Package 'D4TAlink.light'

September 11, 2025

Type Package

Title GDP - Workflow Management

Version 2.1.21

Date 2025-09-09

Author Gregoire Thomas [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6247-9438>)

Maintainer Gregoire Thomas <gregoire.thomas@SQU4RE.com>

Description Tools, methods and processes for the management
of analysis workflows. These lightweight solutions facilitate
structuring R&D activities. These solutions were developed to comply
with Good Documentation Practice (GDP),
with ALCOA+ principles as proposed by the U.S. FDA, and
with FAIR principles as discussed by Jacobsen et al. (2017) <doi:10.1162/dint_r_00024>.

Depends R (>= 4.1.0), rjson, DBI, RSQLite, openssl

Imports officedown, utils, rmarkdown, quarto, Biobase,
openxlsx,reticulate,feather, getPass

Suggests knitr, roxygen2 (>= 3.1.0), WriteXLS,XLConnect,xlsx, testthat

License GPL-3

Encoding UTF-8

Config/testthat/edition 3

RoxygenNote 7.3.2

VignetteBuilder knitr

URL https://bitbucket.org/SQ4/d4talink.light

ByteCompile true

Repository CRAN

BugReports https://bitbucket.org/SQ4/d4talink.light/issues

NeedsCompilation no

Date/Publication 2025-09-11 00:10:02 UTC

Contents

`archiveTask` *Create an archive containing the files of a given task.*

Description

Create an archive containing the files of a given task.

Usage

```
archiveTask(task, file, overwrite = FALSE, ...)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
file	full name of the output zip file
overwrite	overwrite the output zip file is it exists
...	Arguments passed on to utils::zip zipfile The pathname of the zip file: tilde expansion (see path.expand) will be performed. files A character vector of recorded filepaths to be included. flags A character string of flags to be passed to the command: see 'Details'. extras An optional character vector: see 'Details'. zip A character string specifying the external command to be used.

Value

the archive file name invisibly.

---

binaryDir *Get path of binary directory.*

Description

Get path of binary directory.

Usage

```
binaryDir(task, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

binaryFn *Get path of binary file.*

Description

Get path of binary file.

Usage

```
binaryFn(task, type, ext = "rds", subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

catReport *Output R object using function cat.*

Description

Output R object using function cat.

Usage

```
catReport(
  x,
  task,
  type,
  ext = "txt",
  subdir = NULL,
  dirCreate = TRUE,
  sep = "\n",
  eof = "\n",
  ...
)
```

Arguments

x	R object to output.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
sep	separator
eof	EOF
...	R objects (see 'Details' for the types of objects allowed).

Value

the file name invisibly.

D4getenv *Get env var.*

Description

Get env var.

Usage

```
D4getenv(name, desc, quiet = FALSE)
```

Arguments

name	name of variable
desc	description of variable
quiet	suppress error messages, default=FALSE.

Value

Environment variable.

D4TAlink-common-args *Arguments used across the functions of the D4TAlink package.*

Description

Arguments used across the functions of the D4TAlink package.

Arguments

project	Project name.
package	Package name.
taskname	Task name.
author	Author name, system username by default.
sponsor	Sponsor name, default set by setTaskSponsor.
rootpath	Path of the task repository, default set by setTaskRoot.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.
dirType	Directory type, e.g. 'bin' or 'data' or 'doc'.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
pathgen	optional function returning a list of paths, currently pathsGLPG or pathsPMS.

Value

No return value, used for the documentation of the functions of the package.

D4TAlinkTask D4TAlinkTask *Documentation of the D4TAlinkTask class*

Description

The D4TAlinkTask object is created by the initTask function. This object is a list containing the task properties:

task: task name

package: package name

project: project name

sponsor: sponsor name

author: author name

copyright: copyright, by default 'Copyright (c) [sponsor] [year]'

'date': date of the task initialization, formatted as 'year-month-day'

'footer': footer for the task, e.g., 'Copyright (c) [sponsor] [year] - CONFIDENTIAL'

'version': string with task version, '0.0' at the initialization

dependencies: information on R versions and names of loaded/attached dependencies and corresponding versions

There are different functions dedicated for this D4TAlinkTask object:

taskID: Get ID

Value

Not relevant

Examples

```
## Not run:
# set D4TAlink's global parameters
setTaskAuthor("Doe Johns")
setTaskSponsor("mySponsor")

# Create data repository
setTaskRoot(file.path(tempdir(),"D4TAlink_example001"),dirCreate=TRUE)

# Create a task
task <- initTask(project="myProject",
                 package="myPackage",
                 taskname=sprintf("%s_myTask",format(Sys.time(),"%Y%m%d")))

# Output a plot to a PDF file
file <- pdfReport(task,c("plots",1),dim=c(100,100))
opa <- par()$ask
par(ask=FALSE)
hist(rnorm(100))
par(ask=opa)
dev.off()
# View the plot:
utils::browseURL(file)
```

```
# Output tables to an Excel file
d <- list(letters=data.frame(a=LETTERS,b=letters,c=1:length(letters)),
          other=data.frame(a=1:3,b=11:13))
file <- saveReportXls(d,task,"table")
utils::browseURL(file)

# Save an R object to a binary file
saveBinary(d,task,"data")
e <- readBinary(task,"data")
if(!all(names(e)%in%names(d))) stop("error [1]")

# Create and render R markdown file
initTaskRmd(task,overwrite=TRUE)
file <- renderTaskRmd(task) # requires having run 'tinytex::install_tinytex()'
utils::browseURL(file)

# Delete new data repository
unlink(getTaskRoot(),recursive=TRUE)

## End(Not run)
```

`datasourceDir` *Get path of data source directory.*

Description

Get path of data source directory.

Usage

```
datasourceDir(task, subdir = NULL, dirCreate = TRUE)
```

Arguments

`task`	Object of class `D4TAlinkTask`, as created by `initTask`.
`subdir`	(optional) Subdirectory.
`dirCreate`	Logical, if TRUE (by default) the directory is created.

Value

File path.

datasourceFn *Get path of data source file.*

Description

Get path of data source file.

Usage

```
datasourceFn(task, filename, subdir = ".", dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
filename	name of the input file.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

docDir *Get path of documentation directory.*

Description

Get path of documentation directory.

Usage

```
docDir(task, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

docFn *Get path of documentation file.*

Description

Get path of documentation file.

Usage

```
docFn(task, type, ext, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

DTx *Generic function.*

Description

Generic function.

Usage

```
DTx(sponsor = getTaskSponsor(), task = NULL)
```

Arguments

sponsor	Sponsor name, default set by setTaskSponsor.
task	Object of class D4TAlinkTask, as created by initTask.

Value

NULL.

formatTaskDocx *Replace default task fields in 'docx' file.*

Description

Replace default task fields in 'docx' file.

Usage

```
formatTaskDocx(task, ifn)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
ifn	input file name.

Value

the file name invisibly.

getTaskAuthor *Get the name of the task author.*

Description

Get the name of the task author.

Usage

```
getTaskAuthor(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

The current name of the tasks author.

Examples

```
getTaskAuthor(quiet=TRUE)
```

getTaskEnckey *Get the encryption key for binary data files.*

Description

Get the encryption key for binary data files.

Usage

```
getTaskEnckey(ask = FALSE)
```

Arguments

ask	query encryption key from user, default FALSE.

Value

The encryption key invisibly.

getTaskFilepath *Get the path of a file.*

Description

Get the path of a file.

Usage

```
getTaskFilepath(task, type, ext, dirtype, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
dirtype	task directory where file is stored, i.e., 'documentation', 'code', 'data', 'data source' or 'binary data'.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

Full path to file.

`getTaskPaths` *Get the paths of the task.*

Description

Get the paths of the task.

Usage

```
getTaskPaths(task)
```

Arguments

`task`	Object of class `D4TAlinkTask`, as created by `initTask`.

Value

List of task's paths.

`getTaskQmdTemplate` *Get the path to the Quarto task template.*

Description

Get the path to the Quarto task template.

Usage

```
getTaskQmdTemplate(quiet = FALSE)
```

Arguments

`quiet`	suppress error messages, default=FALSE.

Value

The path to the quarto task template.

getTaskRmdTemplate *Get the path to the Rmd task template.*

Description

Get the path to the Rmd task template.

Usage

```
getTaskRmdTemplate(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

The path to the Rmd task template.

getTaskRoot *Get the root of the task repository.*

Description

Get the root of the task repository.

Usage

```
getTaskRoot(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

Path to the current task root.

Examples

```
getTaskRoot(quiet=TRUE)
```

getTaskRscriptTemplate *Get the path to the R script task template.*

Description

Get the path to the R script task template.

Usage

```
getTaskRscriptTemplate(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

The path to the R script task template.

getTaskSponsor *Get the name of the task sponsor.*

Description

Get the name of the task sponsor.

Usage

```
getTaskSponsor(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

The current name of the tasks sponsor.

Examples

```
getTaskSponsor(quiet=TRUE)
```

getTaskStructure *Get repository directory structure.*

Description

Get repository directory structure.

Usage

```
getTaskStructure(quiet = FALSE)
```

Arguments

quiet	suppress error messages, default=FALSE.

Value

The directory structure function.

initTask *Initialize a task*

Description

During the initialization:

- The folder structure for the task is created in the data repository.
- The task properties are also saved in rds and json format.

Please note that it is recommended to load packages for your analysis before initializing the task.

Usage

```
initTask(
  project,
  package,
  taskname,
  sponsor = getTaskSponsor(),
  author = getTaskAuthor(),
  dirCreate = TRUE,
  templateCreate = FALSE,
  overwrite = FALSE
)
```

Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by setTaskSponsor.
author	Author name, system username by default.
dirCreate	logical, if TRUE (by default) the directory structure for the task is created in the repository.
templateCreate	create the prefilled Rmd template for the task, default value: FALSE.
overwrite	logical, if TRUE and the task already exists, overwrite its parameters.

Value

D4TAlinkTask object

---

initTaskQmd *Create task template in Quarto format.*

Description

Create task template in Quarto format.

Usage

```
initTaskQmd(task, encoding = "unknown", overwrite = FALSE, suffix = NA)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples and 'Details'.
overwrite	overwrite Qmd file if exists, default FALSE
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

the file name invisibly.

initTaskRmd *Create task template in Rmd format.*

Description

Create task template in Rmd format.

Usage

```
initTaskRmd(task, encoding = "unknown", overwrite = FALSE, suffix = NA)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples and 'Details'.
overwrite	overwrite Rmd file if exists, default FALSE
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

the file name invisibly.

initTaskRscript *Create task R script.*

Description

Create task R script.

Usage

```
initTaskRscript(task, overwrite = FALSE, encoding = "unknown", suffix = NA)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
overwrite	overwrite R file if exists, default FALSE
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples and 'Details'.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

the file name invisibly.

---

jpegReport *Graphics devices for* JPEG *format bitmap files.*

Description

Graphics devices for JPEG format bitmap files.

Usage

```
jpegReport(
  task,
  type,
  ext = "jpg",
  subdir = NULL,
  dirCreate = TRUE,
  dim = c(500, 500),
  width = NULL,
  height = NULL,
  ...
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Should be plotting be done using Windows GDI or cairographics?
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
dim	device height and width in px.
width	device height in px.
height	device height in px.
...	Arguments passed on to grDevices::jpeg
	filename the path of the output file, up to 511 characters. The page number is substituted if a C integer format is included in the character string, as in the default, and tilde-expansion is performed (see path.expand). (The result must be less than 600 characters long. See png for further details.)
	units The units in which height and width are given. Can be px (pixels, the default), in (inches), cm or mm.
	pointsize the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.

bg the initial background colour: can be overridden by setting par("bg").

quality the 'quality' of the JPEG image, as a percentage. Smaller values will give more compression but also more degradation of the image.

res The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default. If not specified, taken as 72 ppi to set the size of text and line widths.

family A length-one character vector specifying the default font family. The default means to use the font numbers on the Windows GDI versions and "sans" on the cairographics versions.

restoreConsole See the 'Details' section of windows. For type == "windows" only.

antialias Length-one character vector.
For allowed values and their effect on fonts with type = "windows" see windows: for that type if the argument is missing the default is taken from windows.options()$bitmap.aa.win.
For allowed values and their effect (on fonts and lines, but not fills) with type = "cairo" see svg.

symbolfamily For cairographics only: a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for plotmath output). The default value is "default", which means that R will choose a default "symbol" font based on the graphics device capabilities.

Value

the file name invisibly.

---

jpegReportFn *Get path of* jpeg *output file.*

Description

Get path of jpeg output file.

Usage

```
jpegReportFn(task, type, ext = "jpg", subdir = NULL)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

Value

File path.

`listTaskFiles` *List the files associated to a task.*

Description

List the files associated to a task.

Usage

```
listTaskFiles(task, full.names = FALSE, which = NULL)
```

Arguments

`task`	Object of class `D4TAlinkTask`, as created by `initTask`.
`full.names`	a logical value. If `TRUE`, the directory path is prepended to the file names to give a relative file path. If `FALSE`, the file names (rather than paths) are returned.
`which`	list of file types to list.

Value

array of file names.

`listTasks` *List details of all tasks stored in the root directory.*

Description

List details of all tasks stored in the root directory.

Usage

```
listTasks(
  project = NULL,
  package = NULL,
  sponsor = NULL,
  rootpath = getTaskRoot()
)
```

Arguments

project	Project name.
package	Package name.
sponsor	Sponsor name, default set by setTaskSponsor.
rootpath	Path of the task repository, default set by setTaskRoot.

Value

data.frame with the following information for tasks "sponsor", "project", "package", "task".

---

loadTask *Load a task.*

Description

Load a task.

Usage

```
loadTask(
  project,
  package,
  taskname,
  sponsor = getTaskSponsor(),
  author = getTaskAuthor(),
  quiet = FALSE
)
```

Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by setTaskSponsor.
author	Author name, system username by default.
quiet	issue warning if file does not exists.

Value

Object of class D4TAlinkTask or NULL if the task does not exists.

pathsDefault *Task paths generator.*

Description

The paths are: datasrc: [ROOT]/[sponsor]/[project]/[package]/raw/datasource data: [ROOT]/[sponsor]/[project]/[package]/ou
bin: [ROOT]/[sponsor]/[project]/[package]/output/[taskname]/bin code: [ROOT]/[sponsor]/[project]/[package]/progs
doc: [ROOT]/[sponsor]/[project]/[package]/docs log: [ROOT]/[sponsor]/[project]/[package]/output/log

Usage

```
pathsDefault(project, package, taskname, sponsor)
```

Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by setTaskSponsor.

Value

a list of file paths

pathsGLPG *Task paths generator.*

Description

The paths are: datasrc: [ROOT]/[sponsor]/[project]/[package]/raw/datasource data: [ROOT]/[sponsor]/[project]/[package]/ou
bin: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]/bin code: [ROOT]/[sponsor]/[project]/[package]/progs
doc: [ROOT]/[sponsor]/[project]/[package]/docs log: [ROOT]/[sponsor]/[project]/[package]/output/log

Usage

```
pathsGLPG(project, package, taskname, sponsor)
```

Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by setTaskSponsor.

Value

a list of file paths

pathsPMS *Task paths generator.*

Description

The paths are: datasrc: [ROOT]/[sponsor]/PMS_data/[project]/[package]/datasource data: [ROOT]/[sponsor]/PMS_data/[pro
bin: [ROOT]/[sponsor]/PMS_data/[project]/[package]/[taskname]/bin code: [ROOT]/[sponsor]/PMS_code/[project]/[packag
doc: [ROOT]/[sponsor]/PMS_documentation/[project]/[package]/[taskname] log: [ROOT]/[sponsor]/PMS_data/[project]/[pa

Usage

```
pathsPMS(project, package, taskname, sponsor)
```

Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by setTaskSponsor.

Value

a list of file paths

pdfReport *Graphics devices for* pdf *format bitmap files.*

Description

Graphics devices for pdf format bitmap files.

Usage

```
pdfReport(
  task,
  type,
  ext = "pdf",
  subdir = NULL,
  dirCreate = TRUE,
  title = NA,
  file = NA,
  dim = c(297, 210),
  height = NULL,
  width = NULL,
  landscape = NULL,
  ...
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
title	title string to embed as the '/Title' field in the document metadata. Defaults to "R Graphics Output"; use an empty string ("") to omit the field.
file	a character string giving the file path. See the section 'File specifications' for further details.
dim	device height and width in mm.
height	device height in mm.
width	device height in mm.
landscape	if defined, orientation of the document.
...	Arguments passed on to grDevices::pdf

width,height the width and height of the graphics region in inches. The default values are 7.

onefile logical: if true (the default) allow multiple figures in one file. If false, generate a file with name containing the page number for each page. Defaults to TRUE, and forced to true if file is a pipe.

family the initial font family to be used, normally as a character string. See the section 'Families'. Defaults to "Helvetica".

fonts a character vector specifying R graphics font family names for additional fonts which will be included in the PDF file. Defaults to NULL.

version a string describing the PDF version that will be required to view the output. This is a minimum, and will be increased (with a warning) if necessary. Defaults to "1.4", but see 'Details'.

paper the target paper size. The choices are "a4", "letter", "legal" (or "us") and "executive" (and these can be capitalized), or "a4r" and "USr" for rotated ('landscape'). The default is "special", which means that the width and height specify the paper size. A further choice is "default"; if this is selected, the paper size is taken from the option "papersize" if that is set and as "a4" if it is unset or empty. Defaults to "special".

encoding the name of an encoding file. Defaults to "default". The latter is interpreted

on Unix-alikes as '"ISOLatin1.enc"' unless the locale is recognized as corresponding to a language using ISO 8859-{2,5,7,13,15} or KOI8-{R,U}.

on Windows as '"CP1250.enc"' (Central European), "CP1251.enc" (Cyrillic), "CP1253.enc" (Greek) or "CP1257.enc" (Baltic) if one of those codepages is in use, otherwise '"WinAnsi.enc"' (codepage 1252).

The file is looked for in the 'enc' directory of package **grDevices** if the path does not contain a path separator. An extension ".enc" can be omitted.

bg the initial background color to be used. Defaults to "transparent".

fg the initial foreground color to be used. Defaults to "black".

pointsize the default point size to be used. Strictly speaking, in bp, that is 1/72 of an inch, but approximately in points. Defaults to 12.

pagecentre logical: should the device region be centred on the page? – is only relevant for paper != "special". Defaults to TRUE.

colormodel a character string describing the color model: currently allowed values are "srgb", "gray" (or "grey") and "cmyk". Defaults to "srgb". See section 'Color models'.

useDingbats logical. Should small circles be rendered *via* the Dingbats font? Defaults to FALSE. If TRUE, this can produce smaller and better output, but can cause font display problems in broken PDF viewers: although this font is one of the 14 guaranteed to be available in all PDF viewers, that guarantee is not always honoured.
For Unix-alikes (including macOS) see the 'Note' for a possible fix for some viewers.

useKerning logical. Should kerning corrections be included in setting text and calculating string widths? Defaults to TRUE.

fillOddEven logical controlling the polygon fill mode: see polygon for details. Defaults to FALSE.

compress logical. Should PDF streams be generated with Flate compression? Defaults to TRUE.

timestamp logical. If FALSE, omit the '/CreationDate' and '/ModDate' metadata fields. Defaults to TRUE.

producer logical. If FALSE, omit the '/Producer' metadata field. Defaults to TRUE.

author author string to embed as the '/Author' field in the document metadata. Defaults to "", omitting the field.

Value

the file name invisibly.

---

pdfReportFn *Get path of* pdf *output file.*

Description

Get path of pdf output file.

Usage

```
pdfReportFn(task, type, ext = "pdf", subdir = NULL)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

Value

File path.

---

pngReport *Graphics devices for* PNG *format bitmap files.*

Description

Graphics devices for PNG format bitmap files.

Usage

```
pngReport(
  task,
  type,
  ext = "png",
  subdir = NULL,
  dirCreate = TRUE,
  dim = c(500, 500),
  width = NULL,
  height = NULL,
  ...
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Should be plotting be done using Windows GDI or cairographics?
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
dim	device height and width in px.
width	device height in px.
height	device height in px.
...	Arguments passed on to grDevices::png

filename the path of the output file, up to 511 characters. The page number is substituted if a C integer format is included in the character string, as in the default, and tilde-expansion is performed (see path.expand). (The result must be less than 600 characters long. See png for further details.)

units The units in which height and width are given. Can be px (pixels, the default), in (inches), cm or mm.

pointsize the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.

bg the initial background colour: can be overridden by setting par("bg").

res The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default. If not specified, taken as 72 ppi to set the size of text and line widths.

family A length-one character vector specifying the default font family. The default means to use the font numbers on the Windows GDI versions and "sans" on the cairographics versions.

restoreConsole See the 'Details' section of windows. For type == "windows" only.

antialias Length-one character vector.
For allowed values and their effect on fonts with type = "windows" see windows: for that type if the argument is missing the default is taken from windows.options()$bitmap.aa.win.
For allowed values and their effect (on fonts and lines, but not fills) with type = "cairo" see svg.

symbolfamily For cairographics only: a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for plotmath output). The default value is "default", which means that R will choose a default "symbol" font based on the graphics device capabilities.

Value

the file name invisibly.

---

pngReportFn *Get path of* png *output file.*

Description

Get path of png output file.

Usage

```
pngReportFn(task, type, ext = "png", subdir = NULL)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

Value

File path.

---

progDir *Get path of scripts directory.*

Description

Get path of scripts directory.

Usage

```
progDir(task, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

---

qmdFn *Get path of R script file name.*

Description

Get path of R script file name.

Usage

```
qmdFn(task, suffix = NA)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

File path.

---

querySQLite *Load data frame by querying an SQLite database. This function executes a SQL query on an SQLite database file created with* saveSQLite.

Description

Load data frame by querying an SQLite database. This function executes a SQL query on an SQLite database file created with saveSQLite.

Usage

```
querySQLite(query, task, type, subdir = NULL, dirCreate = FALSE, conn = NULL)
```

Arguments

query	SQL query to execute.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
conn	SQLite connection object (optional).

Value

Data frame with query results.

readBinary *Restore R object from binary file.*

Description

Restore R object from binary file.

Usage

```
readBinary(
  task,
  type,
  subdir = NULL,
  dirCreate = FALSE,
  ask = FALSE,
  quiet = FALSE,
  password = NULL
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
ask	query encryption key from user, default FALSE.
quiet	issue warning if file does not exists.
password	encryption password, default NULL

Value

Object stored in binary file, or NULL if file does not exist.

readFeather *Restore R data.frame from Apache Arrow feather file.*

Description

Restore R data.frame from Apache Arrow feather file.

Usage

```
readFeather(task, type, subdir = NULL, dirCreate = FALSE, quiet = FALSE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
quiet	issue warning if file does not exists.

Value

Object stored in Apache Arrow feather file, or NULL if file does not exist.

---

readPickle *Restore R data.frame from a Python pickle file.*

Description

Restore R data.frame from a Python pickle file.

Usage

```
readPickle(task, type, subdir = NULL, dirCreate = FALSE, quiet = FALSE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
quiet	issue warning if file does not exists.

Value

Object stored in Apache Arrow feather file, or NULL if file does not exist.

readReportJSON *Read JSON data into R object.*

Description

Read JSON data into R object.

Usage

```
readReportJSON(task, type, ext = "json", subdir = NULL, dirCreate = FALSE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

the data read, or NULL if the file does not exist.

readReportTable *Read data into vector or list using function scan.*

Description

Read data into vector or list using function scan.

Usage

```
readReportTable(task, type, ext = "csv", subdir = NULL, dirCreate = FALSE, ...)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

... Arguments passed on to utils::read.table

file the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, getwd(). Tilde-expansion is performed where supported. This can be a compressed file (see file).

Alternatively, file can be a readable text-mode connection (which will be opened for reading if necessary, and if so closed (and hence destroyed) at the end of the function call). (If stdin() is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on stdin() will be cleared before return.)

file can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for url.)

header a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns.

sep the field separator character. Values on each line of the file are separated by this character. If sep = "" (the default for read.table) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.

quote the set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.

dec the character used in the file for decimal points.

numerals string indicating how to convert numbers whose conversion to double precision would lose accuracy, see type.convert. Can be abbreviated. (Applies also to complex-number inputs.)

row.names a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names.

If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if row.names is missing, the rows are numbered.

Using row.names = NULL forces row numbering. Missing or NULL row.names generate row names that are considered to be 'automatic' (and not preserved by as.matrix).

col.names a vector of optional names for the variables. The default is to use "V" followed by the column number.

as.is controls conversion of character variables (insofar as they are not converted to logical, numeric or complex) to factors, if not otherwise specified by colClasses. Its value is either a vector of logicals (values are recycled if necessary), or a vector of numeric or character indices which specify which columns should not be converted to factors.

Note: to suppress all conversions including those of numeric columns, set colClasses = "character".

Note that as.is is specified per column (not per variable) and so includes the column of row names (if any) and any columns to be skipped.

tryLogical a logical determining if columns consisting entirely of "F", "T", "FALSE", and "TRUE" should be converted to logical; passed to type.convert, true by default.

na.strings a character vector of strings which are to be interpreted as NA values. Blank fields are also considered to be missing values in logical, integer, numeric and complex fields. Note that the test happens *after* white space is stripped from the input (if enabled), so na.strings values may need their own white space stripped in advance.

colClasses character. A vector of classes to be assumed for the columns. If unnamed, recycled as necessary. If named, names are matched with unspecified values being taken to be NA.
Possible values are NA (the default, when type.convert is used), "NULL" (when the column is skipped), one of the atomic vector classes (logical, integer, numeric, complex, character, raw), or "factor", "Date" or "POSIXct". Otherwise there needs to be an as method (from package **methods**) for conversion from "character" to the specified formal class.
Note that colClasses is specified per column (not per variable) and so includes the column of row names (if any).

nrows integer: the maximum number of rows to read in. Negative and other invalid values are ignored.

skip integer: the number of lines of the data file to skip before beginning to read data.

check.names logical. If TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by make.names) so that they are, and also to ensure that there are no duplicates.

fill logical. If TRUE then in case the rows have unequal length, blank fields are implicitly added. See 'Details'.

strip.white logical. Used only when sep has been specified, and allows the stripping of leading and trailing white space from unquoted character fields (numeric fields are always stripped). See scan for further details (including the exact meaning of 'white space'), remembering that the columns may include the row names.

blank.lines.skip logical: if TRUE blank lines in the input are ignored.

comment.char character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.

allowEscapes logical. Should C-style escapes such as '\n' be processed or read verbatim (the default)? Note that if not within quotes these could be interpreted as a delimiter (but not as a comment character). For more details see scan.

flush logical: if TRUE, scan will flush to the end of the line after reading the last of the fields requested. This allows putting comments after the last field.

`stringsAsFactors` logical: should character vectors be converted to factors? Note that this is overridden by `as.is` and `colClasses`, both of which allow finer control.

`fileEncoding` character string: if non-empty declares the encoding used on a file when given as a character string (not on an existing connection) so the character data can be re-encoded. See the 'Encoding' section of the help for `file`, the 'R Data Import/Export' manual and 'Note'.

`encoding` encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8 (see `Encoding`): it is not used to re-encode the input, but allows R to handle encoded strings in their native encoding (if one of those two). See 'Value' and 'Note'.

`text` character string: if `file` is not supplied and this is, then data are read from the value of `text` via a text connection. Notice that a literal string can be used to include (small) data sets within R code.

`skipNul` logical: should NULs be skipped?

Value

the data read, or `NULL` if the file does not exist.

`readSQLite` *Load data frame from SQLite. This function reads a data frame from an SQLite database file created with* `saveSQLite`.

Description

Load data frame from SQLite. This function reads a data frame from an SQLite database file created with `saveSQLite`.

Usage

```
readSQLite(
  task,
  type,
  subdir = NULL,
  dirCreate = FALSE,
  tableName = "data",
  n = NULL,
  offset = 0
)
```

Arguments

`task`	Object of class `D4TAlinkTask`, as created by `initTask`.
`type`	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]

subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
tableName	name of the table in the SQLite database.
n	number of rows to read (if NULL, all rows are read).
offset	number of rows to skip before reading (default is 0).

Value

Data frame with requested table.

---

renderTaskQmd *Render the task from the Qmd file*

Description

The template of the task is rendered towards pdf or html in the documentation directory of the specified task. Note that on windows, Gnu zip may be required. The path to the executable must be added to the environment variables.

Usage

```
renderTaskQmd(task, output_format = "docx", debug = FALSE, suffix = NA, ...)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
output_format	Target output format (defaults to "html"). The option "all" will render all formats defined within the file or project.
debug	if TRUE execute in the global environment.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.
...	Arguments passed on to quarto::quarto_render input The input file or project directory to be rendered (defaults to rendering the project in the current working directory). output_file Base name for single-file output (e.g. PDF, ePub, MS Word). This sets the output-file Quarto metadata. If NULL, the output filename will be based on the input filename. execute Whether to execute embedded code chunks. execute_params A list of named parameters that override custom params specified within the YAML front-matter. execute_dir The working directory in which to execute embedded code chunks. execute_daemon Keep Jupyter kernel alive (defaults to 300 seconds). Note this option is only applicable for rendering Jupyter notebooks or Jupyter markdown.

`execute_daemon_restart` Restart keepalive Jupyter kernel before render. Note this option is only applicable for rendering Jupyter notebooks or Jupyter markdown.

`execute_debug` Show debug output for Jupyter kernel.

`use_freezer` Force use of frozen computations for an incremental file render.

`cache` Cache execution output (uses knitr cache and jupyter-cache respectively for Rmd and Jupyter input files).

`cache_refresh` Force refresh of execution cache.

`metadata` An optional named list used to override YAML metadata. It will be passed as a YAML file to `--metadata-file` CLI flag. This will be merged over `metadata-file` options if both are specified.

`metadata_file` A yaml file passed to `--metadata-file` CLI flags to override metadata. This will be merged with `metadata` if both are specified, with low precedence on `metadata` options.

`quiet` Suppress warning and other messages, from R and also Quarto CLI (i.e `--quiet` is passed as command line).
`quarto.quiet` R option or `R_QUARTO_QUIET` environment variable can be used to globally override a function call (This can be useful to debug tool that calls `quarto_*` functions directly).
On Github Actions, it will always be `quiet = FALSE`.

`profile` Quarto project profile(s) to use. Either a character vector of profile names or `NULL` to use the default profile.

`quarto_args` Character vector of other `quarto` CLI arguments to append to the Quarto command executed by this function. This is mainly intended for advanced usage and useful for CLI arguments which are not yet mirrored in a dedicated parameter of this R function. See `quarto render --help` for options.

`pandoc_args` Additional command line arguments to pass on to Pandoc.

`as_job` Render as an RStudio background job. Default is `"auto"`, which will render individual documents normally and projects as background jobs. Use the `quarto.render_as_job` R option to control the default globally.

Value

the file name invisibly.

`renderTaskRmd` *Render the task from the Rmd file*

Description

The template of the task is rendered towards pdf or html in the documentation directory of the specified task. Note that on windows, Gnu zip may be required. The path to the executable must be added to the environment variables.

Usage

```
renderTaskRmd(
  task,
  output_format = NULL,
  debug = FALSE,
  clean = TRUE,
  suffix = NA,
  ...
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
output_format	The R Markdown output format to convert to. The option "all" will render all formats defined within the file. The option can be the name of a format (e.g. "html_document") and that will render the document to that single format. One can also use a vector of format names to render to multiple formats. Alternatively, you can pass an output format object (e.g. html_document()). If using NULL then the output format is the first one defined in the YAML frontmatter in the input file (this defaults to HTML if no format is specified there). If you pass an output format object to output_format, the options specified in the YAML header or _output.yml will be ignored and you must explicitly set all the options you want when you construct the object. If you pass a string, the output format will use the output parameters in the YAML header or _output.yml.
debug	if TRUE execute in the global environment.
clean	Using TRUE will clean intermediate files that are created during rendering.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.
...	Arguments passed on to rmarkdown::render input The input file to be rendered. This can be an R script (.R), an R Markdown document (.Rmd), or a plain markdown document. output_file The name of the output file. If using NULL then the output filename will be based on filename for the input file. If a filename is provided, a path to the output file can also be provided. Note that the output_dir option allows for specifying the output file path as well, however, if also specifying the path, the directory must exist. If output_file is specified but does not have a file extension, an extension will be automatically added according to the output format. To avoid the automatic file extension, put the output_file value in I(), e.g., I('my-output'). output_dir The output directory for the rendered output_file. This allows for a choice of an alternate directory to which the output file should be written (the default output directory of that of the input file). If a path is provided with a filename in output_file the directory specified here will take precedence. Please note that any directory path provided will create any necessary directories if they do not exist.

`output_options` List of output options that can override the options specified in metadata (e.g. could be used to force `self_contained` or `mathjax = "local"`). Note that this is only valid when the output format is read from metadata (i.e. not a custom format object passed to `output_format`).

`output_yaml` Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the `output_yaml` top-level parameter in the YAML front matter, _output.yml or _output.yaml, and then uses the first existing one.

`intermediates_dir` Intermediate files directory. If a path is specified then intermediate files will be written to that path. If `NULL`, intermediate files are written to the same directory as the input file.

`knit_root_dir` The working directory in which to knit the document; uses knitr's `root.dir` knit option. If `NULL` then the behavior will follow the knitr default, which is to use the parent directory of the document.

`runtime` The runtime target for rendering. The `static` option produces output intended for static files; `shiny` produces output suitable for use in a Shiny document (see `run`). The default, `auto`, allows the `runtime` target specified in the YAML metadata to take precedence, and renders for a `static` runtime target otherwise.

`params` A list of named parameters that override custom params specified within the YAML front-matter (e.g. specifying a dataset to read or a date range to confine output to). Pass `"ask"` to start an application that helps guide parameter configuration.

`knit_meta` (This option is reserved for expert use.) Metadata generated by **knitr**.

`envir` The environment in which the code chunks are to be evaluated during knitting (can use `new.env()` to guarantee an empty new environment).

`run_pandoc` An option for whether to run pandoc to convert Markdown output.

`quiet` An option to suppress printing during rendering from knitr, pandoc command line and others. To only suppress printing of the last "Output created: " message, you can set `rmarkdown.render.message` to `FALSE`

`encoding` Ignored. The encoding is always assumed to be UTF-8.

Value

the file name invisibly.

`reportDir` *Get path of report directory.*

Description

Get path of report directory.

Usage

```
reportDir(task, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

---

reportFn *Get path of output file.*

Description

Get path of output file.

Usage

```
reportFn(task, type, ext, subdir = NULL, dirCreate = TRUE)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

File path.

reportXlsFn *Get path of* xlsx *output file.*

Description

Get path of xlsx output file.

Usage

```
reportXlsFn(task, type, ext = "xlsx", subdir = NULL)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

Value

File path.

restoreTask *Restore an archive containing the files of a given task from a file created with archiveTask.*

Description

Restore an archive containing the files of a given task from a file created with archiveTask.

Usage

```
restoreTask(file, overwrite = FALSE, list = FALSE, code = TRUE, ...)
```

Arguments

file	full name of the input zip file
overwrite	If TRUE, overwrite existing files (the equivalent of unzip -o), otherwise ignore such files (the equivalent of unzip -n).
list	If TRUE, list the files and extract none. The equivalent of unzip -l.
code	restore code, default TRUE
...	Arguments passed on to utils::unzip

`zipfile` The pathname of the zip file: tilde expansion (see `path.expand`) will be performed.

`files` A character vector of recorded filepaths to be extracted: the default is to extract all files.

`junkpaths` If `TRUE`, use only the basename of the stored filepath when extracting. The equivalent of `unzip -j`.

`exdir` The directory to extract files to (the equivalent of `unzip -d`). It will be created if necessary.

`unzip` The method to be used. An alternative is to use `getOption("unzip")`, which on a Unix-alike may be set to the path to a `unzip` program.

`setTimes` logical. For the internal method only, should the file times be set based on the times in the zip file? (NB: this applies to included files, not to directories.)

Value

if list FALSE, the task imported incisibly, otherwise the list of files in the archive.

`rmdFn` *Get path of R script file name.*

Description

Get path of R script file name.

Usage

```
rmdFn(task, suffix = NA)
```

Arguments

`task`	Object of class `D4TAlinkTask`, as created by `initTask`.
`suffix`	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

File path.

rscriptFn *Get path of R script file name.*

Description

Get path of R script file name.

Usage

```
rscriptFn(task, suffix = NA)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

Value

File path.

saveBinary *Save R object in binary file.*

Description

Save R object in binary file.

Usage

```
saveBinary(
  object,
  task,
  type,
  subdir = NULL,
  dirCreate = TRUE,
  encrypt = FALSE,
  ask = FALSE
)
```

Arguments

object	R object to serialize.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
encrypt	encrypt the output, default: FALSE. If character string, then use the string as password.
ask	query encryption key from user, default FALSE.

Value

the file name invisibly.

---

saveBinaryE *Save R object in encrypted binary file.*

Description

Save R object in encrypted binary file.

Usage

```
saveBinaryE(object, task, type, subdir = NULL, dirCreate = TRUE, ask = FALSE)
```

Arguments

object	R object to serialize.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
ask	query encryption key from user, default FALSE.

Value

the file name invisibly.

saveFeather *Save R object in Apache Arrow feather file.*

Description

Save R object in Apache Arrow feather file.

Usage

```
saveFeather(object, task, type, subdir = NULL, dirCreate = TRUE)
```

Arguments

object	data.frame to serialize.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

the file name invisibly.

savePickle *Save R object in Python pickle file.*

Description

Save R object in Python pickle file.

Usage

```
savePickle(object, task, type, subdir = NULL, dirCreate = TRUE)
```

Arguments

object	data.frame to serialize.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

the file name invisibly.

---

saveReportJSON *Output R object in JSON format.*

Description

Output R object in JSON format.

Usage

```
saveReportJSON(x, task, type, ext = "json", subdir = NULL, dirCreate = TRUE)
```

Arguments

x	R object to output.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

Value

the file name invisibly.

---

saveReportTable *Output R object using function write.csv.*

Description

Output R object using function write.csv.

Usage

```
saveReportTable(
  x,
  task,
  type,
  ext = "csv",
  subdir = NULL,
  dirCreate = TRUE,
  gzip = FALSE,
  ...
)
```

Arguments

x	R object to output.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
gzip	unused.
...	Arguments passed on to utils::write.table

file either a character string naming a file or a connection open for writing. "" indicates output to the console.

append logical. Only relevant if file is a character string. If TRUE, the output is appended to the file. If FALSE, any existing file of the name is destroyed.

quote a logical value (TRUE or FALSE) or a numeric vector. If TRUE, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If FALSE, nothing is quoted.

sep the field separator string. Values within each row of x are separated by this string.

eol the character(s) to print at the end of each line (row). For example, eol = "\r\n" will produce Windows' line endings on a Unix-alike OS, and eol = "\r" will produce files as expected by Excel:mac 2004.

na the string to use for missing values in the data.

dec the string to use for decimal points in numeric or complex columns: must be a single character.

row.names either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.

col.names either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written. See the section on 'CSV files' for the meaning of col.names = NA.

qmethod a character string specifying how to deal with embedded double quote characters when quoting strings. Must be one of "escape" (default for write.table), in which case the quote character is escaped in C style by a backslash, or "double" (default for write.csv and write.csv2), in which case it is doubled. You can specify just the initial letter.

fileEncoding character string: if non-empty declares the encoding to be used on a file (not a connection) so the character data can be re-encoded as they are written. See file.

Value

the file name invisibly.

saveReportXls *Save R object in binary file.*

Description

Save R object in binary file.

Usage

```
saveReportXls(
  x,
  task,
  type,
  ext = "xlsx",
  subdir = NULL,
  dirCreate = TRUE,
  AdjWidth = TRUE,
  FreezeRow = 1,
  FreezeCol = 3,
  metadata = "metadata",
  metadata.append = NULL,
  ...
)
```

Arguments

x	object to save. It can be either a data frame, an object of type AnnotatedDataFrame, or a list thereof.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
AdjWidth	If TRUE, will adjust the worksheet column widths based upon the longest entry in each column. This is approximate.
FreezeRow	Rows including this row and above this row will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
FreezeCol	Columns including this column and to the left of this column will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
metadata	prefix for names of worksheets holding metadata.

`metadata.append`
: array of metadata field names to be appended in header of tables.

`...`
: Arguments passed on to `WriteXLS::WriteXLS`

`ExcelFileName` The name of the Excel file to be created. If the file extension is *.XLS*, an Excel 2003 file will be created. If the file extension is *.XLSX*, an Excel 2007 file will be created. Must be a valid Excel filename. May include an existing path. `normalizePath` is used to support tilde expansion, etc.

`SheetNames` A character vector containing the names of each worksheet to be created. If `NULL` (the default), the names of the dataframes will be used instead. Worksheet names may be up to 31 characters in length and must be unique. If specified, `length(SheetNames)` must be the same as `length(x)`. NOTE: The order of the names here must match the order of the data frames as listed in `x`.

`perl` Name of the perl executable to be called.

`verbose` Output step-by-step status messages during the creation of the Excel file. Default is FALSE.

`Encoding` Define the character encoding to be used for the exported data frames. Defaults to `UTF-8`.

`AllText` If `TRUE`, all cell contents of the Excel file will be written as text. Default is FALSE. See Details.

`row.names` If `TRUE`, the row names of the data frames are included in the Excel file worksheets.

`col.names` If `TRUE`, the column names of the data frames are included in the Excel file worksheets.

`AutoFilter` If `TRUE`, will add autofiltering to each column in each worksheet. Note that not all spreadsheet applications support this feature.

`BoldHeaderRow` If `TRUE`, will apply a bold font to the header row for each worksheet.

`ReadOnly` If `TRUE`, each worksheet will be set to Read Only (Protected mode) to prevent inadvertent changes to the contents when the file is opened in Excel or a compatible application. See Details.

`na` The string to use for missing values in the data. Defaults to ""

`envir` The environment in which to look for the data frames named in `x`. This defaults to the environment in which `WriteXLS` was called.

Value

the file name invisibly.

`saveSQLite` *Save R data frame to SQLite file. This function saves a data frame to an SQLite database file. It can create indices on specified columns for faster querying. The database file is stored in the task's binary directory.*

Description

Save R data frame to SQLite file. This function saves a data frame to an SQLite database file. It can create indices on specified columns for faster querying. The database file is stored in the task's binary directory.

Usage

```
saveSQLite(
  object,
  task,
  type,
  subdir = NULL,
  dirCreate = TRUE,
  index = NULL,
  tableName = "data",
  overwrite = FALSE,
  append = FALSE,
  row.names = FALSE
)
```

Arguments

object	data.frame to serialize.
task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.
index	character vector of column names to create indices on.
tableName	name of the table in the SQLite database.
overwrite	if TRUE, overwrite existing table.
append	if TRUE, append to existing table.
row.names	if TRUE, save row names as a column.

Value

The file name invisibly.

Examples

```
## Not run:
task <- loadTask("myproject","mypackage","mytask")
# create a sample data frame
df <- data.frame(a = 1:5, b = letters[1:5])
# save it to SQLite
fn <- saveSQLite(df, task, "testdb",tableName="table1",
                 index = c("a"))
```

```
# read it back
df_loaded <- readSQLite(task,"testdb",tableName="table1")
# print the loaded data frame
print(df_loaded)
# add second data frame
df2 <- data.frame(a = 6:10, b = letters[6:10])
saveSQLite(df2, task,"testdb",tableName="table2")
# query the data
query_result <- querySQLite("SELECT * FROM table2 WHERE a > 8",
                            task,"testdb")
print(query_result)
#' # append to the first table
saveSQLite(df2, task,"testdb",tableName="table1",append=TRUE)
# read the updated table
df_loaded <- readSQLite(task,"testdb",tableName="table1")
# print the updated data frame
print(df_loaded)
# cleanup
unlink(fn)

## End(Not run)
```

---

scanReport *Read data into vector or list using function* scan.

Description

Read data into vector or list using function scan.

Usage

```
scanReport(
  task,
  type,
  ext = "txt",
  subdir = NULL,
  dirCreate = TRUE,
  what = "",
  ...
)
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

dirCreate Logical, if TRUE (by default) the directory is created.

what the type of what gives the type of data to be read. (Here 'type' is used in the sense of typeof.) The supported types are logical, integer, numeric, complex, character, raw and list. If what is a list, it is assumed that the lines of the data file are records each containing length(what) items ('fields') and the list components should have elements which are one of the first six (atomic) types listed or NULL, see section 'Details' below.

... Arguments passed on to base::scan

file the name of a file to read data values from. If the specified file is "", then input is taken from the keyboard (or whatever stdin() reads if input is redirected or R is embedded). (In this case input can be terminated by a blank line or an EOF signal, 'Ctrl-D' on Unix and 'Ctrl-Z' on Windows.) Otherwise, the file name is interpreted *relative* to the current working directory (given by getwd()), unless it specifies an *absolute* path. Tilde-expansion is performed where supported. When running R from a script, file = "stdin" can be used to refer to the process's stdin file stream.
This can be a compressed file (see file).
Alternatively, file can be a connection, which will be opened if necessary, and if so closed at the end of the function call. Whatever mode the connection is opened in, any of LF, CRLF or CR will be accepted as the EOL marker for a line and so will match sep = "\n".
file can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for url.)
To read a data file not in the current encoding (for example a Latin-1 file in a UTF-8 locale or conversely) use a file connection setting its encoding argument (or scan's fileEncoding argument).

nmax the maximum number of data values to be read, or if what is a list, the maximum number of records to be read. If omitted or not positive or an invalid value for an integer (and nlines is not set to a positive value), scan will read to the end of file.

n integer: the maximum number of data values to be read, defaulting to no limit. Invalid values will be ignored.

sep by default, scan expects to read 'white-space' delimited input fields. Alternatively, sep can be used to specify a character which delimits fields. A field is always delimited by an end-of-line marker unless it is quoted.
If specified this should be the empty character string (the default) or NULL or a character string containing just one single-byte character.

quote the set of quoting characters as a single character string or NULL. In a multibyte locale the quoting characters must be ASCII (single-byte).

dec decimal point character. This should be a character string containing just one single-byte character. (NULL and a zero-length character vector are also accepted, and taken as the default.)

skip the number of lines of the input file to skip before beginning to read data values.

nlines if positive, the maximum number of lines of data to be read.

na.strings character vector. Elements of this vector are to be interpreted as missing (NA) values. Blank fields are also considered to be missing values

in logical, integer, numeric and complex fields. Note that the test happens *after* white space is stripped from the input (if enabled), so `na.strings` values may need their own white space stripped in advance.

`flush` logical: if `TRUE`, `scan` will flush to the end of the line after reading the last of the fields requested. This allows putting comments after the last field, but precludes putting more than one record on a line.

`fill` logical: if `TRUE`, `scan` will implicitly add empty fields to any lines with fewer fields than implied by `what`.

`strip.white` vector of logical value(s) corresponding to items in the `what` argument. It is used only when `sep` has been specified, and allows the stripping of leading and trailing 'white space' from `character` fields (other fields are always stripped). Note: white space inside quoted strings is not stripped.

If `strip.white` is of length 1, it applies to all fields; otherwise, if `strip.white[i]` is `TRUE` *and* the `i`-th field is of mode character (because `what[i]` is) then the leading and trailing unquoted white space from field `i` is stripped.

`quiet` logical: if `FALSE` (default), scan() will print a line, saying how many items have been read.

`blank.lines.skip` logical: if `TRUE` blank lines in the input are ignored, except when counting `skip` and `nlines`.

`multi.line` logical. Only used if `what` is a list. If `FALSE`, all of a record must appear on one line (but more than one record can appear on a single line). Note that using `fill = TRUE` implies that a record will be terminated at the end of a line.

`comment.char` character: a character vector of length one containing a single character or an empty string. Use `""` to turn off the interpretation of comments altogether (the default).

`allowEscapes` logical. Should C-style escapes such as '`\n`' be processed (the default) or read verbatim? Note that if not within quotes these could be interpreted as a delimiter (but not as a comment character).

The escapes which are interpreted are the control characters '`\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`' and octal and hexadecimal representations like '`\040`' and '`\0x2A`'. Any other escaped character is treated as itself, including backslash. Note that Unicode escapes (starting '`\u`' or '`\U`': see Quotes) are never processed.

`fileEncoding` character string: if non-empty declares the encoding used on a file (not a connection nor the keyboard) so the character data can be re-encoded. See the 'Encoding' section of the help for `file`, and the 'R Data Import/Export Manual'.

`encoding` encoding to be assumed for input strings. If the value is `"latin1"` or `"UTF-8"` it is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input (see `fileEncoding`). See also 'Details'.

`text` character string: if `file` is not supplied and this is, then data are read from the value of `text` via a text connection.

`skipNul` logical: should NULs be skipped when reading character fields?

Value

the data read, or NULL if the file does not exist.

---

setTaskAuthor *Set the name of the tasks author.*

Description

Set the name of the tasks author.

Usage

```
setTaskAuthor(author)
```

Arguments

author	Author name, system username by default.

Value

The current name of the tasks author.

Examples

```
setTaskAuthor("Doe Johns")
```

---

setTaskEnckey *Set the encryption key for binary data files.*

Description

Set the encryption key for binary data files.

Usage

```
setTaskEnckey(key)
```

Arguments

key	encryption key, if NULL then query key from user.

Value

NULL invisibly.

setTaskQmdTemplate *Set the path to the Quarto task template.*

Description

Set the path to the Quarto task template.

Usage

```
setTaskQmdTemplate(file, encoding = "unknown")
```

Arguments

file	path to the Quarto task template.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples and 'Details'.

Value

The path to the Quato task template invisibly.

setTaskRmdTemplate *Set the path to the Rmd task template.*

Description

Set the path to the Rmd task template.

Usage

```
setTaskRmdTemplate(file, encoding = "unknown")
```

Arguments

file	path to the Rmd task template.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples and 'Details'.

Value

The path to the Rmd task template invisibly.

`setTaskRoot` *Set the root of the task repository.*

Description

Set the root of the task repository.

Usage

```
setTaskRoot(rootpath, dirCreate = FALSE)
```

Arguments

`rootpath`	Path of the task repository, default set by `setTaskRoot`.
`dirCreate`	Logical, if TRUE (by default) the directory is created.

Value

Path to the current task root.

`setTaskRscriptTemplate` *Set the path to the R script task template.*

Description

Set the path to the R script task template.

Usage

```
setTaskRscriptTemplate(file)
```

Arguments

`file`	path to the Rmd task template.

Value

The path to the Rmd task template invisibly.

setTaskSponsor *Set the name of the tasks sponsor.*

Description

Set the name of the tasks sponsor.

Usage

```
setTaskSponsor(sponsor)
```

Arguments

sponsor	Sponsor name, default set by setTaskSponsor.

Value

The current name of the tasks sponsor.

Examples

```
setTaskSponsor("SQU4RE")
```

setTaskStructure *Set task repository directory structure.*

Description

Set task repository directory structure.

Usage

```
setTaskStructure(pathgen)
```

Arguments

pathgen	optional function returning a list of paths, currently pathsGLPG or pathsPMS.

Value

The task directory structure function invisibly.

Examples

```
fun <- function(project,package,taskname,sponsor) {
  basePath <- file.path("%ROOT%",sponsor,project,package)
  list(
    root = "%ROOT%",
    datasrc = file.path(basePath, "raw", "data_source"),
    data = file.path(basePath, "output","adhoc",taskname),
    bin  = file.path(basePath, "output","adhoc",taskname,"bin"),
    code = file.path(basePath, "progs"),
    doc  = file.path(basePath, "docs"),
    log  = file.path(basePath, "output","log")
  )
}
setTaskStructure(fun)
```

---

taskID *Get task identifier string.*

Description

Get task identifier string.

Usage

```
taskID(task, sep = "/")
```

Arguments

task	Object of class D4TAlinkTask, as created by initTask.
sep	the field separator character, default: "/".

Value

String with task ID as:[sponsor][sep][project][sep][package][sep][task]

Index