Package ‘simplace’

October 9, 2024

Type Package

Title Interface to Use the Modelling Framework 'SIMPLACE'

Version 5.1.2

Date 2024-10-09

Encoding UTF-8

Maintainer Gunther Krauss <guntherkrauss@uni-bonn.de>

Description Interface to interact with the modelling framework 'SIMPLACE' and to parse the results of simulations.

License GPL-2

URL https://github.com/gk-crop/simplace_rpkg/

SystemRequirements Java (>= 17.0)

Imports rJava (>= 0.9-13)

LazyLoad yes

VignetteBuilder knitr

Suggests knitr, rmarkdown

RoxygenNote 7.3.2

NeedsCompilation no

Author Gunther Krauss [aut, cre]

Repository CRAN

Date/Publication 2024-10-09 08:10:02 UTC

Contents

`closeProject` *Close Project*

Description

Call to the shutDown method of the simulation.

Usage

```
closeProject(simplace)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`

Value

No return value, called for the side effect of closing the simulation project

See Also

`openProject`

`createSimulation` *Creates a simulation and substitute parameters*

Description

Creates a simulation from the opened project and substitutes the values of the parameters given in the parameter list. Simulation won't be queued by default.

Usage

```
createSimulation(simplace, parameterList = NULL, queue = FALSE)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`parameterList`	a list with the parameter name as key and parametervalue as value
`queue`	boolean - simulation is added to queue if true, else start new queue

Value

id of the created simulation

See Also

`runSimulations`, `resetSimulationQueue`

`findFirstSimplaceInstallation` *Search for simplace installation and returns first match*

Description

Checks directories if they contain simplace_core, simplace_modules and optionally simplace_run (or a data directory given by the user) and returns the first match. There is no check whether the installation is really working.

Usage

```
findFirstSimplaceInstallation(
  directories = c(),
  tryStandardDirs = TRUE,
  simulationsDir = "simplace_run",
  ignoreSimulationsDir = FALSE
)
```

Arguments

`directories`	a list of additional directories where to look -
`tryStandardDirs`	whether to check for typical installation directories (default)
`simulationsDir`	directory that contains user simulations (e.g. simplace_run)
`ignoreSimulationsDir`	don't check for the simulation dir

Details

Beside the checks for some standard directories (like home directory, current working dir and drives c: to g:) and their subdirectories (workspace, simplace, java/simplace) the user can give a vector of additional directories. Directories given by the user are checked first.

Value

matching directory/ies as character vector

`findSimplaceInstallations` *Search for simplace installations and returns results as vector*

Description

Checks directories if they contain simplace_core, simplace_modules and optionally simplace_run (or a data directory given by the user) and returns the matches. There is no check whether the installation is really working.

Usage

```
findSimplaceInstallations(
  directories = c(),
  tryStandardDirs = TRUE,
  firstMatchOnly = FALSE,
  simulationsDir = "simplace_run",
  ignoreSimulationsDir = FALSE,
  verbose = TRUE
)
```

Arguments

`directories`	a list of additional directories where to look -
`tryStandardDirs`	whether to check for typical installation directories (default)
`firstMatchOnly`	returns only the first installation found
`simulationsDir`	directory that contains user simulations (e.g. simplace_run)

`ignoreSimulationsDir`	don't check for the simulation dir
`verbose`	prints messages if no or more than one installation found

Details

Beside the checks for some standard directories (like home directory, current working dir and drives c: to g:) and their subdirectories (workspace, simplace, java/simplace) the user can give a vector of additional directories.

Value

matching directory/ies as character vector

`getDatatypesOfResult` *Get the datatypes of the result variables*

Description

Get the datatypes of each variable (i.e. data column). The output is a named character vector, where each element is named by the variables name.

Usage

```
getDatatypesOfResult(result)
```

Arguments

`result`	handle to the data container returned by `getResult`

Value

named character vector with the datatypes

`getResult` *Fetch output from a simulation*

Description

The output is a JavaObject containing the variable names, data types, units and the values. Output can be converted with `resultToList` or `resultToDataframe` to R objects. Only MEMORY outputs are accessible. For CSV or database outputs you have to read the data by generic methods.

Usage

```
getResult(simplace, outputId, simulationId = nullString)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`outputId`	id of the output. Only MEMORY outputs are accessible.
`simulationId`	id of the simulation

Value

handle to the data container which has to be processed afterwards

`getSimplaceDirectories` *Get the directories (work-, output-, projects- and data-dir)*

Description

Get the directories (work-, output-, projects- and data-dir)

Usage

```
getSimplaceDirectories(simplace)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`

Value

character vector with the directories

See Also

`setSimplaceDirectories`

`getSimulationIDs` *Lists IDs of the performed simulations*

Description

Returns a vector with the IDs of the simulations. IDs are required to get the output of the simulations.

Usage

```
getSimulationIDs(simplace)
```

Arguments

`simplace`	handle to the SimplaceWrapper object

Value

character vector with the IDs

`getUnitsOfResult` *Get the units of the result variables*

Description

Get the units of each variable (i.e. data column) in a human readable format. The output is a named character vector, where each element is named by the variables name.

Usage

```
getUnitsOfResult(result)
```

Arguments

`result`	handle to the data container returned by `getResult`

Value

named character vector with the units

`initSimplace` *Initialisation of Framework*

Description

Initializes the JVM and creates the SimplaceWrapper object which is used to interact with Simplace.

Usage

```
initSimplace(
  InstallationDir = findFirstSimplaceInstallation(),
  WorkDir = paste0(InstallationDir, "simplace_run/simulation/"),
  OutputDir = paste0(InstallationDir, "simplace_run/output/"),
  ProjectsDir = nullString,
  DataDir = nullString,
  additionalClasspaths = c(),
  javaparameters = getOption("java.parameters"),
  force.init = TRUE
)
```

Arguments

`InstallationDir`	directory where simplace_core, simplace_modules and simplace_run are located
`WorkDir`	working directory where solutions, projects and data resides (_WORKDIR_)
`OutputDir`	directory for output (_OUTPUTDIR_)
`ProjectsDir`	optional directory for project data (_PROJECTSDIR_)
`DataDir`	optional directory for data (_DATADIR_)
`additionalClasspaths`	vector with class paths relative to InstallationDir that are to be added
`javaparameters`	parameters that are passed to the java virtual machine
`force.init`	(re)initialize a running JVM, see `.jinit`

Value

handle to the SimplaceWrapper object

`initSimplaceDefault` *Initialises Simplace with work- and outputdir for different settings*

Description

Initialises Simplace with work- and outputdir for different settings

Usage

```
initSimplaceDefault(setting = "run")
```

Arguments

`setting`	one of "run", "modules"," lapclient" or "wininstall"

Value

handle to the SimplaceWrapper object

See Also

`initSimplace`

`openProject` *Opens a Simplace project*

Description

Initializes a project. Solution is mandatory, project is optional. Solution and project files can be specified by giving absolute paths or paths relative to the simplace directory. Instead of using solution and project files, one can use the content of the solution / project directly as a string or a "xml_document" class.

Usage

```
openProject(simplace, solution, project = nullString, parameterList = NULL)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`solution`	solution file with absolute path or path relative to workdir
`project`	project file with absolute path or path relative to workdir, can be omitted to run solution only
`parameterList`	a list with the parameter name as key and parametervalue as value

Value

invisibly a list with java FWSimsession object as well as the solution, project and parameterList

See Also

`closeProject`

`resetSimulationQueue` *Clears the list of simulations*

Description

Simulation list is cleared

Usage

```
resetSimulationQueue(simplace)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`

Value

No return value, called for the side effect of clearing the simulation list

See Also

`createSimulation`, `runSimulations`

---

resultToDataframe *Convert result to dataframe*

Description

All scalar output columns are transformed to appropriate R objects and then glued together in a dataframe. Array outputs columns are ignored.

Usage

```
resultToDataframe(result, expand = FALSE, from = NULL, to = NULL)
```

Arguments

`result`	handle to the data container returned by `getResult`
`expand`	if true columns with arrays are partially expanded
`from`	start of the result range, if to/from are not set, full result is returned
`to`	end of the result range, if to/from are not set, full result is returned

Value

data.frame with scalar output columns

See Also

`resultToList` returns the output columns as list

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
parameter <- list(vTempLimit = 32)
simid <- createSimulation(simplace,parameter)
runSimulations(simplace)
result <- getResult(simplace,"DIAGRAM_OUT", simid);
closeProject(simplace)
resultframe <- resultToDataframe(result)
resultframe[3,]
## End(Not run)
```

resultToList *Convert result to list*

Description

Converts all scalar output columns to appropriate R lists. Columns containing arrays are left unchanged, unless expand is TRUE.

Usage

```
resultToList(result, expand = FALSE, from = NULL, to = NULL)
```

Arguments

`result`	handle to the data container returned by `getResult`
`expand`	if true columns with arrays are partially expanded
`from`	start of the result range, if to/from are not set, full result is returned
`to`	end of the result range, if to/from are not set, full result is returned

Value

list with output columns

See Also

`resultToDataframe` returns the scalar output columns as `data.frame`

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
parameter <- list(vTempLimit = 32)
simid <- createSimulation(simplace,parameter)
runSimulations(simplace)
closeProject(simplace)
result <- getResult(simplace,"DIAGRAM_OUT", simid);
resultlist <- resultToList(result)
resultlist$CURRENT.DATE
## End(Not run)
```

runProject *Runs the opened project*

Description

Runs the simulation(s) as defined in the solution and project files. There is no accessible MEMORY output, but one can load the CSV or database output.

Usage

```
runProject(simplace)
```

Arguments

simplace	handle to the SimplaceWrapper object returned by initSimplace

Value

No return value, called for the side effect of running opened project

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution, Project)
runProject(simplace)
closeProject(simplace)
## End(Not run)
```

runSimulations *Run the created simulations*

Description

Run the created simulations from the queue. If the queue is empty, the last created simulation will be run.

Usage

```
runSimulations(simplace, selectsimulation = FALSE)
```

Arguments

simplace	handle to the SimplaceWrapper object returned by initSimplace
selectsimulation	if true keeps a selected simulation

Value

No return value, called for the side effect of running the simulation

See Also

createSimulation, resetSimulationQueue

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
parameters <- list()
parameters$vBaseLUE <- 3.0
s1 <- createSimulation(simplace, parameters,queue=TRUE)
parameters$vBaseLUE <- 3.2
s2 <- createSimulation(simplace, parameters,queue=TRUE)
runSimulations(simplace)
parameters$vBaseLUE <- 2.8
s3 <- createSimulation(simplace, parameters,queue=TRUE)
runSimulations(simplace)

closeProject(simplace)
## End(Not run)
```

`setAllSimulationValues`
Changes values of the all simulations in queue

Description

Sets values of arbitrary SimVariables in a simplace simulation. Useful if you want to couple simplace with another simulation and interchange values daily.

Usage

```
setAllSimulationValues(simplace, parameterLists = NULL)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`parameterLists`	a list of parameter lists for each simulation

Value

No return value, called for the side effect of changing parameters in all simulations

Examples

```
## Not run:
for(i in 1:365)
{
  params <- list()
  params[[1]] <- list(vBaseLUE=3.0 + i/2000)
  params[[2]] <- list(vBaseLUE=3.0 - i/2000)
  setAllSimulationValues(simplace,params)
  stepAllSimulations(simplace)
}

## End(Not run)
```

---

setCheckLevel *Sets the check level of the framework*

Description

Sets the check level. OFF does no check at all, STRICT the most severe. You have to call `initSimplace` first.

Usage

```
setCheckLevel(simplace, level)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`level`	is a string with possible values: "CUSTOM,"STRICT","INTENSE","LAZY","OFF","ONLY"

Value

No return value, called for the side effect of setting the check level

Examples

```
## Not run:
setCheckLevel(simplace, "STRICT")
## End(Not run)
```

setLogLevel — *Sets the log level of the framework*

Description

Sets the level of logger output - FATAL is least verbose, TRACE most verbose. You have to call initSimplace first.

Usage

```
setLogLevel(level)
```

Arguments

level	is a string with possible values: FATAL, ERROR, WARN, INFO, DEBUG, TRACE

Value

No return value, called for the side effect of setting the log level

Examples

```
## Not run:
setLogLevel("INFO")
## End(Not run)
```

setProjectLines — *Sets the lines of the project data files that should be used when running a project.*

Description

You have to call the function after initSimplace but before openProject.

Usage

```
setProjectLines(simplace, lines)
```

Arguments

simplace	handle to the SimplaceWrapper object returned by initSimplace
lines	either a vector of integers or a string of numbers separated by commas

Value

No return value, called for the side effect of selecting project to be run

Examples

```
## Not run:
setProjectLines(simplace, "1,3,6,9-17,33")
setProjectLines(simplace, c(1,2,3,9:17,33))
## End(Not run)
```

---

setSimplaceDirectories

Set working-, output-, projects- and data-directory

Description

One can specify all or only some of the directories. Only the directories specified will be set.

Usage

```
setSimplaceDirectories(
  simplace,
  WorkDir = nullString,
  OutputDir = nullString,
  ProjectsDir = nullString,
  DataDir = nullString
)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`WorkDir`	working directory where solutions, projects and data resides (_WORKDIR_)
`OutputDir`	directory for output (_OUTPUTDIR_)
`ProjectsDir`	optional directory for project data (_PROJECTSDIR_)
`DataDir`	optional directory for data (_DATADIR_)

Value

No return value, called for the side effect of setting framework directories

See Also

`getSimplaceDirectories`

`setSimulationValues` *Changes values of the current simulation*

Description

Sets values of arbitrary SimVariables in a simplace simulation. Useful if you want to couple simplace with another simulation and interchange values daily.

Usage

```
setSimulationValues(simplace, parameterList = NULL, simulationNumber = 1)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`parameterList`	a list with the parameter name as key and parametervalue as value
`simulationNumber`	number of simulation in the queue whose parameters should be set (default first simulation)

Value

No return value, called for the side effect of changing parameters in the current simulation

Examples

```
## Not run:
for(i in 1:365)
{
  param <- list(vBaseLUE=3.0 + i/2000)
  setSimulationValues(simplace,param)
  stepSimulation(simplace)
}

## End(Not run)
```

`setSlotCount` *Sets number of used CPUs*

Description

Sets the number of processors that are used parallel. The function can be used only after `initSimplace` has been called.

Usage

```
setSlotCount(count)
```

Arguments

`count`	number of processors

Value

No return value, called for the side effect of setting the number of processors used for simulation runs

---

`simplace` *simplace: Interface to use the modelling framework 'SIMPLACE'*

Description

Interface to interact with the modelling framework 'SIMPLACE' and to parse the results of simulations

Details

Package needs a Java Runtime Environment as well as an installation of 'SIMPLACE'. See www.simplace.net for more information about 'SIMPLACE'.

Author(s)

Gunther Krauss

References

www.simplace.net

See Also

Useful links:

- `https://github.com/gk-crop/simplace_rpkg/`
- `https://r-forge.r-project.org/projects/simplace/`

Examples

```
     ## Not run:
       SimplaceInstallationDir <- "D:/java/simplace/"

       SimplaceWorkDir <- "D:/java/simplace/simplace_run/simulation/"
       SimplaceOutputDir <-  "D:/java/simplace/simplace_run/output/"

      Solution <- "D:/java/simplace/simplace_run/simulation/gk/solution/complete/Complete.sol.xml"

       simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)

       openProject(simplace, Solution)

       parameter <- list()
       parameter$vTempLimit <- 32

       simid <- createSimulation(simplace,parameter)
       runSimulations(simplace)

       result <- getResult(simplace,"DIAGRAM_OUT", simid);

       closeProject(simplace)

       resultlist <- resultToList(result)
       resultframe <- resultToDataframe(result)

   ## End(Not run)
```

`stepAllSimulations` *Run all simulations in queue stepwise*

Description

Performs `count` steps of the simulation and returns the values from the actual variable map. Can be called consecutively.

Usage

```
stepAllSimulations(simplace, count = 1, filter = NULL, parameterLists = NULL)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`count`	number of steps to be performed
`filter`	vector of the variable names to be included in the result. If not set, all variables are returned
`parameterLists`	a list of parameter lists for each simulation

Value

handle to an array of data containers which has to be processed afterwards

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
createSimulation(simplace)
vm <- stepAllSimulations(simplace,count=22)
vm_s <- stepAllSimulations(simplace,filter=c("CURRENT.DATE","LintulBiomass.sWSO"),count=18)
closeProject(simplace)
## End(Not run)
```

`stepSimulation` *Run simulation stepwise*

Description

Performs `count` steps of the simulation and returns the values from the actual variable map. Can be called consecutively.

Usage

```
stepSimulation(
  simplace,
  count = 1,
  filter = NULL,
  parameterList = NULL,
  simulationNumber = 1
)
```

Arguments

`simplace`	handle to the SimplaceWrapper object returned by `initSimplace`
`count`	number of steps to be performed
`filter`	vector of the variable names to be included in the result. If not set, all variables are returned
`parameterList`	list of parameter values indexed by parameter name
`simulationNumber`	number of simulation in the queue that should be run stepwise (default first simulation)

Value

handle to the data container which has to be processed afterwards

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
createSimulation(simplace)
vm <- stepSimulation(simplace,count=22)
vm_s <- stepSimulation(simplace,filter=c("CURRENT.DATE","LintulBiomass.sWSO"),count=18)
closeProject(simplace)
## End(Not run)
```

`varmapToList` *Converts the varmap to a list*

Description

Converts the varMap to a list. All elements are converted to appropriate R objects. Arrays are expanded to vectors by default.

Usage

```
varmapToList(varmap, expand = TRUE)
```

Arguments

`varmap`	the varMap returned by `stepSimulation`
`expand`	if `TRUE` expand array objects to vector.

Value

list with parameter name as key and parameter value as value

Examples

```
## Not run:
simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
openProject(simplace, Solution)
createSimulation(simplace)
varmap <- stepSimulation(simplace,count=22)
closeProject(simplace)
varlist <- varmapToList(varmap)
varlist$startdate - 365
varlist$LintulBiomass.sWSO
## End(Not run)
```

Index