

Package: `simplaceUtil` (via `r-universe`)

September 8, 2024

Title Provides Utility Functions and ShinyApps to work with the modeling framework 'SIMPLACE'

Version 0.6.9

Date 2024-07-10

Description Provides Utility Functions and ShinyApps to work with the modeling framework 'SIMPLACE'. It visualises components of a solution, runs simulations and displays results.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1)

Imports DiagrammeR, rlang, dplyr, shiny, stringr, tidyr, xml2, shinyFiles, DT, ggplot2, simplace

VignetteBuilder knitr

Suggests knitr, rmarkdown, rsvg, DiagrammeRsvg

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository <https://gk-crop.r-universe.dev>

RemoteUrl <https://github.com/gk-crop/simplaceUtil>

RemoteRef HEAD

RemoteSha 10fabca0b30f25669afc3c7d0a340b8da65c9be5

Contents

<code>addComponentInput</code>	3
<code>addCSVOutput</code>	4
<code>addMemoryOutput</code>	5
<code>addOutputVariable</code>	5
<code>addTimingSimComponent</code>	6
<code>addUserVariable</code>	7
<code>changeAllOutputTypesToMemory</code>	8
<code>changeOutputType</code>	8

componentsToGraph	9
createCodeStubsForSimVariables	9
createResourceStubsFromCsv	10
createResourceStubsFromXml	11
enhanceFunctionWithBoundaries	12
fetchDescriptionFromWebsite	14
fetchSimComponentlistFromWebsite	14
fetchSimVariablesFromWebsite	15
filterEdges	15
getComponents	16
getElementsFromSolutionFile	16
getLinkingFromComponent	17
getLinkingToComponent	17
getLinks	18
getMemoryOutputIds	18
getNeighborhood	19
getOutputFileNames	20
getSolutionFromFile	20
getSolutionFromText	21
getSolutionInfoAsDataframe	21
getTextFromSolution	22
getUserVariables	22
parseDate	23
plotLayeredOutput	23
plotScalarOutput	24
removeComponentInput	25
removeNonMemoryOutputs	25
removeOutput	26
removeOutputVariable	26
replaceVariable	27
replaceVariablesWithValues	27
runSimplaceGuiApp	28
setInputValue	28
setInputValueForCategory	29
simplaceUtil	29
solutionToGraph	30
swapComponents	30
transformLayeredData	31
writeSolutionToFile	31

addComponentInput	<i>Adds an input to a sim component.</i>
-------------------	--

Description

Notice: One can only add inputs that are defined by the sim component. If source is not given, then the parameter value is used.

Usage

```
addComponentInput(  
    sol,  
    componentid,  
    id,  
    source = NULL,  
    value = NULL,  
    datatype = NULL,  
    unit = NULL,  
    description = NULL  
)
```

Arguments

sol	solution object
componentid	id of sim component
id	id of variable
source	source of variable
value	value of variable (is used only if source is not given or NULL)
datatype	datatype (optional)
unit	unit (optional)
description	short description (optional)

Value

modified solution object

addCSVOutput	<i>Adds a new CSV output to solution.</i>
--------------	---

Description

Any output with the same id will be removed. The added output has no output variables. One has to add new variables via addOutputVariable.

Usage

```
addCSVOutput(  
    sol,  
    filename,  
    outputid,  
    frequency = "DAILY",  
    rule = NULL,  
    resetrule = NULL,  
    cachesize = 10,  
    divider = ","  
)
```

Arguments

sol	solution object
filename	name of the output file
outputid	id of new output
frequency	one of DAILY, YEARLY, BOOLEAN, COMPLEX
rule	optional rule, when frequency is BOOLEAN or COMPLEX
resetrule	optional resetrule, when frequency is BOOLEAN or COMPLEX
cachesize	optional cachesize
divider	character that is used as a divider for the csv file

Value

modified solution object

addMemoryOutput	<i>Adds a new memory output to solution.</i>
-----------------	--

Description

Any output with the same id will be removed. The added output has no output variables. One has to add new variables via addOutputVariable.

Usage

```
addMemoryOutput(  
    sol,  
    outputid,  
    frequency = "DAILY",  
    rule = NULL,  
    resetrule = NULL,  
    cachesize = 10  
)
```

Arguments

sol	solution object
outputid	id of new output
frequency	one of DAILY, YEARLY, BOOLEAN, COMPLEX
rule	optional rule, when frequency is BOOLEAN or COMPLEX
resetrule	optional resetrule, when frequency is BOOLEAN or COMPLEX
cachesize	optional cachesize

Value

modified solution object

addOutputVariable	<i>Adds output variable to an existing output</i>
-------------------	---

Description

Adds output variable to an existing output

Usage

```

addOutputVariable(
    sol,
    outputid,
    id,
    rule,
    datatype,
    mode = NULL,
    unit = NULL,
    description = NULL
)

```

Arguments

sol	solution object
outputid	id of output where the variable should be added
id	name of the new variable
rule	rule for the variable
datatype	of the variable
mode	one of FIRST, LAST, AVG, SUM (optional)
unit	unit of the variable (optional)
description	short description (optional)

Value

modified solution object

addTimingSimComponent *Adds elements for timing the execution time of sim components (and transformers)*

Description

Adds elements for timing the execution time of sim components (and transformers)

Usage

```

addTimingSimComponent(
    sol,
    filename = NULL,
    componentlist = NULL,
    interfaceid = "automatic_timing_interface",
    outputid = "automatic_timing_output",
    simcomponentid = "AutomaticTiming"
)

```

Arguments

sol	solution object
filename	optional filename to write the timing information to csv file
componentlist	optional list of component ids (if empty, all components will be timed)
interfaceid	id for the interface (optional)
outputid	id for the output (optional)
simcomponentid	id for the timing simcomponent (optional)

addUserVariable	<i>Adds an user variable to the variables section</i>
-----------------	---

Description

Adds an user variable to the variables section

Usage

```
addUserVariable(sol, id, value, datatype, unit = NULL, description = NULL)
```

Arguments

sol	solution object
id	id of the variable
value	value of the variable
datatype	of the variable
unit	unit (optional)
description	short description (optional)

Value

modified solution object

changeAllOutputTypesToMemory

Changes all non-MEMORY outputs from solution to MEMORY outputs.

Description

When running large amount of runs (e.g. calibration) it's recommended to avoid to write outputs on disk.

Usage

```
changeAllOutputTypesToMemory(sol)
```

Arguments

sol	solution object
-----	-----------------

Value

modified solution object

changeOutputType

Changes the type of an output from CSV to MEMORY or vice versa

Description

Changes the type of an output from CSV to MEMORY or vice versa

Usage

```
changeOutputType(sol, outputid, type, filename = NULL, divider = ",")
```

Arguments

sol	solution object
outputid	id of output to change
type	one of 'CSV' or 'MEMORY'
filename	name of the output file if type changes to 'CSV'
divider	character that is used as a divider for the csv file

Value

modified solution object

componentsToGraph	<i>Creates a graph from component and links dataframe</i>
-------------------	---

Description

Creates a graph from component and links dataframe

Usage

```
componentsToGraph(comp, links, showinterfaces = FALSE, ...)
```

Arguments

comp	components dataframe
links	links dataframe
showinterfaces	if TRUE, include interfaces in graph
...	options passed to DiagrammeR::create_graph()

Value

graph object of class `dgr_graph`

createCodeStubsForSimVariables	<i>Create java and xml code stubs from a csv file</i>
--------------------------------	---

Description

The function creates Java code stubs for defining the fields, adding variables and initialising values. It creates XML subs for parameter file, as well as resource definition for parameter and input data, inputs for simcomponent and outputs.

Usage

```
createCodeStubsForSimVariables(  
  variables,  
  component = "MyComponent",  
  outfolder = NULL,  
  ...  
)
```

Arguments

variables	dataframe or path to csv file with variables
component	name of SimComponent
outfolder	optional, if given code is written the folder's files
...	parameters passed to read.table (e.g. sep, dec etc.)

Details

The dataframe / CSV file should have columns

- contenttype (one of: constant, input, state, rate, out)
- id
- description
- datatype (Simplace Datatype or corresponding Java Datatype)
- unit
- min
- max
- default

Value

list of code snippets

createResourceStubsFromCsv

Creates XML stubs for interface and resource section from CSV file structure

Description

Creates XML stubs for interface and resource section from CSV file structure

Usage

```
createResourceStubsFromCsv(
  filename,
  id,
  sep = ",",
  keyvals = NULL,
  arraycolumns = NULL,
  frequence = "DAILY",
  rule = NULL,
  data = NULL
)
```

Arguments

filename	name of the CSV file
id	id for the resource
sep	separator for CSV file
keyvals	named vector of column indices or names that act as key column
arraycolumns	vector of column indices or names that are arrays
frequence	frequence attribute for the resource
rule	rule attribute for the resource
data	optional data.frame (otherwise data will be loaded from file)

Value

list of two strings ('interface' and 'resource')

Examples

```
stubs <- createResourceStubsFromCsv(  
  filename = system.file("input", "weather.csv", package="simplaceUtil"),  
  id = "weather",  
  sep = ",",  
  keyvals = c("CURRENT.DATE" = "Date")  
)  
cat(stubs$interface)  
cat(stubs$resource)  
  
stubs <- createResourceStubsFromCsv(  
  filename = system.file("input", "soil.csv", package="simplaceUtil"),  
  id = "soil",  
  sep = ";",  
  keyvals = c("vSoilType" = "soiltype"),  
  arraycolumns = 6:20  
)  
cat(stubs$resource)
```

createResourceStubsFromXml

Creates XML stubs for interface and resource section from XML file structure

Description

Creates XML stubs for interface and resource section from XML file structure

Usage

```
createResourceStubsFromXml(
  filename,
  id,
  keyvals = NULL,
  frequency = "DAILY",
  rule = NULL,
  xmlnode = NULL
)
```

Arguments

filename	name of the xml file
id	id for the resource
keyvals	named vector of column indices or names that act as key column
frequency	frequency attribute for the resource
rule	rule attribute for the resource
xmlnode	optional xml_node (otherwise xml will be read from filename)

Value

list of two strings ('interface' and 'resource')

Examples

```
stubs <- createResourceStubsFromXml(
  filename = system.file("input", "crop.xml", package="simplaceUtil"),
  id="soil",
  keyvals = c("vSoilType"=1)
)
cat(stubs$resource)
```

enhanceFunctionWithBoundaries

Modifies function to return a penalty value for parameters outside boundaries

Description

The method takes a function as well as values for lower and upper boundaries and returns a modified function. The modified function returns the value of the original function when the parameters are within boundaries and the penalty value otherwise.

Usage

```
enhanceFunctionWithBoundaries(  
  fun,  
  l_bound,  
  u_bound,  
  penalty_value = Inf,  
  boundary_fun = NULL,  
  param_pos = 1,  
  ...  
)
```

Arguments

fun	function to be modified
l_bound	vector with lower boundary values
u_bound	vector with upper boundary values
penalty_value	value if parameter outside boundaries
boundary_fun	optional function for complex boundary conditions
param_pos	argument position of the parameter
...	arguments passed to original function

Details

Optionally an own function can be supplied to calculate whether the parameter is valid. The boundary function must take 3 arguments: parameter, lower boundary and upper boundary and must return TRUE or FALSE.

A main use case of this method are optimisation / calibration tasks. If the optimisation method and the function to optimise are both ignorant to boundaries one can turn the function into a boundary sensitive one.

Value

a modified function that considers boundaries

Examples

```
sqrt_bd <- enhanceFunctionWithBoundaries(sqrt, 0, 10)  
sqrt_bd(-1)  
sqrt_bd(1)  
sqrt_bd(11)
```

fetchDescriptionFromWebsite

Fetches the Description (as HTML source code) of a SimComponent from Simplace Website

Description

Fetches the Description (as HTML source code) of a SimComponent from Simplace Website

Usage

```
fetchDescriptionFromWebsite(class, version = "current")
```

Arguments

class	class name of the SimComponent
version	Use current, trunk or numeric version #.#

Value

string with the description text (including HTML tags)

fetchSimComponentlistFromWebsite

Fetches the list of SimComponents from the Simplace Website

Description

Fetches the list of SimComponents from the Simplace Website

Usage

```
fetchSimComponentlistFromWebsite(version = "current")
```

Arguments

version	Use current, trunk or numeric version #.#
---------	---

Value

character vector with the class names of all SimComponents

`fetchSimVariablesFromWebsite`*Fetches the SimVariables table for a SimComponent from Simplace Website*

Description

Fetches the SimVariables table for a SimComponent from Simplace Website

Usage

```
fetchSimVariablesFromWebsite(class, version = "current")
```

Arguments

<code>class</code>	class name of the SimComponent
<code>version</code>	Use current, trunk or numeric version #.#

Value

a data.frame with the SimVariables

`filterEdges`*Selects edges of specific timestep*

Description

Selects edges of specific timestep

Usage

```
filterEdges(graph, linkage = "allsteps")
```

Arguments

<code>graph</code>	graph object of class <code>dgr_graph</code>
<code>linkage</code>	all steps, same step or previous step

Value

graph object of class `dgr_graph`

<code>getComponents</code>	<i>Gets components for a solution</i>
----------------------------	---------------------------------------

Description

Gets components for a solution

Usage

```
getComponents(x)
```

Arguments

x	xml object (solution)
---	-----------------------

Value

data.frame with the solution components (resources, sim components, outputs)

<code>getElementsFromSolutionFile</code>	<i>Get component and links dataframe from solution file</i>
--	---

Description

Get component and links dataframe from solution file

Usage

```
getElementsFromSolutionFile(file)
```

Arguments

file	solution
------	----------

Value

list with solution (xml2 object) and components, links and variables data.frame

`getLinkingFromComponent`*Get components that are linked from given component*

Description

Get components that are linked from given component

Usage

```
getLinkingFromComponent(  
    graph,  
    names,  
    distance = 50,  
    linkage = "allsteps",  
    type = "all"  
)
```

Arguments

graph	graph object of class dgr_graph
names	names (id attributes) of the selected components
distance	maximum distance from given components
linkage	all steps, same step or previous step
type	type of components

Value

graph object of class dgr_graph

`getLinkingToComponent` *Get components that link to given components*

Description

Get components that link to given components

Usage

```
getLinkingToComponent(  
    graph,  
    names,  
    distance = 50,  
    linkage = "allsteps",  
    type = "all"  
)
```

Arguments

graph	graph object of class <code>dgr_graph</code>
names	names (id attributes) of selected components
distance	maximum distance from given components
linkage	all steps, same step or previous step
type	type of components

Value

graph object of class `dgr_graph`

`getLinks` *Get the dataframe of links between components*

Description

Get the dataframe of links between components

Usage

```
getLinks(x, df)
```

Arguments

x	xml object (solution)
df	data.frame of components

Value

data.frame with the linked variables between components

`getMemoryOutputIds` *Get ids of memory outputs*

Description

Get ids of memory outputs

Usage

```
getMemoryOutputIds(comp)
```

Arguments

comp	components dataframe
------	----------------------

Value

character vector with the memory output ids

getNeighborhood	<i>Get the simcomponents in the neighborhood of given components</i>
-----------------	--

Description

Get the simcomponents in the neighborhood of given components

Usage

```
getNeighborhood(
  graph,
  names,
  distance = 1,
  linkage = "allsteps",
  type = "all",
  set_op = "union"
)
```

Arguments

graph	graph object of class dgr_graph
names	names (id attributes) of the selected sim components
distance	maximum number of steps from given component
linkage	all steps, same step or previous step
type	type of component
set_op	"union" - all neighbours of selected components or "intersect" - common neighbours of all selectected components

Value

graph object of class dgr_graph

getOutputFileNames *Get filenames of file outputs*

Description

Get filenames of file outputs

Usage

```
getOutputFileNames(comp, variables, additional = NULL)
```

Arguments

comp	components dataframe
variables	variables dataframe
additional	additional variables as named vector c("var1"="value1", ...), useful for directory placeholder

Value

named character vector with the filenames

getSolutionFromFile *Reads a solution from file*

Description

Reads a solution from file

Usage

```
getSolutionFromFile(file)
```

Arguments

file	filename of the solution
------	--------------------------

Value

parsed solution as xml_document

`getSolutionFromText` *Reads a solution from text*

Description

Reads a solution from text

Usage

`getSolutionFromText(text)`

Arguments

`text` string with xml markup

Value

parsed solution as `xml_document`

`getSolutionInfoAsDataframe`
Get info for a solution

Description

Get info for a solution

Usage

`getSolutionInfoAsDataframe(file, workdir)`

Arguments

`file` solution file
`workdir` working directory

Value

data.frame with all solution components as well as meta data for solution file

getTextFromSolution *Converts a solution to text*

Description

Converts a solution to text

Usage

```
getTextFromSolution(sol)
```

Arguments

sol solution object (xml_document)

Value

text with xml code

getUserVariables *Gets user variables for a solution*

Description

Gets user variables for a solution

Usage

```
getUserVariables(x)
```

Arguments

x xml object (solution)

Value

data.frame with the user defined variable

parseDate	<i>Converts date (class character) column from output file to column of class Date</i>
-----------	--

Description

Converts date (class character) column from output file to column of class Date

Usage

```
parseDate(  
  data,  
  newName = "CURRENT.DATE",  
  format = "%d.%m.%Y",  
  oldName = "CURRENT.DATE"  
)
```

Arguments

data	data.frame
newName	name of the transformed date column
format	date format
oldName	name of the column that holds the date to be transformed

Value

data.frame with date column of class Date

plotLayeredOutput	<i>Plots layered output</i>
-------------------	-----------------------------

Description

Plots layered output

Usage

```
plotLayeredOutput(  
  data,  
  column,  
  simulationid = NULL,  
  date_from = NULL,  
  date_to = NULL,  
  datecol = "CURRENT.DATE",  
  nrow = NULL,  
  ncol = NULL  
)
```

Arguments

data	data from memory output or file
column	column name used for the fill color
simulationid	plot only data for simulationids
date_from	simulation date from where values are plotted
date_to	simulation date until values are plotted
datecol	column name for date (default CURRENT.DATE)
nrow	number of panel rows when plotting multiple simulation ids
ncol	number of panels in a row when plotting multiple simulation ids

plotScalarOutput	<i>Plots scalar output</i>
------------------	----------------------------

Description

Plots scalar output

Usage

```
plotScalarOutput(
  data,
  column_x,
  columns_y,
  simulationid = NULL,
  date_from = NULL,
  date_to = NULL,
  datecol = "CURRENT.DATE",
  nrow = NULL,
  ncol = NULL
)
```

Arguments

data	data from memory output or file
column_x	column name for x values
columns_y	column name(s) for y values (vector of names)
simulationid	plot only data for simulationids
date_from	simulation date from where values are plotted
date_to	simulation date until values are plotted
datecol	column name for date (default CURRENT.DATE)
nrow	number of panel rows when plotting multiple simulation ids
ncol	number of panels in a row when plotting multiple simulation ids

removeComponentInput *Removes an input for component*

Description

Removes an input for component

Usage

```
removeComponentInput(sol, componentid, id)
```

Arguments

sol	solution object
componentid	id of the sim component
id	id of the input

Value

modified solution object

removeNonMemoryOutputs
Removes all non-MEMORY outputs from solution.

Description

When running large amount of runs (e.g. calibration) it's recommended to avoid to write outputs on disk.

Usage

```
removeNonMemoryOutputs(sol)
```

Arguments

sol	solution object
-----	-----------------

Value

modified solution object

removeOutput	<i>Removes output with given id</i>
--------------	-------------------------------------

Description

Notice: the interface for the output will also be removed.

Usage

```
removeOutput(sol, outputid)
```

Arguments

sol	solution object
outputid	id of output to remove

Value

modified solution object

removeOutputVariable	<i>Removes an output variable from a given output</i>
----------------------	---

Description

Removes an output variable from a given output

Usage

```
removeOutputVariable(sol, outputid, id)
```

Arguments

sol	solution object
outputid	id of output from where the variable should be removed
id	name of the variable to remove

Value

modified solution object

replaceVariable	<i>Replaces a variable with another.</i>
-----------------	--

Description

Notice: there is no check, whether the ids exist. Variables are replaced literarily when they are the only content of an attribute or element content. In other attributes and element contents $\${oldid}$ will be replaced by $\${newid}$

Usage

```
replaceVariable(sol, oldid, newid)
```

Arguments

sol	solution object
oldid	id of variable to be replaced
newid	id of the replacing variable

Details

Notice that the id of variables from resources, SimComponents etc. should be prefixed by the enclosing resource's, SimComponent's etc. id.

Value

modified solution object

replaceVariablesWithValues	<i>Replaces variables with content</i>
----------------------------	--

Description

Replaces variables with content

Usage

```
replaceVariablesWithValues(text, variables, additional = NULL)
```

Arguments

text	text to replace
variables	variables dataframe
additional	additional variables as named vector $c("var1"="value1", \dots)$, useful for directory placeholder

runSimplaceGuiApp	<i>Runs Shiny app</i>
-------------------	-----------------------

Description

Runs Shiny app

Usage

```
runSimplaceGuiApp(...)
```

Arguments

... parameters passed to shiny::runApp

setInputValue	<i>Sets the value of an input for a given parent element</i>
---------------	--

Description

Sets the value of an input for a given parent element

Usage

```
setInputValue(sol, parentid, id, value, datatype = NULL)
```

Arguments

sol	solution object
parentid	id of input parent
id	id of the input
value	new value
datatype	datatype (optional)

Value

modified solution object

`setInputValueForCategory`*Sets the value of an input in all elements of the given category*

Description

It's main use is to change consistently inputs of all transformers, e.g. layerthickness. It can also be used to set sim component inputs to a fixed value.

Usage

```
setInputValueForCategory(sol, category, id, value, datatype = NULL)
```

Arguments

<code>sol</code>	solution object
<code>category</code>	tag of categories containing inputs, e.g. transform or simcomponent
<code>id</code>	id of the input
<code>value</code>	new value
<code>datatype</code>	datatype (optional)

Value

modified solution object

`simplaceUtil`*Package with utility function for working with Simplace*

Description

Provides functions to work with the modeling framework Simplace

Details

- get elements of a simplace solution as dataframes
- visualise the structure of a solution as a graph
- helper functions for transforming output data
- simplified plot functions for simulation output variables
- shinyApp to visualise graphs, run simulations and plot results

Author(s)

Gunther Krauss

Examples

```
## Not run:
# run the GUI
runSimplaceGuiApp()

## End(Not run)

# visualise the solution structure as graph
graph <- solutionToGraph(system.file("solution", "Yield.sol.xml",
  package = "simplaceUtil"))
DiagrammeR::render_graph(graph)
```

solutionToGraph	<i>Creates a graph from a solution</i>
-----------------	--

Description

Creates a graph from a solution

Usage

```
solutionToGraph(file, showinterfaces = FALSE, ...)
```

Arguments

file	solution file
showinterfaces	if true include interfaces
...	options passed to DiagrammeR::create_graph()

Value

graph object of class dgr_graph

swapComponents	<i>Swaps the order of SimComponents in the solution</i>
----------------	---

Description

Rearranges the order of SimComponents. All components that are mentioned in the vector order will be rearranged according to their position in the order vector. All other components remain on the same position. E. g. an order of c(5, 6, 1, 3) will put components on position 1,3,5,6 to position 5, 6, 1, 3.

Usage

```
swapComponents(sol, order)
```

Arguments

sol	solution object
order	a vector of component positions

Value

modified solution object

`transformLayeredData` *Transform layered output data in long format*

Description

Transform layered output data in long format

Usage

```
transformLayeredData(data)
```

Arguments

data	dataframe
------	-----------

`writeSolutionToFile` *Writes solution to file*

Description

Writes solution to file

Usage

```
writeSolutionToFile(sol, file)
```

Arguments

sol	solution object (xml_document)
file	filename for the solution

Value

nothing, function writes a file as a side effect

Index

addComponentInput, 3
addCSVOutput, 4
addMemoryOutput, 5
addOutputVariable, 5
addTimingSimComponent, 6
addUserVariable, 7

changeAllOutputTypesToMemory, 8
changeOutputType, 8
componentsToGraph, 9
createCodeStubsForSimVariables, 9
createResourceStubsFromCsv, 10
createResourceStubsFromXml, 11

DiagrammeR::create_graph(), 9

enhanceFunctionWithBoundaries, 12

fetchDescriptionFromWebsite, 14
fetchSimComponentlistFromWebsite, 14
fetchSimVariablesFromWebsite, 15
filterEdges, 15

getComponents, 16
getElementsFromSolutionFile, 16
getLinkingFromComponent, 17
getLinkingToComponent, 17
getLinks, 18
getMemoryOutputIds, 18
getNeighborhood, 19
getOutputFileNames, 20
getSolutionFromFile, 20
getSolutionFromText, 21
getSolutionInfoAsDataframe, 21
getTextFromSolution, 22
getUserVariables, 22

parseDate, 23
plotLayeredOutput, 23
plotScalarOutput, 24

removeComponentInput, 25
removeNonMemoryOutputs, 25
removeOutput, 26
removeOutputVariable, 26
replaceVariable, 27
replaceVariablesWithValues, 27
runSimplaceGuiApp, 28

setInputValue, 28
setInputValueForCategory, 29
simplaceUtil, 29
simplaceUtil-package (simplaceUtil), 29
solutionToGraph, 30
swapComponents, 30

transformLayeredData, 31

writeSolutionToFile, 31