Take all objects of lm class from repository
md5hashes <- searchInLocalRepo(repoDir=exampleRepoDir, "class:lm")
```

```
# Adding new tag "test" explicitly
addTagsRepo(md5hashes, exampleRepoDir, tags = "test")

# Adding new tag "R^2: " using FUN parameter
addTagsRepo(md5hashes, exampleRepoDir, function(x) paste0("R^2:",summary(x)$r.square))

# And now: Tags related to "m1" artifact are
getTagsLocal("9e66edd297c2f291446f3503c01d443a", exampleRepoDir, "")

# And now: Tags related to "m2" artifact are
getTagsLocal("da1bcaf68752c146903f700c1a458438", exampleRepoDir, "")

# One more look at our Repo
showLocalRepo(exampleRepoDir, method = "tags")

# Deleting example repository
deleteLocalRepo(exampleRepoDir, deleteRoot=TRUE)
rm(exampleRepoDir)

## End(Not run)
```

`aformat` *Show Artifact's List of Foramts*

Description

`aformat` extracts artifact's formats. Having formats one may decide which should he read. Currently only rda format is supported for artifact and txt/png for miniatures.

Usage

```
aformat(md5hash = NULL)
```

Arguments

`md5hash`	One of the following (see aread): A character vector which elements are consisting of at least three components separated with '/': Remote user name, Remote repository and name of the artifact (MD5 hash) or it's abbreviation. MD5 hashes of artifacts in current local default directory or its abbreviations.

Value

A vector of characters.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
setLocalRepo(system.file("graphGallery", package = "archivist"))
aformat("2a6e492cb6982f230e48cf46023e2e4f")

# old
aformat("pbiecek/graphGallery/2a6e492cb6982f230e48cf46023e2e4f")
# png
aformat("pbiecek/graphGallery/7f3453331910e3f321ef97d87adb5bad")

## End(Not run)
```

---

ahistory *Show Artifact's History*

Description

ahistory extracts artifact's history and creates a data frame with history of calls and md5hashes of partial results. The overloaded print.ahistory function prints this history in a concise way. The overloaded print.ahistoryKable function prints this history in the same way as kable. When alink=TRUE one can create history table/kable with hooks to partial results (artifacts) as in the alink function.

Usage

```
ahistory(artifact = NULL, md5hash = NULL, repoDir = aoptions("repoDir"),
  format = "regular", alink = FALSE, ...)
```

Arguments

artifact	An artifact which history is supposed to be reconstructed. It will be converted into md5hash.
md5hash	If artifact is not specified then md5hash is used.
repoDir	A character denoting an existing directory in which an artifact will be saved.

format	A character denoting whether to print history in either a "regular" (default) way or like in a "kable" function. See Notes.
alink	Whether to provide hooks to objects like in alink. See examples.
...	Further parameters passed to alink function. Used when format = "kable" and alink = TRUE.

Details

All artifacts created with %a% operator are archivised with detailed information about it's source (both call and md5hash of the input). The function ahistory reads all artifacts that precede artifact and create a description of the input flow. The generic print.ahistory function plots the history in a human readable way.

Value

A data frame with two columns - names of calls and md5hashes of partial results.

Demonstration

This function is well explained on this http://r-bloggers.com/r-hero-saves-backup-city-with-archivist-and-github blog post.

Note

There are provided functions (print.ahistory and print.ahistoryKable) to print the artifact's history. History can be printed either in a regular way which is friendy for the console output or in a kable format which prints the artifact's history in a way kable function would. This is convenient when one prints history in .Rmd files using rmarkdown.

Moreover when user passes format = 'kable' and alink = TRUE then one can use links for remote Repository. Then mdhashes are taken from Local Repository, so user has to specify repo, user and repoDir even though they are set globally, because repo is a substring of repoDir and during evalutation of ... R treats repo as repoDir.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
createLocalRepo("ahistory_check", default = TRUE)
library(dplyr)
iris %a%
filter(Sepal.Length < 6) %a%
 lm(Petal.Length~Species, data=.) %a%
 summary() -> artifact

ahistory(artifact)
ahistory(artifact, format = "kable")
print(ahistory(artifact, format = "kable"), format = "latex")
ahistory(artifact, format = "kable", alink = TRUE, repoDir = "ahistory_check",
repo = "repo", user = "user")


repoDir <- file.path(getwd(), "ahistory_check")
deleteLocalRepo(repoDir, deleteRoot = TRUE)
aoptions('repoDir', NULL, unset = TRUE)
```

`alink` *Return a Link To Download an Artifact Stored on Remote Repository*

Description

`alink` returns a link to download an artifact from the Remote Repository. Artifact has to be already archived on GitHub, e.g with `archivist.github::archive` function (recommended) or saveToRepo function and traditional Git manual synchronization. To learn more about artifacts visit archivist-package.

Usage

```
alink(md5hash, repo = aoptions("repo"), user = aoptions("user"),
  subdir = aoptions("subdir"), branch = "master",
  repoType = aoptions("repoType"), format = "markdown", rawLink = FALSE)
```

Arguments

`md5hash`	A character assigned to the artifact through the use of a cryptographical hash function with MD5 algorithm. If it is specified in a format of `'repo/user/md5hash'` then `user` and `repo` parameters are omitted.
`repo`	The Remote `Repository` on which the artifact that we want to download is stored.
`user`	The name of a user on whose `Repository` the the artifact that we want to download is stored.

subdir	A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in the main folder on the Remote repository, this should be set to subdir = "/" as default.
branch	A character containing a name of the Remote Repository's branch on which the Repository is archived. Default branch is master.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.
format	In which format the link should be returned. Possibilites are markdown (default) or latex.
rawLink	A logical denoting whether to return raw link or a link in the format convention. Default value is FALSE.

Details

For more information about md5hash see md5hash.

Value

This function returns a link to download artifact that is archived on GitHub.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
# link in markdown format
alink('pbiecek/archivist/134ecbbe2a8814d98f0c2758000c408e')
# link in markdown format with additional subdir
alink(user='BetaAndBit',repo='PieczaraPietraszki',
     md5hash = '1569cc44e8450439ac52c11ccac35138',
     subdir = 'UniwersytetDzieci/arepo')
# link in latex format
alink(user = 'MarcinKosinski', repo = 'Museum',
      md5hash = '1651caa499a2b07a3bdad3896a2fc717', format = 'latex')
# link in raw format
alink('pbiecek/graphGallery/f5185c458bff721f0faa8e1332f01e0f', rawLink = TRUE)
alink('pbiecek/graphgallerygit/02af4f99e440324b9e329faa293a9394', repoType='bitbucket')
```

aoptions *Default Options for Archivist*

Description

The function aoptions sets and gets default options for other archivist functions.

Usage

```
aoptions(key, value = NULL, unset = FALSE)
```

Arguments

key	A character denoting name of the parameter.
value	New value for the 'key' parameter.
unset	Set to TRUE if want to set parameter to NULL, i.e. when unsetting Repository aoptions('repoDir', NULL, T).

Details

The function aoptions with two parameters sets default value of key parameter for other archivist functions. The function aoptions with one parameter returns value (stored in an internal environment)) of the given key parameter.

Value

The function returns value that corresponds to a selected key.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

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


## EXAMPLE 1 : SET default local repository using aoptions() function.

# creating example repository
exampleRepoDir <- tempfile()
createLocalRepo(exampleRepoDir)

# "repodDir" parameter in each archivist function will be default and set to exampleRepoDir.
aoptions(key = "repoDir", value = exampleRepoDir)

data(iris)
data(swiss)
# From this moment repoDir parameter may be ommitted in the following functions
saveToLocalRepo(iris)
saveToLocalRepo(swiss)
showLocalRepo()
showLocalRepo(method = "tags")
zipLocalRepo()
file.remove(file.path(getwd(), "repository.zip"))
iris2 <- loadFromLocalRepo( "ff575c2" , value = TRUE)
searchInLocalRepo("name:i", fixed = F)
getTagsLocal("ff575c261c949d073b2895b05d1097c3")
rmFromLocalRepo("4c43f")
showLocalRepo()
summaryLocalRepo()

# REMEMBER that in deleteRepo you MUST specify repoDir parameter!
# deleteLocalRepo doesn't take setLocalRepo's settings into consideration
deleteLocalRepo( exampleRepoDir, deleteRoot = TRUE)
rm( exampleRepoDir )

## EXAMPLE 2 : SET default Github repository using aoptions() function.
aoptions(key = "user", value = "pbiecek")
aoptions(key = "repo", value = "archivist")

# From this moment user and repo parameters may be ommitted in the following functions:
showRemoteRepo()
loadFromRemoteRepo( "ff575c261c949d073b2895b05d1097c3")
this <- loadFromRemoteRepo( "ff", value = T)
file.remove(file.path(getwd(), "repository.zip")) # We can remove this zip file
searchInRemoteRepo( "name:", fixed= FALSE)
getTagsGithub("ff575c261c949d073b2895b05d1097c3")
summaryRemoteRepo( )
searchInRemoteRepo( pattern=c("varname:Sepal.Width", "class:lm", "name:myplot123"),
                          intersect = FALSE )
## EXAMPLE 3 : SET default Github repository using aoptions() function.
showRemoteRepo('Museum', 'MarcinKosinski', subdir = 'ex1')
```

```
aoptions('repo', 'Museum')
aoptions('user', 'MarcinKosinski')
aoptions('subdir', 'ex1')
aoptions('branch', 'master')
showRemoteRepo()
showRemoteRepo(subdir = 'ex2')

aoptions('subdir')

## End(Not run)
```

---

aread *Read Artifacts Given as md5hashes from the Repository*

Description

aread reads the artifact from the Repository. It's a wrapper around loadFromLocalRepo and loadFromRemoteRepo.

Usage

```
aread(md5hash)
```

Arguments

md5hash	One of the following: A character vector which elements are consisting of at least three components separated with '/': Remote user name, Remote repository and name of the artifact (MD5 hash) or it's abbreviation. MD5 hashes of artifacts in current local default directory or its abbreviations.

Details

Function aread reads artifacts (by md5hashes) from Remote Repository. It uses loadFromLocalRepo and loadFromRemoteRepo functions with different parameter's specification.

Note

Before you start using this function, remember to set local or Remote repository to default by using setLocalRepo() or setRemoteRepo functions.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
# read the object from local directory
setLocalRepo(system.file("graphGallery", package = "archivist"))
pl <- aread("7f3453331910e3f321ef97d87adb5bad")
# To plot it remember to have ggplot2 in version 2.1.0
# as this is stated in asession("7f3453331910e3f321ef97d87adb5bad") .
# The state of R libraries can be restored to the same state in
# which 7f3453331910e3f321ef97d87adb5bad was created with the restoreLibs function.

# read the object from Remote
pl <- aread("pbiecek/graphGallery/7f3453331910e3f321ef97d87adb5bad")
# To plot it remember to have ggplot2 in version 2.1.0
# as this is stated in asession("pbiecek/graphGallery/7f3453331910e3f321ef97d87adb5bad") .
# The state of R libraries can be restored to the same state in
# which 7f3453331910e3f321ef97d87adb5bad was created with the restoreLibs function.
```

---

asearch *Read Artifacts Given as a List of Tags*

Description

asearch searches for artifacts that contain all specified Tags and reads all of them from a default or Remote Repository. It's a wrapper around searchInLocalRepo and loadFromLocalRepo and their Remote versions.

Usage

```
asearch(patterns, repo = NULL)
```

Arguments

patterns	A character vector of Tags. Only artifacts that contain all Tags are returned.
repo	One of following: A character with Remote user name and Remote repository name separated by '/'. NULL in this case search will be performed in the default repo, either local or Remote

Details

Function asearch reads all artifacts that contain given list of Tags from default or Remote Repository. It uses both loadFromLocalRepo and searchInLocalRepo functions (or their Remote versions) but has shorter name and different parameter's specification.

Value

This function returns a list of artifacts (by their values).

Note

Remember that if you want to use local repository you should set it to default.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
### default LOCAL version
## objects preparation

# 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)

## creating example default local repository
exampleRepoDir <- tempfile()
```

```
createLocalRepo(repoDir = exampleRepoDir)
## setting default local repository
setLocalRepo( repoDir = exampleRepoDir )

saveToLocalRepo(myplot123)
saveToLocalRepo(iris)
saveToLocalRepo(model)
saveToLocalRepo(model2)
saveToLocalRepo(model3)

## Searching for objects of class:lm
lm <- asearch(patterns = "class:lm")

## Searching for objects of class:lm with coefname:Petal.Width
lm_c_PW <- asearch(patterns = c("class:lm","coefname:Petal.Width"))

# Note that we searched for objects. Then loaded them from repository by their value.


## deleting example repository
deleteLocalRepo(repoDir = exampleRepoDir, deleteRoot = TRUE)
rm(exampleRepoDir)

### default GitHub version
## Setting default github repository
setRemoteRepo( user = "pbiecek", repo = "archivist")

showRemoteRepo(method = "tags")$tag
searchInRemoteRepo(pattern = "class:lm")
searchInRemoteRepo(pattern = "class:gg")
getTagsRemote(md5hash = "cd6557c6163a6f9800f308f343e75e72", tag = "")

## Searching for objects of class:lm
asearch(patterns = c("class:lm"))
## Searching for objects of class:gg
ggplot_objects <- asearch(patterns = c("class:gg"))
# names(ggplot_objects)
# To plot them remember to have ggplot2 in version 2.1.0
# as this is stated in asession("pbiecek/archivist/13b2724139eb2c62578b4dab0d7b2cea") or
asession("pbiecek/archivist/7f3453331910e3f321ef97d87adb5bad") .
# The state of R libraries can be restored to the same state in
# which those objects were created with the restoreLibs function.


### Remote version
## Note that repo argument is passed in the following way to asearch:
## repo = "GitHub user name/GitHub repository name"

## Searching for objects of class:gg
asearch("pbiecek/graphGallery",
        patterns = c("class:gg",
                     "labelx:Sepal.Length")) -> ggplots_objects_v2
```

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

---

asession *Show Artifact's Session Info*

Description

asession extracts artifact's session info. This allows to check in what conditions the artifact was created.

Usage

```
asession(md5hash = NULL)
```

Arguments

md5hash	One of the following (see aread): A character vector which elements are consisting of at least three components separated with '/': Remote user name, Remote repository and name of the artifact (MD5 hash) or it's abbreviation. MD5 hashes of artifacts in current local default directory or its abbreviations.

Value

An object of the class session_info.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
setLocalRepo(system.file("graphGallery", package = "archivist"))
asession("2a6e492cb6982f230e48cf46023e2e4f")

# no session info
asession("pbiecek/graphGallery/2a6e492cb6982f230e48cf46023e2e4f")
# nice session info
asession("pbiecek/graphGallery/7f3453331910e3f321ef97d87adb5bad")

## End(Not run)
```

---

atrace *Add Tracing For All Objects Created By Given Function*

Description

`atrace` add call to saveToLocalRepo at the end of a given function.

Usage

```
atrace(FUN = "lm", object = "z")
```

Arguments

`FUN`	name of a function to be traced (character)
`object`	name of an object that should be traced (character)

Details

Function `atrace` calls the trace function.

Author(s)

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

See Also

Other archivist: `Repository`, `Tags`, `%a%`, `addHooksToPrint`, `addTagsRepo`, `aformat`, `ahistory`, `alink`, `aoptions`, `archivist-package`, `aread`, `asearch`, `asession`, `cache`, `copyLocalRepo`, `createLocalRepo`, `createMDGallery`, `deleteLocalRepo`, `getRemoteHook`, `getTagsLocal`, `loadFromLocalRepo`, `md5hash`, `restoreLibs`, `rmFromLocalRepo`, `saveToLocalRepo`, `searchInLocalRepo`, `setLocalRepo`, `shinySearchInLocalRepo`, `showLocalRepo`, `splitTagsLocal`, `summaryLocalRepo`, `zipLocalRepo`

Examples

```
# read the object from local directory
createLocalRepo("arepo_test", default=TRUE)
atrace("lm", "z")
lm(Sepal.Length~Sepal.Width, data=iris)
asearch("class:lm")
untrace("lm")
```

---

cache *Enable Caching of the Function Results with the use of Archivist*

Description

cache function stores all results of function calls in local Repository. All results are stored together with md5 hashes of the function calls. If a function is called with the same arguments, then its results can be loaded from the repository.

One may specify expiration date for live objects. It may be useful for objects that can be changed externally (like queries to database).

Usage

```
cache(cacheRepo = NULL, FUN, ..., notOlderThan = NULL)
```

Arguments

cacheRepo	A repository used for storing cached objects.
FUN	A function to be called.
...	Arguments of FUN function .
notOlderThan	load an artifact from the database only if it was created after notOlderThan.

Details

cache function stores all results of function calls in local Repository specified by the cacheRepo argument. The md5 hash of FUN and it's arguments is added as a Tag to the repository. This Tag has the following structure "cacheId:md5hash". Note that cache is a good solution if objects are not that big but calculations are time consuming (see Examples). If objects are big and calculations are easy, then disk input-output operations may take more time than calculations itself.

Value

Result of the function call with additional attributes: tags - md5 hash of the function call and call - "".

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

Przemyslaw Biecek, <Przemyslaw.Biecek@gmail.com>

See Also

For more detailed information, check the **archivist** package Use Cases.

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
# objects preparation
library(lubridate)
cacheRepo <- tempfile()
createLocalRepo( cacheRepo )

## Example 1:
# cache is useful when objects used by FUN are not that big but calculations
# are time-comsuming. Take a look at this example:
fun <- function(n) {replicate(n, summary(lm(Sepal.Length~Species, iris))$r.squared)}

# let's check time of two evaluations of cache function
system.time( res <- cache(cacheRepo, fun, 1000) )
system.time( res <- cache(cacheRepo, fun, 1000) )
# The second call is much faster. Why is it so? Because the result of fun
# function evaluation has been stored in local cacheRepo during the first evaluation
# of cache. In the second call of cache we are simply loading the result of fun
# from local cacheRepo Repository.

## Example 2:
testFun <- function(x) {cat(x);x}

# testFun will be executed and saved to cacheRepo
tmp <- cache(cacheRepo, testFun, "Say hallo!")

# testFun execution will be loaded from repository
tmp <- cache(cacheRepo, testFun, "Say hallo!")

# testFun will be executed once again as it fails with expiration date. It will
# be saved to cacheRepo.
tmp <- cache(cacheRepo, testFun, "Say hallo!", notOlderThan = now())

# testFun execution will be loaded from repository as it
# passes with expiration date [within hour]
tmp <- cache(cacheRepo, testFun, "Say hallo!", notOlderThan = now() - hours(1))

deleteLocalRepo( cacheRepo, TRUE)
```

```
rm( cacheRepo )
```

---

copyLocalRepo *Copy an Existing Repository into Another Repository*

Description

copy*Repo copies artifacts from one Repository into another Repository. It adds new files to existing gallery folder in repoTo Repository. copyLocalRepo copies local Repository while copyRemoteRepo copies remote Repository.

Usage

```
copyLocalRepo(repoFrom = NULL, repoTo, md5hashes)

copyRemoteRepo(repoTo, md5hashes, repo = aoptions("repo"),
  user = aoptions("user"), branch = aoptions("branch"),
  subdir = aoptions("subdir"), repoType = aoptions("repoType"))
```

Arguments

repoFrom	While copying local repository. A character that specifies the directory of the Repository from which artifacts will be copied. If it is set to NULL (by default), it will use the repoDir specified in setLocalRepo.
repoTo	A character that specifies the directory of the Repository into which artifacts will be copied.
md5hashes	A character vector containing md5hashes of artifacts to be copied.
repo	While coping the remote repository. A character containing a name of the remote repository on which the "repoFrom" - Repository is archived. By default set to NULL - see Note.
user	While coping the remote repository. A character containing a name of the remote user on whose account the "repoFrom" - Repository is created. By default set to NULL - see Note.
branch	While coping with the remote repository. A character containing a name of Remote Repository's branch on which the "repoFrom" - Repository is archived. Default branch is master.
subdir	While working with the remote repository. A character containing a name of a directory on the remote repository on which the "repoFrom" - Repository is stored. If the Repository is stored in the main folder on the remote repository, this should be set to FALSE as default.
repoType	A character containing a type of the remote repository. Currently it can be 'Remote' or 'bitbucket'.

Details

Functions copyLocalRepo and copyRemoteRepo copy artifacts from the archivist's Repositories stored in a local folder or on the Remote. Both of them use md5hashes of artifacts which are to be copied in md5hashes parameter. For more information about md5hash see md5hash.

Note

If repo and user are set to NULL (as default) in remote mode then global parameters set in setRemoteRepo function are used. If one would like to copy whole Repository we suggest to extract all md5hashes in this way unique(showLocalRepo(repoDir)[,1]).

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:


## Using archivist remote Repository to copy artifacts
# creating example Repository

exampleRepoDir <- tempfile()
createLocalRepo( exampleRepoDir )

# Searching for md5hashes of artifacts (without data related to them)
# in the archivist remote  Repository
hashes <- searchInRemoteRepo( pattern="name", user="pbiecek", repo="archivist", fixed=FALSE )

# Copying selected artifacts from archivist Remote  Repository into exampleRepoDir Repository

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

# See how the gallery folder in our exampleRepoDir Repository
# with copies of artifacts from archivist Remote  Repository looks like
list.files( path = file.path( exampleRepoDir, "gallery" ) )

# See how the backpack database in our exampleRepoDir Repository looks like
showLocalRepo( repoDir = exampleRepoDir )

# removing an example Repository
```

```
deleteLocalRepo( exampleRepoDir, deleteRoot=TRUE )

rm( exampleRepoDir )

# many archivist-like Repositories on one Remote repository

dir <- paste0(getwd(), "/ex1")
createLocalRepo( dir )
copyRemoteRepo( repoTo = dir , md5hashes = "ff575c261c949d073b2895b05d1097c3",
                user="MarcinKosinski", repo="Museum",
                branch="master", subdir="ex2")

# Check if the copied artifact is on our dir Repository

showLocalRepo( repoDir = dir) # It is in backpack database indeed
list.files( path = file.path( dir, "gallery" ) ) # it is also in gallery folder

# removing an example Repository
deleteLocalRepo( dir, TRUE)

rm(dir)

## Using graphGallery Repository attached to the archivist package to copy artifacts

# creating example Repository

exampleRepoDir <- tempfile()
createLocalRepo( exampleRepoDir )

# Searching for md5hashes of artifacts (without data related to them)
# in the graphGallery  Repository
archivistRepo <- system.file( "graphGallery", package = "archivist")
# You may use:
# hashes <- unique(showLocalRepo(repoDir)[,1])
# to extract all artifacts from repository
hashes <- searchInLocalRepo( pattern="name",
                             repoDir =  archivistRepo,
                             fixed=FALSE )

# Copying selected artifacts from archivist Remote  Repository into exampleRepoDir Repository

copyLocalRepo( repoFrom = archivistRepo, repoTo = exampleRepoDir , md5hashes= hashes )

# See how the backpack database in our exampleRepoDir Repository looks like
showLocalRepo( repoDir = exampleRepoDir )

# removing an example Repository

deleteLocalRepo( exampleRepoDir, deleteRoot=TRUE )

rm( exampleRepoDir )
rm( archivistRepo )
```

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

---

createLocalRepo *Create an Empty Repository*

Description

createLocalRepo creates an empty Repository in the given directory in which archived artifacts will be stored.

Usage

```
createLocalRepo(repoDir, force = TRUE, default = FALSE)

createEmptyRepo(...)
```

Arguments

repoDir	A character that specifies the directory for the Repository which is to be made.
force	If force = TRUE and repoDir parameter specifies the directory that doesn't exist, then function call will force to create new repoDir directory. Default set to force = TRUE.
default	If default = TRUE then repoDir is set as default Local Repository.
...	All arguments are being passed to createLocalRepo.

Details

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

All artifacts which are 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, each 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 is produced by 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. (the so called 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. As a result, the artifact is available immediately in backpack.db database for other collaborators.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir, default =  TRUE )
data(iris)
saveToLocalRepo(iris)
showLocalRepo()
showLocalRepo(method = "tags")
deleteLocalRepo( repoDir = exampleRepoDir, unset = TRUE, deleteRoot = TRUE)

## End(Not run)
```

`createMDGallery` *Create the Summary for Each Artifact in a Markdown Format*

Description

`createMDGallery` creates a summary for each artifact from Repository stored on a GitHub. For each artifact tihd function creates a markdown file with: the download link, artifact's Tags (when `addTags = TRUE`) and miniature (`addMiniature = TRUE`) if the artifact was archived with it's miniature and `Tags`. The miniature is a print or head over an artifact or it's `png` when it was a plot. But this function only supports `png` miniatures.

Usage

```
createMDGallery(output, repo = aoptions("repo"), user = aoptions("user"),
  branch = aoptions("branch"), subdir = aoptions("subdir"),
  repoType = aoptions("repoType"), addTags = FALSE, addMiniature = FALSE,
  maxTags = 100)
```

Arguments

output	A name of the file in which artifacts should be summarized.
repo	A character containing a name of the Remote repository on which the Repository is stored. By default set to NULL - see Note.
user	A character containing a name of the Github user on whose account the repo is created. By default set to NULL - see Note.
branch	A character containing a name of the Remote Repository's branch on which the Repository is stored. Default branch is master.
subdir	A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in the main folder of the Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.
addTags	Logical, whether to add artfiact's Tags to the output.
addMiniature	Logical, whether to add artfiact's miniature/plots to the output.
maxTags	Integer. The maximal length of chunks output when describing Tags of artifact.

Details

To learn more about artifacts visit archivist-package.

Note

If repo and user are set to NULL (as default) in the Remote mode then global parameters set in setRemoteRepo (or via aoptions) function are used.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Markdown example: https://github.com/pbiecek/archivist/issues/144#issuecomment-174192366

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:

createMDGallery(user = 'MarcinKosinski', repo = 'Museum',
 'README_test1.md', addTags = TRUE)
createMDGallery('graphGallery', 'pbiecek', addMiniature = TRUE,
 'README_test2.md', addTags = TRUE)


## End(Not run)
```

---

deleteLocalRepo *Delete the Existing Repository from the Given Directory*

Description

deleteLocalRepo deletes the existing Repository from the given directory. As a result all artifacts from gallery folder are removed and database backpack.db is deleted.

Usage

```
deleteLocalRepo(repoDir, deleteRoot = FALSE, unset = FALSE)

deleteRepo(...)
```

Arguments

repoDir	A character that specifies the directory for the Repository which is to be deleted.
deleteRoot	A logical value that specifies if the repository root directory should be deleted for Local Repository.
unset	A logical. If deleted repoDir was set to be default Local Repository and unset is TRUE, then repoDir is unset as a default Local Repository (aoptions('repoDir/repo', NULL, T)).
...	All arguments are being passed to deleteLocalRepo.

Note

Remember that using tempfile() instead of tempdir() in examples section is crucial. tempdir() is existing directory in which R works so calling deleteLocalRepo(exampleRepoDir, deleteRoot=TRUE) removes important R files. You can find out more information about this problem at stackoverflow webpage.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir, default =  TRUE )
data(iris)
saveToLocalRepo(iris)
deleteLocalRepo( repoDir = exampleRepoDir, unset = TRUE, deleteRoot = TRUE)
```

---

getRemoteHook *Get http Hook for Remote Repo*

Description

getRemoteHook returns http adress of the remote Repository. Then it can be used to download artifacts from the remote Repository.

Usage

```
getRemoteHook(repo = aoptions("repo"), user = aoptions("user"),
  branch = aoptions("branch"), subdir = aoptions("subdir"),
  repoType = aoptions("repoType"))
```

Arguments

repo	A character containing a name of a Git repository on which the Repository is archived.
user	A character containing a name of a Git user on whose account the repo is created.
branch	A character containing a name of Git Repository's branch on which the Repository is archived. Default branch is master.
subdir	A character containing a name of a directory on Git repository on which the Repository is stored. If the Repository is stored in main folder on Git repository, this should be set to subdir = "/" as default.

repoType — A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# objects preparation
getRemoteHook("graphGallery", "pbiecek")

## End(Not run)
```

---

getTagsLocal *Return Tags Corresponding to md5hash*

Description

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

Usage

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

getTagsRemote(md5hash, repo = aoptions("repo"), user = aoptions("user"),
  branch = aoptions("branch"), subdir = aoptions("subdir"),
  repoType = aoptions("repoType"), tag = "name")
```

Arguments

md5hash	A character containing md5hash of artifacts which Tags are desired to be returned.
repoDir	A character denoting an existing directory in which artifacts are stored.
tag	A regular expression denoting type of a Tag that we search for (see Examples). Default tag = "name".
repo	While working with the Remote repository. A character containing a name of the Remote repository on which the Repository is stored. By default set to NULL - see Note.
user	While working with the Remote repository. A character containing a name of the Remote user on whose account the repo is created. By default set to NULL - see Note.
branch	While working with the Remote repository. A character containing a name of the Remote repository's branch on which the Repository is stored. Default branch is master.
subdir	While working with the Remote repository. A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in main folder on the Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Details

getTagsLocal and getTagsRemote return Tags, of a specific type described by tag parameter, related to md5hash of an artifact. To learn more about artifacts visit archivist-package.

Value

The character vector of Tags (see Tags) related to md5hash of an artifact.

Note

If repo and user are set to NULL (as default) in Remote mode then global parameters set in setRemoteRepo function are used.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, loadFromLocalRepo,

md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
### Local version

## EXAMPLE with pipe operator %a%

# Creating empty repository
exampleRepoDir <- tempfile()
createLocalRepo( exampleRepoDir )

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

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

# We search for a Tag with default "name" regular expression corresponding to
# hash md5hash.
getTagsLocal( md5hash = hash, exampleRepoDir )

# Deleting example respository
deleteLocalRepo( exampleRepoDir, TRUE)
rm( exampleRepoDir )

## EXAMPLE with data iris
exampleRepoDir <- tempfile()
createLocalRepo( exampleRepoDir )

data(iris)
saveToRepo(iris, repoDir = exampleRepoDir )
showLocalRepo(exampleRepoDir)
showLocalRepo(exampleRepoDir, method = "tags")

# We can notice that there is only one md5hash
# (and second for archiveSessionInfo) in repo so we will use it
hash <- showLocalRepo(exampleRepoDir)[1,1]
```

```
# We search for a Tag with "varname" regular expression corresponding to
# hash md5hash.
getTagsLocal( md5hash = hash, exampleRepoDir, tag = "varname" )
# There are 5 different Tags with "varname" regular expression

# We needn't use the whole expression "varname". We may use its abbreviation
# and get the same result.
getTagsLocal( md5hash = hash, exampleRepoDir, tag = "varna" )

deleteLocalRepo( exampleRepoDir, TRUE)
rm( exampleRepoDir )

### Remote version
## EXAMPLE: pbiecek archivist repository on GitHub

showRemoteRepo(user="pbiecek", repo="archivist")
# We search for a Tag with default "name" regular expression corresponding to
# "cd6557c6163a6f9800f308f343e75e72" md5hash.
getTagsRemote( "cd6557c6163a6f9800f308f343e75e72",
               user="pbiecek", repo="archivist")

## EXAMPLE: many archivist-like Repositories on one Github repository
# We search for a Tag with default "name" regular expression corresponding to
# "ff575c261c949d073b2895b05d1097c3" md5hash.
getTagsRemote("ff575c261c949d073b2895b05d1097c3", user="MarcinKosinski",
              repo="Museum", branch="master", subdir="ex1")
```

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

Description

`loadFromLocalRepo` loads an artifact from a local Repository into the workspace. `loadFromRemoteRepo` loads an artifact from a github / git / mercurial Repository into the workspace. To learn more about artifacts visit archivist-package.

Usage

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

loadFromRemoteRepo(md5hash, repo = aoptions("repo"),
  user = aoptions("user"), branch = aoptions("branch"),
  subdir = aoptions("subdir"), repoType = aoptions("repoType"),
  value = FALSE)
```

Arguments

md5hash	A character assigned to the artifact through the use of a cryptographical hash function with MD5 algorithm, or it's abbreviation.
repoDir	A character denoting an existing directory from which an artifact will be loaded.
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).
repo	While working with a Remote repository. A character containing a name of a Remote repository on which the Repository is archived. By default set to NULL - see Note.
user	While working with a Remote repository. A character containing a name of a Remote user on whose account the repo is created. By default set to NULL - see Note.
branch	While working with a Remote repository. A character containing a name of Remote Repository's branch on which the Repository is archived. Default branch is master.
subdir	While working with a Remote repository. A character containing a name of a directory on Remote repository on which the Repository is stored. If the Repository is stored in main folder on Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'Remote' or 'bitbucket'.

Details

Functions loadFromLocalRepo and loadFromRemoteRepo load artifacts from the archivist Repositories stored in a local folder or on git. Both of them take md5hash as a parameter, which is a result of saveToRepo function. For each artifact, md5hash is a unique string of length 32 that is produced by 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 md5hash abbreviation will be loaded from Repository.

Note that user and repo should be used only when working with a git 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 git mode.

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

Note

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

If repo and user are set to NULL (as default) in Remote mode then global parameters set in setRemoteRepo function are used.

You should remember while using loadFromRepo wrapper that repoDir is a parameter used only in loadFromLocalRepo while repo, user, branch and subdir are used only in loadFromRemoteRepo. When you mix those parameters you will receive an error message.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# objects preparation

#' exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)
data(iris)
saveToLocalRepo(iris, repoDir=exampleRepoDir, archiveSessionInfo = TRUE)
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

loadFromLocalRepo(md5hash = '7f3453331910e3f321ef97d87adb5bad',
  repoDir = system.file("graphGallery", package = "archivist"), value = TRUE) -> pl
deleteLocalRepo(exampleRepoDir, TRUE)
rm(exampleRepoDir)


#
#Remote Version
#

# check the state of the Repository
summaryRemoteRepo( user="pbiecek", repo="archivist" )
showRemoteRepo( user="pbiecek", repo="archivist" )
showRemoteRepo( user="pbiecek", repo="archivist", method = "tags" )

rm( model )
```

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


loadFromRemoteRepo( "ff575c261c", user="pbiecek", repo="archivist")
NewObjects <- loadFromRemoteRepo( NAMEmd5hash, user="pbiecek", repo="archivist", value = TRUE )
loadFromRemoteRepo( CLASSmd5hash, user="pbiecek", repo="archivist")


## Loading artifacts from the repository which is built in the archivist package
## and saving them on the example repository

# Creating an example Repository - on which artifacts loaded from the
# archivist package repository will be saved
exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)

# Directory of the archivist package repository
repo_archivist <- system.file("graphGallery", package = "archivist")

# We are checking what kind of objects
# are stored in the archivist package repository
summaryLocalRepo(repoDir = repo_archivist)

# Let's say that we are interested in
# an artifact of class ggplot.
GGPLOTmd5hash <- searchInLocalRepo(pattern = "class:ggplot",
                                   repoDir = repo_archivist)
# There are eight of them.
# We load the first one by its value (parameter value = TRUE)
# and assign it to the p variable.
p <- loadFromLocalRepo(GGPLOTmd5hash[1], repoDir = repo_archivist,
                       value = TRUE)

# Finally, we may save the artifact on the example Repository.
# Note that md5hash is different from the one which is stored in
# the archivist package repository.
saveToRepo(p, repoDir = exampleRepoDir)

# Making sure that the artifact is stored on the example repository
showLocalRepo(repoDir = exampleRepoDir, method = "tags")

# removing an example Repository

deleteLocalRepo( exampleRepoDir, TRUE)

rm( exampleRepoDir )
```

```
# many archivist-like Repositories on one Remote repository

loadFromRemoteRepo( "ff575c261c949d073b2895b05d1097c3",
user="MarcinKosinski", repo="Museum", branch="master", subdir="ex2")


loadFromRemoteRepo( "ff575c261c949d073b2895b05d1097c3",
                    user="MarcinKosinski", repo="Museum", branch="master",
                    subdir="ex1")

#github
loadFromRemoteRepo(md5hash = "08dc0b66975cded92b5cd8291ebdc955",
               repo = "graphGallery", user = "pbiecek",
               repoType = "github", value = TRUE)

#git
loadFromRemoteRepo(md5hash = "08dc0b66975cded92b5cd8291ebdc955",
               repo = "graphGalleryGit", user = "pbiecek",
               repoType = "bitbucket", value = TRUE)

# mercurial
loadFromRemoteRepo(md5hash = "08dc0b66975cded92b5cd8291ebdc955",
               repo = "graphGalleryM", user = "pbiecek",
               repoType = "bitbucket", value = TRUE)

## 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 each artifact, `md5hash` is a unique string of length 32 that is produced by digest function which uses a cryptographical MD5 hash algorithm. The `md5hash` of each artifact that is archived in the Repository is also saved on the Repository along with the artifact's `Tags` - see Tags. It enables to distinguish artifacts in the Repository and facilitates searching and loading them.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

See Also

Functions that take `md5hash` as a parameter are:

- addTagsRepo,
- copyLocalRepo,
- copyRemoteRepo,
- loadFromLocalRepo,
- loadFromRemoteRepo,
- getTagsRemote,
- getTagsLocal,
- rmFromLocalRepo.

Functions returning `md5hash` as a value are:

- saveToLocalRepo,
- searchInLocalRepo,
- searchInRemoteRepo,
- shinySearchInLocalRepo.

Functions returning `md5hashes` as a `data.frame` are:

- showLocalRepo,
- showRemoteRepo.

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

Other archivist: `Repository`, `Tags`, `%a%`, `addHooksToPrint`, `addTagsRepo`, `aformat`, `ahistory`, `alink`, `aoptions`, `archivist-package`, `aread`, `asearch`, `asession`, `atrace`, `cache`, `copyLocalRepo`, `createLocalRepo`, `createMDGallery`, `deleteLocalRepo`, `getRemoteHook`, `getTagsLocal`, `loadFromLocalRepo`, `restoreLibs`, `rmFromLocalRepo`, `saveToLocalRepo`, `searchInLocalRepo`, `setLocalRepo`, `shinySearchInLocalRepo`, `showLocalRepo`, `splitTagsLocal`, `summaryLocalRepo`, `zipLocalRepo`

`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 a folder with an SQLite database stored in a file named `backpack` and a subdirectory named `gallery`.

`backpack` contains two tables:*artifact* and *tag*. *artifact* table consists of three columns:

- `md5hash`,
- `name`,
- `createdDate`,

while *tag* table consists of the following three columns:

- `artifact`,
- `tag`,
- `createdDate`.

`gallery` collects the following objects:

- artifacts and artifacts' data saved as `.rda` files,
- artifacts' miniatures saved as `.txt` and `.png` files.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

See Also

Functions using `Repository` are:

- addTagsRepo,
- ahistory,
- aread,
- asearch,
- cache,
- getTagsLocal,
- getTagsRemote,
- splitTagsLocal,
- splitTagsRemote,
- loadFromLocalRepo,
- loadFromRemoteRepo,
- rmFromLocalRepo,
- saveToRepo,
- searchInLocalRepo,
- searchInRemoteRepo,
- shinySearchInLocalRepo,

- showLocalRepo,
- showRemoteRepo,
- summaryLocalRepo,
- summaryRemoteRepo.

Function creating `Repository` is:

- createLocalRepo.

Function deleting `Repository` is:

- deleteLocalRepo.

Functions coping `Repository` are:

- copyLocalRepo,
- copyRemoteRepo.

Functions creating a zip archive from an existing `Repository` are:

- zipLocalRepo,
- zipRemoteRepo.

Functions setting global path to the `Repository` are:

- setLocalRepo,
- setRemoteRepo.

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

Other archivist: `Tags`, `%a%`, `addHooksToPrint`, `addTagsRepo`, `aformat`, `ahistory`, `alink`, `aoptions`, `archivist-package`, `aread`, `asearch`, `asession`, `atrace`, `cache`, `copyLocalRepo`, `createLocalRepo`, `createMDGallery`, `deleteLocalRepo`, `getRemoteHook`, `getTagsLocal`, `loadFromLocalRepo`, `md5hash`, `restoreLibs`, `rmFromLocalRepo`, `saveToLocalRepo`, `searchInLocalRepo`, `setLocalRepo`, `shinySearchInLocalRepo`, `showLocalRepo`, `splitTagsLocal`, `summaryLocalRepo`, `zipLocalRepo`

`restoreLibs` *Restore Versions of Libraries*

Description

Function `restoreLibs` gets either `session info` or artifact's `md5hash` and restore libraries/packages to versions attached when the object was saved in the repo. Typical use case is following. We have saved an object and now we are restoring it, but with current version of packages something is not working. The function `restoreLibs()` reverts all libraries that were attached previously to their previous versions.

Usage

```
restoreLibs(md5hash, session_info = NULL, lib.loc = NULL)
```

Arguments

md5hash	One of the following (see aread): A character vector which elements are consisting of at least three components separated with '/': Remote user name, Remote repository and name of the artifact (MD5 hash) or it's abbreviation. MD5 hashes of artifacts in current local default directory or its abbreviations.
session_info	Object with versions of packages to be installed. If not supplied then it will be extracted from md5hash md5hash
lib.loc	A character vector describing the location of R library trees to restore archived libraries, or NULL. The default value of NULL corresponds to all libraries currently known to .libPaths(). Non-existent library trees are silently ignored.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com> \ Przemyslaw Biecek, <przemyslaw.biecek@gmail.com>

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
## objects preparation
restoreLibs(md5hash = "pbiecek/graphGallery/7f3453331910e3f321ef97d87adb5bad")

## End(Not run)
```

---

rmFromLocalRepo *Remove an Artifact Given as a md5hash from the Repository*

Description

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

Usage

```
rmFromLocalRepo(md5hash, repoDir = aoptions("repoDir"), removeData = FALSE,
  removeMiniature = FALSE, force = FALSE, many = FALSE)

rmFromRepo(...)
```

Arguments

md5hash	A character assigned to the artifact through the use of a cryptographical hash function with MD5 algorithm, or it's abbreviation. This object will be removed. If many parameter is set to TRUE then md5hash will be a character vector.
repoDir	A character denoting an existing directory from which an artifact will be removed.
removeData	A logical value denoting whether to remove data along with the artifact specified by the md5hash. Defualt FALSE.
removeMiniature	A logical value denoting whether to remove a miniature along with the artifact specified by the md5hash. Defualt FALSE.
force	A logical value denoting whether to remove data related to more than one artifact. Defualt FALSE.
many	A logical value. To accelerate the speed of removing many objects, you can set this parameter to TRUE and pass a vector of artifacts' md5hashes to a md5hash parameter. It is not possible to use a vector of artifacts' md5hashes abbreviations - see Note. By default, set to FALSE.
...	All arguments are being passed to rmFromLocalRepo.

Details

rmFromLocalRepo removes an artifact given as a md5hash from the Repository. To be more precise, an artifact is removed both from backpack.db file(the SQLite database)and gallery subdirectory, where the artifacts are stored as md5hash.rda files.

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

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

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

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

Note

md5hash can be a result of the searchInLocalRepo function called by tag = NAME argument, where NAME is a Tag that describes the property of the artifacts to be deleted.

It is not possible to use a vector of artifacts' md5hashes abbreviations while using many = TRUE argument. This assumption was made to protect a user from removing, by accident, too many artifacts from the Repository.

For more information about Tags check Tags.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# objects preparation
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 - on which examples will work

exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)
myplot123Md5hash <- saveToLocalRepo(myplot123, repoDir=exampleRepoDir)
irisMd5hash <- saveToLocalRepo(iris, repoDir=exampleRepoDir)
modelMd5hash  <- saveToLocalRepo(model, repoDir=exampleRepoDir)
agn1Md5hash <- saveToLocalRepo(agn1, repoDir=exampleRepoDir)
fannyxMd5hash <- saveToLocalRepo(fannyx, repoDir=exampleRepoDir)

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

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

# remove examples

rmFromLocalRepo(fannyxMd5hash, repoDir = exampleRepoDir)
# removeData = FALSE default argument provides from removing archived
# fannyxMd5hash object's data from the Repository and the gallery
rmFromLocalRepo(irisMd5hash, repoDir = exampleRepoDir)

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

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


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

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

# but in the Repository database (backpack.db)
# there are three identical rows describing the object
# as well as three identical rows describing object's data.
```

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

# let's see how the Repository looks like: summary
summaryLocalRepo( exampleRepoDir )
# in spite of multiplying object's appearance in database it is

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

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

# rest of the artifacts can be removed for example by
# looking for dates of creation and then removing all objects
# created in a specific period of time

obj2rm <- searchInLocalRepo( pattern = list(dateFrom = Sys.Date(), dateTo = Sys.Date()),
                             repoDir = exampleRepoDir )
sapply(obj2rm, rmFromLocalRepo, repoDir = exampleRepoDir)
# above function call removed all objects which were created in these examples.
# Note that in the gallery folder there may be still some miniatures as
# removeMiniature parameter is set to FALSE

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

# one can also delete objects of a specific class
modelMd5hash  <- saveToLocalRepo(model, repoDir=exampleRepoDir)
model2Md5hash  <- saveToLocalRepo(model2, repoDir=exampleRepoDir)
model3Md5hash  <- saveToLocalRepo(model3, repoDir=exampleRepoDir)
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)

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


# one can remove object specifying only its md5hash abbreviation
(myplo123Md5hash <- saveToLocalRepo(myplot123, repoDir=exampleRepoDir))
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)

# If md5hash is "db50a4e667581f8c531acd78ad24bfee" then
# model abbreviation might be : "db50a"
# Note that with each evaluation of createEmptyRepo function new md5hashes
# are created. This is why, in your evaluation of the code, artifact
```

```
# myplo123Md5hash will have a different md5hash and the following
# instruction will result in an error.
rmFromLocalRepo("db40a", repoDir = exampleRepoDir, removeData = TRUE)
summaryLocalRepo( repoDir = exampleRepoDir )


# removing an example Repository

deleteLocalRepo( exampleRepoDir, TRUE)


######
######
REMOVING MANY ARTIFACTS
######
######

data(iris)

# 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 so that we may see it on our computer

exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir, force = TRUE)
saveToLocalRepo( iris, repoDir=exampleRepoDir)
saveToLocalRepo( model, repoDir=exampleRepoDir )
saveToLocalRepo( agn1, repoDir=exampleRepoDir )
saveToLocalRepo( fannyx, repoDir=exampleRepoDir )
saveToLocalRepo( lda1, repoDir=exampleRepoDir )
saveToLocalRepo( glmnet1, repoDir=exampleRepoDir )

ArtifactsAndData <- unique(showLocalRepo(repoDir = exampleRepoDir)[,1])
ArtifactsData <- unique(searchInLocalRepo(pattern = "relationWith", fixed = FALSE,
                                  repoDir = exampleRepoDir))
Artifacts <- setdiff(ArtifactsAndData, ArtifactsData)


# Removing many artifacts with many = TRUE argument
rmFromLocalRepo(Artifacts, repoDir = exampleRepoDir, many = TRUE)

# We may notice, in two ways, that artifacts' data is still in "exampleRepoDir".
# Either we may look into gallery folder of "exampleRepoDir"
list.files(file.path(exampleRepoDir, "gallery"))
# or show how database.db file looks like.
showLocalRepo(repoDir = exampleRepoDir) # artifacts' data is there indeed!

# If we want to remove artifact's data now we simply call rmFromLocalRepo function
# with removeData = TRUE additional argument.
rmFromLocalRepo(Artifacts, repoDir = exampleRepoDir, removeData = TRUE,  many = TRUE)

# We receive a warning as Artifacts are no longer in the repository.
# However, let's check what happened with Artifact's data.
showLocalRepo(repoDir = exampleRepoDir) # They were removed.
# Perhaps you may think that "exampleRepoDir" is empty as database indicates. However,
# if you look into gallery folder there will be some ".txt" or ".png" files.
list.files(file.path(exampleRepoDir, "gallery"))

# Those are probably, the so called, Miniatures. Let's try to remove them.
# In order to do it we call rmFromLocalRepo function with removeMiniature = TRUE argument.
rmFromLocalRepo(Artifacts, many = TRUE, repoDir = exampleRepoDir, removeMiniature = TRUE)

# Again we receive a warning as Artifacts are no longer in the repository but ...
list.files(file.path(exampleRepoDir, "gallery"))
# gallery folder is empty now! Artifact's miniature's were removed.


# Of course we may have done all these instructions by one simple function call.
# rmFromLocalRepo(Artifacts, many = TRUE, repoDir = exampleRepoDir,
```

```
#            removeData = TRUE, removeMiniature = TRUE)
# Nevertheless, it may be instructive to see how it is done step by step.

# removing an example Repository
deleteLocalRepo(repoDir = exampleRepoDir, deleteRoot = TRUE)

rm( exampleRepoDir )

## End(Not run)
```

---

saveToLocalRepo *Save an Artifact into a Repository*

Description

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

Usage

```
saveToLocalRepo(artifact, repoDir = aoptions("repoDir"), archiveData = TRUE,
  archiveTags = TRUE, archiveMiniature = TRUE, archiveSessionInfo = TRUE,
  force = TRUE, value = FALSE, ..., userTags = c(),
  silent = aoptions("silent"), ascii = FALSE,
  artifactName = deparse(substitute(artifact)))

saveToRepo(artifact, repoDir = aoptions("repoDir"), archiveData = TRUE,
  archiveTags = TRUE, archiveMiniature = TRUE, archiveSessionInfo = TRUE,
  force = TRUE, value = FALSE, ..., userTags = c(),
  silent = aoptions("silent"), ascii = FALSE,
  artifactName = deparse(substitute(artifact)))

asave(artifact, repoDir = aoptions("repoDir"), archiveData = TRUE,
  archiveTags = TRUE, archiveMiniature = TRUE, archiveSessionInfo = TRUE,
  force = TRUE, value = FALSE, ..., userTags = c(),
  silent = aoptions("silent"), ascii = FALSE,
  artifactName = deparse(substitute(artifact)))
```

Arguments

artifact	An arbitrary R artifact to be saved. For supported artifacts see details.
repoDir	A character denoting an existing directory in which an artifact will be saved.
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.

archiveSessionInfo
: A logical value denoting whether to archive the session info that describes the context in this given artifact was created.

force
: A logical value denoting whether to archive `artifact` if it was already archived in a Repository.

value
: A logical value. Should the result be (default `value = FALSE`) the `md5hash` of a stored artifact or should the result be an input artifact (`value = TRUE`), so that valueing code can be used. See examples.

...
: Graphical parameters denoting width and height of a miniature. See details. Further arguments passed to head. See Details section about `firtsRows` parameter

userTags
: A character vector with Tags. These Tags will be added to the repository along with the artifact.

silent
: If TRUE produces no warnings.

ascii
: A logical value. An `ascii` argument is passed to save function.

artifactName
: The name of the artifact with which it should be archived. If `NULL` then object's MD5 hash will be used instead.

Details

`saveToLocalRepo` 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 `saveToLocalRepo` call the database is refreshed, so the artifact is available immediately in the database for other collaborators. Each artifact is archived in a `md5hash.rda` file. This file will be saved in a folder (under `repoDir` directory) named `gallery`. For each artifact, `md5hash` is a unique string of length 32 that is produced by 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 loadFromRemoteRepo. 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 searchInRemoteRepo functions. Artifacts can be searched by their Tags, `names`, `classes` or `archiving date`.

`firstRows` parameter.

If the artifact is of class `data.frame` or user set `archiveData = TRUE` for artifact that stores data within it, it is possible to specify how many rows of that data (or that `data.frame`) should be archived in a `miniature`. This can be done by adding the argument `firstRows` with the n corresponding to the number of rows (as in head). Note that, the data can be extracted only from the artifacts that are supported by the **archivist** package; see Tags.

Graphical parameters.

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 listed here Tags.

Value

As a result of calling this function a character string is returned, which determines the md5hash of the artifact. If archiveData is TRUE, the result will also have an attribute, named data, which determines md5hash of the data needed to compute the artifact.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

In the following way one can specify his own Tags for artifacts by setting artifact's attribute before call of the saveToLocalRepo function: attr(x, "tags") = c("name1", "name2"), where x is an artifact and name1, name2 are Tags specified by a user. It can be also done in a new, simpler way by using userTags parameter like this:

- saveToLocalRepo(model, repoDir, userTags = c("my_model", "do not delete")).

Specifing additional Tags by attributes can be beneficial when one uses addHooksToPrint.

Important: if one wants to archive data from artifacts which is one of: survfit, glmnet, qda, lda, trellis, htest class, and this dataset is transformed within the artifact's formula then saveToLocalRepo will not archive this dataset. saveToLocalRepo only archives datasets that already exist in any of R environments.

Example: The data set will not be archived here.

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

Example: The data set will be archived here.

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

Author(s)

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

See Also

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

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)
data(swiss)
saveToLocalRepo(swiss, repoDir=exampleRepoDir, archiveSessionInfo = TRUE)
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

loadFromLocalRepo(md5hash = '2a6e492cb6982f230e48cf46023e2e4f',
  repoDir = system.file("graphGallery", package = "archivist"), value = TRUE) -> model

saveToLocalRepo(model, repoDir=exampleRepoDir,
            userTags = c("do not delete", "my favourite model"))
aoptions('repoDir', system.file("graphGallery", package = "archivist"))
showLocalRepo(method = "tags")
data(iris)
asave(iris, silent = FALSE) # iris was used in pl
aoptions('repoDir', NULL, unset = TRUE)
deleteLocalRepo(exampleRepoDir, TRUE)
rm(exampleRepoDir)
```

---

searchInLocalRepo *Search for an Artifact in the Repository Using Tags*

Description

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

Usage

```
searchInLocalRepo(pattern, repoDir = aoptions("repoDir"), fixed = TRUE,
  intersect = TRUE)

searchInRemoteRepo(pattern, repo = aoptions("repo"),
  user = aoptions("user"), branch = "master", subdir = aoptions("subdir"),
  repoType = aoptions("repoType"), fixed = TRUE, intersect = TRUE)

multiSearchInLocalRepo(...)

multiSearchInRemoteRepo(...)
```

Arguments

pattern	If fixed = TRUE: a character denoting a Tag which is to be searched for in the Repository. It is also possible to specify pattern as a list of length 2 with dateFrom and dateTo; see details. If fixed = FALSE: a regular expression

specifying the beginning of a Tag, which will be used to search for artifacts. If of length more than one and if intersect = TRUE then artifacts that match all conditions are returned. If intersect = FALSE then artifacts that match any condition are returned. See examples.

repoDir — A character denoting an existing directory in which artifacts will be searched for.

fixed — A logical value specifying how artifacts should be searched for. If fixed = TRUE (default) then artifacts are searched for by using pattern = "Tag" argument. If fixed = FALSE then artifacts are searched for by using pattern = "regular expression" argument. The latter is wider and more flexible method, e.g., using pattern = "name", fixed = FALSE arguments enables to search for all artifacts in the Repository.

intersect — A logical value. Used only when length(pattern) > 1 & is.character(pattern). See pattern for more details.

repo — While working with the Remote repository. A character containing a name of the Remote repository on which the Repository is stored. By default set to NULL - see Note.

user — While working with the Remote repository. A character containing a name of the Remote user on whose account the repo is created. By default set to NULL - see Note.

branch — While working with the Remote repository. A character containing a name of the Remote Repository's branch on which the Repository is stored. Default branch is master.

subdir — While working with the Remote repository. A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in the main folder of the Remote repository, this should be set to subdir = "/" as default.

repoType — A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

... — Used for old deprecated functions.

Details

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

Tags, used in a pattern parameter, should be determined according to the format: "TagKey:TagValue" - see examples.

Value

searchInRepo returns character vector of md5hashes of artifacts that were searched for. Those are hashes assigned to artifacts while they were saved in the Repository by the saveToLocalRepo function. If the artifact is not in the Repository then a logical value FALSE is returned.

Note

If repo, user, subdir and repoType are not specified in the Remote mode then global parameters set in setRemoteRepo function are used.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# objects preparation

  showLocalRepo(method = "md5hashes",
     repoDir = system.file("graphGallery", package = "archivist"))
  showLocalRepo(method = "tags",
     repoDir = system.file("graphGallery", package = "archivist"))

  # Tag search, fixed version
  searchInLocalRepo( "class:ggplot", repoDir = exampleRepoDir )
  searchInLocalRepo( "name:", repoDir = exampleRepoDir )
  # Tag search, regex version
  searchInLocalRepo( "class", repoDir = exampleRepoDir, fixed = FALSE )

  # Github version
  # check the state of the Repository
  summaryRemoteRepo( user="pbiecek", repo="archivist" )
  showRemoteRepo( user="pbiecek", repo="archivist" )
  showRemoteRepo( user="pbiecek", repo="archivist", method = "tags" )
  # Tag search, fixed version
```

```
searchInRemoteRepo( "varname:Sepal.Width", user="pbiecek", repo="archivist" )
searchInRemoteRepo( "class:lm", user="pbiecek", repo="archivist", branch="master" )
searchInRemoteRepo( "name:myplot123", user="pbiecek", repo="archivist" )

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

# also on Github

# Remeber to set dateTo parameter to actual date because sometimes we update datasets.
searchInRemoteRepo( pattern = list( dateFrom = "2015-10-01", dateTo = "2015-11-30" ),
                    user="pbiecek", repo="archivist", branch="master" )


# many archivist-like Repositories on one Remote repository

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

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

# multi versions
searchInRemoteRepo( pattern=c("varname:Sepal.Width", "class:lm", "name:myplot123"),
                        user="pbiecek", repo="archivist", intersect = FALSE )


## End(Not run)
```

`setLocalRepo` *Set Repository's Global Path*

Description

`setLocalRepo` sets local Repository's global path. `setRemoteRepo` similarly sets Remote Repository's path. See examples.

Usage

```
setLocalRepo(repoDir)

setRemoteRepo(user, repo, branch = "master", subdir = "/",
  repoType = "github")
```

Arguments

`repoDir`	A character denoting a directory of a Repository that we want to make dafault.
`user`	While working with the Remote repository. A character containing a name of the Remote user that we want to make default.

repo	While working with the Remote repository. A character containing a name of the Remote repository that we want to make default.
branch	While working with the Remote repository. A character containing a name of the Remote Repository's branch that we want to make default. Default branch is master.
subdir	While working with the Remote repository. A character containing a name of the Repository's directory on Remote that we want to make default. If the Repository is stored in the main folder on the Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Details

If you are working on a local Repository and you are tired of specifying repoDir parameter in every function call that uses this parameter, you can set Repository's path globally using setLocalRepo function and omit repoDir parameter in future function calls.

If you are working on the Remote Repository and you are tired of specifying user, repo, branch and subdir parameters in every function call that uses these parameters, you can set Remote Repository's path globally using setRemoteRepo function and omit user, repo, branch and subdir parameters in future function calls. See examples.

For local repositories, in this way, in the following function calls: loadFromLocalRepo,searchInLocalRepo, rmFromLocalRepo, zipLocalRepo, addTagsRepo, shinySearchInLocalRepo, getTagsLocal, showLocalRepo, summaryLocalRepo repoDir parameter may be omitted. For remote repositories, in this way, in the following function calls: zipRemoteRepo, loadFromRemoteRepo, searchInRemoteRepo, getTagsRemote, showRemoteRepo, summaryRemoteRepo, copyRemoteRepo parameters user, repo, branch, subdir may be omitted.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

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

See Also

https://github.com/pbiecek/archivist/wiki

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo,

createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
## Local version
exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)
setLocalRepo(exampleRepoDir)

data(iris)
data(swiss)

# From this moment repoDir parameter may be ommitted in the following functions
saveToRepo(iris)
saveToRepo(swiss)
showLocalRepo()
showLocalRepo(method = "tags")
iris2 <- loadFromLocalRepo( "ff575c2" , value = TRUE)
searchInLocalRepo("name:i", fixed = FALSE)
getTagsLocal("ff575c261c949d073b2895b05d1097c3")
rmFromLocalRepo("4c43f")
showLocalRepo()
summaryLocalRepo()

# REMEMBER that in deleteLocalRepo you MUST specify repoDir parameter!
# deleteRepo doesn't take setLocalRepo's settings into consideration
deleteLocalRepo( exampleRepoDir, deleteRoot=TRUE)
rm( exampleRepoDir )

## Github version
setRemoteRepo( user="MarcinKosinski", repo="Museum", branch="master",
               subdir="ex1" )

# From this moment user, repo, branch, subdir parameters may be ommitted
# in the following functions:
showRemoteRepo()
loadFromRemoteRepo( "ff575c261c949d073b2895b05d1097c3")
this <- loadFromRemoteRepo( "ff", value = TRUE)
zipRemoteRepo()
file.remove(file.path(getwd(), "repository.zip")) # We can remove this zip file
searchInRemoteRepo( "name:", fixed= FALSE)
getTagsRemote("ff575c261c949d073b2895b05d1097c3")
summaryRemoteRepo( )

# To use multisearchInRemoteRepo we should use repository with more than one artifact.
setRemoteRepo( user="pbiecek", repo="archivist"  )

# From this moment user and repo parameters may be ommitted in the following functions
showRemoteRepo()
searchInRemoteRepo( pattern=c("varname:Sepal.Width", "class:lm", "name:myplot123"),
```

```
                        intersect = FALSE )

## End(Not run)
```

---

shinySearchInLocalRepo

Shiny Based Live Search for an Artifact in a Repository Using Tags

---

Description

shinySearchInLocalRepo searches for an artifact in a Repository using Tags. To create an application one needs to point the name of artifacts' repository. The application is generated on the run. As for now there are two controllers exposed. A text input field and a slider. Tags that are typed into text field are used for searching in repository. Objects that have the same Tags are presented on the right panel. These object might be also downloaded just by click. To learn more about artifacts visit archivist-package.

Usage

```
shinySearchInLocalRepo(repoDir = NULL, host = "0.0.0.0")
```

Arguments

repoDir	A character denoting an existing directory in which artifacts will be searched. If set to NULL (by default), uses the repoDir specified in setLocalRepo.
host	A host IP adress, see the host argument in shiny::runApp.

Details

shinySearchInLocalRepo searches for artifacts in a Repository using their Tags (e.g., name, class or archiving date). Tags are submitted in a text input in a shiny application. Many Tags may be specified, they should be comma separated. User can specify more Tags like phase, project, author etc. when artifact is created.

In the search query one can add Tags starting with sort: or sort:-. As a result, miniatures will be sorted appropriately. For example sort:class will sort class Tags, while sort:-class will sort class tags backwards. sort:createdDate will sort createdDate Tag and sort:-createdDate will sort createdDate Tag backwards.

Tags, submitted in the text field, should be given according to the format: "TagKey:TagValue" - see examples.

Value

shinySearchInLocalRepo runs a shiny application.

shiny

This function use tools from the fantastic **shiny** package, so you'll need to make sure to have it installed.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
  # assuming that there is a 'repo' dir with a valid archivist repository
  shinySearchInLocalRepo( repoDir = 'repo' )

## End(Not run)
```

---

showLocalRepo *View the List of Artifacts from the Repository*

Description

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

Usage

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

showRemoteRepo(repo = aoptions("repo"), user = aoptions("user"),
  branch = aoptions("branch"), subdir = aoptions("subdir"),
  repoType = aoptions("repoType"), method = "md5hashes")
```

Arguments

repoDir	A character denoting an existing directory of the Repository for which metadata will be returned.
method	A character specifying a method to be used to show the Repository. Available methods: md5hashes (default), tags and sets - see archivist2::saveSetToRepo.

repo	While working with the Remote repository. A character containing a name of the Remote repository on which the Repository is stored. By default set to NULL - see Note.
user	While working with the Remote repository. A character containing a name of the Remote user on whose account the repo is created. By default set to NULL - see Note.
branch	While working with the Remote repository. A character containing a name of the Remote Repository's branch on which the Repository is stored. Default branch is master.
subdir	While working with the Remote repository. A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in the main folder of the Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Details

showLocalRepo and showRemoteRepo 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 showRemoteRepo shows the artifacts of the Repository existing on the remote 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.

Value

If parameter method is set as md5hashes then a data.frame with artifacts' names and artifacts'md5hashes will be returned.

If parameter method is set as tags then a data.frame with Tags and artifacts' md5hashes will be returned.

Also in both cases a data.frame contains an extra column with the date of creation of the Tag or md5hash.

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

Note

If repo and user are set to NULL (as default) in the Remote mode then global parameters set in setRemoteRepo (or via aoptions) function are used.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
# objects preparation

showLocalRepo(method = "md5hashes",
   repoDir = system.file("graphGallery", package = "archivist"))
showLocalRepo(method = "tags",
   repoDir = system.file("graphGallery", package = "archivist"))


# Remote version

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

# many archivist-like Repositories on one Remote repository

showRemoteRepo( user="MarcinKosinski", repo="Museum", branch="master",
                subdir="ex1")
showRemoteRepo( user="MarcinKosinski", repo="Museum", branch="master",
                subdir="ex2")

## Remote options
showRemoteRepo('archivist', 'pbiecek')
aoptions('user', 'pbiecek')
aoptions('repo', 'archivist')
loadFromRemoteRepo("ff575c261c", value = TRUE) -> iris123

showRemoteRepo('Museum', 'MarcinKosinski', subdir = 'ex1')
aoptions('repo', 'Museum')
aoptions('user', 'MarcinKosinski')
aoptions('subdir', 'ex1')
aoptions('branch', 'master')
showRemoteRepo()
showRemoteRepo(subdir = 'ex2')

aoptions('subdir')



## End(Not run)
```

splitTagsLocal *Split Tags in Repository*

Description

splitTagsLocal and splitTagsRemote functions split tag column from *tag* table placed in backpack.db into two separate columns: tagKey and tagValue.

Usage

```
splitTagsLocal(repoDir = aoptions("repoDir"))

splitTagsRemote(repo = aoptions("repo"), user = aoptions("user"),
  branch = aoptions("branch"), subdir = aoptions("subdir"),
  repoType = aoptions("repoType"))
```

Arguments

repoDir	While working with the local repository. A character denoting an existing directory of the Repository.
repo	While working with the Github repository. A character containing a name of the Github repository on which the Repository is stored. By default set to NULL - see Note.
user	While working with the Github repository. A character containing a name of the Github user on whose account the repo is created. By default set to NULL - see Note.
branch	While working with the Github repository. A character containing a name of the Github Repository's branch on which the Repository is stored. Default branch is master.
subdir	While working with the Github repository. A character containing a name of a directory on the Github repository on which the Repository is stored. If the Repository is stored in the main folder of the Github repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Details

tag column from *tag* table has normally the follwing structure: TagKey:TagValue. splitTagsLocal and splitTagsRemote functions can be used to split tag column into two separate columns: tagKey and tagValue. As a result functions from dplyr package can be used to easily summarize, search, and extract artifacts' Tags. See examples.

Value

A data.frame with 4 columns: artifact, tagKey, tagValue and createdDate, corresponding to the current state of Repository.

Note

If repo and user are set to NULL (as default) in the Github mode then global parameters set in setRemoteRepo function are used.

Sometimes we can use addTags* function or userTags parameter in saveToRepo to specify a Tag which might not match TagKey:TagValue structure. It is simply Tag. In this case tagKey = userTags and tagValue = Tag. See examples.

To learn more about Tags and Repository structure check Tags and Repository.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

Witold Chodor , <witoldchodor@gmail.com>

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:
## LOCAL VERSION

setLocalRepo(system.file("graphGallery", package = "archivist"))
head(showLocalRepo(method = "tags"))
head(splitTagsLocal() )

## Github Version
# Let's check how does table tag look like while we are using the
# Gitub repository.
# We will choose only special columns of data frames that show Tags
head(showRemoteRepo( user = "pbiecek", repo = "archivist", method = "tags" )[,2])
head(splitTagsRemote( user = "pbiecek", repo = "archivist" )[,2:3])

head(splitTagsRemote("PieczaraPietraszki", "BetaAndBit", "master", "UniwersytetDzieci/arepo"))

## End(Not run)
```

---

summaryLocalRepo *View the Summary of the Repository*

Description

summaryRepo summarizes the current state of the Repository.

Usage

```
summaryLocalRepo(repoDir = aoptions("repoDir"))

summaryRemoteRepo(repo = aoptions("repo"), user = aoptions("user"),
  branch = "master", subdir = aoptions("subdir"),
  repoType = aoptions("repoType"))
```

Arguments

repoDir	A character denoting an existing directory of the Repository for which a summary will be returned.
repo	While working with the Remote repository. A character containing a name of the Remote repository on which the Repository is stored. By default set to NULL - see Note.
user	While working with the Remote repository. A character containing a name of the Remote user on whose account the repo is created. By default set to NULL - see Note.
branch	While working with the Remote repository. A character containing a name of the Remote Repository's branch on which the Repository is stored. Default branch is master.
subdir	While working with the Remote repository. A character containing a name of a directory on the Remote repository on which the Repository is stored. If the Repository is stored in the main folder of the Remote repository, this should be set to subdir = "/" as default.
repoType	A character containing a type of the remote repository. Currently it can be 'github' or 'bitbucket'.

Details

summaryRepo summarizes 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 then it is counted as one artifact or database in print(summaryRepo).

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

If repo and user are set to NULL (as default) in the Remote mode then global parameters set in setRemoteRepo function are used.

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, zipLocalRepo

Examples

```
## Not run:

showLocalRepo(repoDir = system.file("graphGallery", package = "archivist"))
#
# Remote version
#

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

# many archivist-like Repositories on one Remote repository

summaryRemoteRepo(user="MarcinKosinski", repo="Museum",
branch="master", subdir="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. They can be the artifact's name, class or archiving date. Furthermore, for various artifact's classes more different Tags are available.

A Tag is represented as a string and usually has the following structure "TagKey:TagValue", e.g., "name:iris".

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

The list of supported artifacts which are divided thematically is presented below. The newest list is also available on **archivist** wiki on Github.

Regression Models

`lm`
- name
- class
- coefname
- rank
- df.residual
- date

`summary.lm`
- name
- class
- sigma
- df
- r.squared
- adj.r.squared
- fstatistic
- fstatistic.df
- date

`glmnet`
- name
- class
- dim
- nulldev
- npasses
- offset
- nobs
- date

`survfit`
- name
- class
- n
- type
- conf.type
- conf.int
- strata
- date

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`
- name
- class
- memb.exp
- dunn_coeff
- normalized dunn_coeff
- k.crisp
- objective
- tolerance
- iterations
- converged
- maxit
- clus.avg.widths
- avg.width
- date

`lda`
- name
- class
- N
- lev
- counts
- prior
- svd
- date

`qda`
- name
- class
- N
- lev
- counts
- prior
- ldet
- terms
- date

Statistical Tests

`htest`
- name
- class
- method

- data.name
- null.value
- alternative
- statistic
- parameter
- p.value
- conf.int.
- estimate
- date

When none of above is specified, Tags are assigned by default

`default`
- name
- class
- date

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

Note

In the following way one can specify his own `Tags` for artifacts by setting artifact's attribute before call of the `saveToLocalRepo` function: `attr(x, "tags" ) = c( "name1", "name2" )`, where `x` is an artifact and `name1, name2` are `Tags` specified by a user. It can be also done in a new, simpler way by using `userTags` parameter like this:

- `saveToLocalRepo(model, repoDir, userTags = c("my_model", "do not delete"))`.

Specifing additional `Tags` by attributes can be beneficial when one uses addHooksToPrint.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

See Also

Functions using `Tags` are:

- addTagsRepo
- getTagsLocal
- getTagsRemote
- saveToLocalRepo
- searchInLocalRepo,
- searchInRemoteRepo.

Other archivist: `Repository`, `%a%`, `addHooksToPrint`, `addTagsRepo`, `aformat`, `ahistory`, `alink`, `aoptions`, `archivist-package`, `aread`, `asearch`, `asession`, `atrace`, `cache`, `copyLocalRepo`, `createLocalRepo`, `createMDGallery`, `deleteLocalRepo`, `getRemoteHook`, `getTagsLocal`, `loadFromLocalRepo`, `md5hash`, `restoreLibs`, `rmFromLocalRepo`, `saveToLocalRepo`, `searchInLocalRepo`, `setLocalRepo`, `shinySearchInLocalRepo`, `showLocalRepo`, `splitTagsLocal`, `summaryLocalRepo`, `zipLocalRepo`

Examples

```
## Not run:
# examples
# data.frame object
data(iris)
exampleRepoDir <- tempfile()
createLocalRepo(repoDir = exampleRepoDir)
saveToLocalRepo( iris, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, deleteRoot=TRUE )

# 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( myplot123, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, deleteRoot=TRUE )

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width,
           data= iris)
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
asave( model, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( agn1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, 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)
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
```

```
saveToLocalRepo( fannyx, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
asave( lda1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( qda1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )


# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( glmnet1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# 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 run; 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( trellis.plot, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( this.test, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteLocalRepo( exampleRepoDir, TRUE )

# 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( myFit , repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )[,-3]
deleteLocalRepo( exampleRepoDir, TRUE)

# origin of the artifacts stored as a name - chaining code
library(dplyr)
```

```
exampleRepoDir <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
data("hflights", package = "hflights")
hflights %>%
  group_by(Year, Month, DayofMonth) %>%
  select(Year:DayofMonth, ArrDelay, DepDelay) %>%
  saveToLocalRepo( exampleRepoDir, value = 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) %>%
  saveToLocalRepo( exampleRepoDir )
  # chaining code is finished and after last operation the
  # artifact is stored
showLocalRepo( exampleRepoDir, "tags" )[,-3]
showLocalRepo( exampleRepoDir )
deleteLocalRepo( exampleRepoDir, TRUE)

rm( exampleRepoDir )

## End(Not run)
```

`zipLocalRepo` *Create a zip Archive From an Existing Repository*

Description

`zipLocalRepo` and `zipRemoteRepo` create a zip archive from an existing Repository. `zipLocalRepo` zips local `Repository`, `zipRemoteRepo` zips `Repository` stored on Github.

Usage

```
zipLocalRepo(repoDir = aoptions("repoDir"), repoTo = getwd(),
  zipname = "repository.zip")

zipRemoteRepo(repoTo = getwd(), user = aoptions("user"),
  repo = aoptions("repo"), branch = "master", subdir = aoptions("subdir"),
  repoType = aoptions("repoType"), zipname = "repository.zip")
```

Arguments

`repoDir`	A character that specifies the directory of the Repository which will be zipped.
`repoTo`	A character that specifies the directory in which there will be created zip archive from `Repository` stored in `repoDir` or Remote directory. By default set to working directory (`getwd()`).

zipname	A character that specifies name of the zipped repository. It is assumed that this file does not exist or does not contain backpack.db file. An attempt to override will produce an error.
user	While working with the Remote repository. A character containing a name of the Remote user on whose account the repo is created. By default set to NULL - see Note.
repo	While working with the Remote repository. A character containing a name of the Remote repository on which the Repository, which is to be zipped, is archived. By default set to NULL - see Note.
branch	While working with the Remote repository. A character containing a name of the Remote repository's branch on which Repository, which is to be zipped, is archived. Default branch is master.
subdir	While working with a Remote repository. A character containing a name of a directory on Remote repository on which the Repository, which is to be zipped, is stored. If the Repository is stored in the main folder on the Remote repository, this should be set to FALSE as default.
repoType	A character containing a type of the remote repository. Currently it can be 'Remote' or 'bitbucket'.

Note

The function might not work if Rtools are not installed. To solve this problem follow these Instructions.

If repo and user are set to NULL (as default) in Github mode then global parameters set in setRemoteRepo function are used.

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

Author(s)

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

See Also

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo

Other archivist: Repository, Tags, %a%, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, 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 <- tempfile()
createLocalRepo( repoDir = exampleRepoDir )
saveToLocalRepo( myplot123, repoDir=exampleRepoDir )
saveToLocalRepo( iris, repoDir=exampleRepoDir )
saveToLocalRepo( model, repoDir=exampleRepoDir )



zipLocalRepo( exampleRepoDir )

deleteLocalRepo( exampleRepoDir, TRUE)

rm( exampleRepoDir )

# Remote version

zipRemoteRepo( user="MarcinKosinski",
repo="Museum", branch="master", subdir="ex1" )

zipRemoteRepo( user="pbiecek", repo="archivist", repoTo = getwd( ) )


## End(Not run)
```

Description

An extended pipe operator %>% from magrittr package version 1.0.1. Enables archiving artifacts with their chaining code - see examples and vignettes.

Usage

```
lhs %a% rhs
```

Arguments

lhs	An artifact that will be used as an argument of rhs by %a% operator.
rhs	A function call using lhs as an argument by %a% operator.

Details

The extension works as follows, the result of %a% operator is archived together with lhs (as an artifact) and rhs (as a Tag). This allows to present a history of an artifact. This option works only if a default repository is set.

Demonstration

This function is well explained on this http://r-bloggers.com/r-hero-saves-backup-city-with-archivist-and-github blog post.

Note

Bug reports and feature requests can be sent to https://github.com/pbiecek/archivist/issues

See Also

Other archivist: Repository, Tags, addHooksToPrint, addTagsRepo, aformat, ahistory, alink, aoptions, archivist-package, aread, asearch, asession, atrace, cache, copyLocalRepo, createLocalRepo, createMDGallery, deleteLocalRepo, getRemoteHook, getTagsLocal, loadFromLocalRepo, md5hash, restoreLibs, rmFromLocalRepo, saveToLocalRepo, searchInLocalRepo, setLocalRepo, shinySearchInLocalRepo, showLocalRepo, splitTagsLocal, summaryLocalRepo, zipLocalRepo

Examples

```
## Not run:

library(dplyr)

## Usage of %a% operator without setting default repository
# We will receive sepcial warning
iris %a% summary()

## Archiving artifacts with their chaining code
# Creating empty repository
exampleRepoDir <- tempfile()
createLocalRepo( exampleRepoDir, default = TRUE ) # Remember to set repo to default
```

```
# Start using %a% operator
data("hflights", package = "hflights")
hflights %a%
  group_by(Year, Month, DayofMonth) %a%
  select(Year:DayofMonth, ArrDelay, DepDelay) %a%
  summarise(arr = mean(ArrDelay, na.rm = TRUE),
            dep = mean(DepDelay, na.rm = TRUE)) %a%
  filter(arr > 30 | dep > 30)

# Let's check how Tags of subsequent artifacts look like
showLocalRepo()
getTagsLocal("a8ce013a8e66df222be278122423dc60", tag = "") #1
getTagsLocal("9d91fe67fd51f3bfdc9db0a596b12b38", tag = "") #2
getTagsLocal("617ded4953ac986524a1c24703363980", tag = "") #3
getTagsLocal("3f1ac0a27485be5d52e1b0a41d165abc", tag = "") #4
getTagsLocal("0cb04315482de73d7f5a1081953236f8", tag = "") #5
getTagsLocal("5629bc43e36d219b613076b17c665eda", tag = "") #6

# Deleting existing repository
deleteLocalRepo(exampleRepoDir, deleteRoot = TRUE)
rm(exampleRepoDir)

## End(Not run)
```

Index