Package 'rsyncrosim'

July 25, 2019

Type Package

Title The R Interface to SyncroSim: http://syncrosim.com/

Version 1.1.3

Author ApexRMS <rsyncrosim@apexrms.com> and Josie Hughes <josie.hughes@canada.ca>

Maintainer ApexRMS <rsyncrosim@apexrms.com>

Description rsyncrosim provides an interface to SyncroSim, a generalized framework for managing scenario-based datasets. Simulation models can be added to SyncroSim in order to transform these datasets, taking advantage of general features such as defining scenarios of model inputs, running Monte Carlo simulations, and summarizing model outputs.

License GPL-3 | file LICENSE

LazyData TRUE

Encoding UTF-8

Imports methods,
DBI,
RSQLite,
Rcpp,
rgdal,
raster

Suggests knitr,
testthat,
ggplot2

SystemRequirements SyncroSim (>=2.1.0)

Collate 'AAAClassDefinitions.R'
'addModule.R'
'addPackage.R'
'addPackageFile.R'
'addRow.R'
'addon.R'
'backup.R'
'basePackage.R'
'breakpoint.R'
'command.R'
'dataframeFromSSim.R'
'datasheet.R'
'datasheetRaster.R'

'datasheets.R'
'dateModified.R'
'delete.R'
'deleteModule.R'
'deletePackage.R'
'dependency.R'
'description.R'
'disableAddon.R'
'enableAddon.R'
'filepath.R'
'getFromXProjScn.R'
'internalHelpers.R'
'name.R'
'scenarioId.R'
'projectId.R'
'sqlStatement.R'
'scenario.R'
'project.R'
'ssimLibrary.R'
'session.R'
'internalWrappers.R'
'mergeDependencies.R'
'model.R'
'module.R'
'owner.R'
'package.R'
'parentId.R'
'print.R'
'printCmd.R'
'readOnly.R'
'rsyncrosim.R'
'run.R'
'runLog.R'
'saveDatasheet.R'
'silent.R'
'ssimEnvironment.R'
'ssimUpdate.R'
'updatePackage.R'
'version.R'

RoxygenNote 6.0.1

VignetteBuilder knitr

R topics documented:

`addBreakpoint` *Add a Scenario breakpoint.*

Description

When the Scenario is run the breakpoint's callback function will be called for the specified iterations or timesteps.

Usage

```
addBreakpoint(x, transformerName, breakpointType, arguments, callback)

## S4 method for signature 'Scenario'
addBreakpoint(x, transformerName, breakpointType,
  arguments, callback)
```

Arguments

`x`	A SyncroSim Scenario
`transformerName`	A Stochastic Time Transformer (e.g. stsim_Runtime)
`breakpointType`	bi: before iteration; ai: after iteration; bt:before timestep; at: after timestep
`arguments`	A vector of timesteps or iterations e.g. c(1,2)
`callback`	A function to be called when the breakpoint is hit

Details

Breakpoints are only supported for Stochastic Time Transformers.

Value

A SyncroSim Scenario with an updated list of breakpoints

Examples

```
## Not run:
callbackFunction <- function(x, iteration, timestep) {
  print(paste0('Breakpoint hit: ', scenarioId(x)))
}

myScenario = addBreakpoint(myScenario, "stsim_Runtime", "bi", callbackFunction)

## End(Not run)
```

addModule *Add module*

Description

Add module or modules to SyncroSim Deprecated. See: addPackage and addPackageFile

Usage

```
addModule(filename, session = NULL)

## S4 method for signature 'character'
addModule(filename, session = NULL)
```

Arguments

filename	Character string or vector of these. The path to an .ssimpkg file on disk, or a vector of filepaths.
session	Session.

addon *addon(s) of an SsimLibrary or Session*

Description

The addon(s) of an SsimLibrary or Session.

Usage

```
addon(ssimObject)

## S4 method for signature 'character'
addon(ssimObject)

## S4 method for signature 'missingOrNULL'
addon(ssimObject)

## S4 method for signature 'Session'
addon(ssimObject)

## S4 method for signature 'SsimObject'
addon(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario or Session.

Value

A dataframe of addons.

Examples

```
## Not run:
addon(ssimLibrary(name="mylib"))

## End(Not run)
```

---

addPackage *Adds a package to SyncroSim*

Description

Adds a package to SyncroSim

Usage

```
addPackage(name, session = NULL)

## S4 method for signature 'ANY,character'
addPackage(name, session = NULL)

## S4 method for signature 'ANY,missingOrNULL'
addPackage(name, session = NULL)

## S4 method for signature 'ANY,Session'
addPackage(name, session = NULL)
```

Arguments

name	Character string. The name of the package to install from the online package server.
session	Session.

---

addPackageFile *Adds a package to SyncroSim*

Description

Adds a package to SyncroSim

Usage

```
addPackageFile(filename, session = NULL)

## S4 method for signature 'ANY,character'
addPackageFile(filename, session = NULL)

## S4 method for signature 'ANY,missingOrNULL'
addPackageFile(filename, session = NULL)

## S4 method for signature 'ANY,Session'
addPackageFile(filename, session = NULL)
```

Arguments

filename	Character string. The path to a SyncroSim package file.
session	Session.

---

addRow *Add row(s) to a dataframe.*

Description

Adds row(s) to a dataframe.

Usage

```
addRow(targetDataframe, value)

## S4 method for signature 'data.frame'
addRow(targetDataframe, value)
```

Arguments

targetDataframe	Dataframe.
value	Dataframe, character string vector, or list. Columns in value should be a subset of columns in targetDataframe.

Details

Preserves the types and factor levels of the targetDataframe. Fills missing values if possible using factor levels. If value is a named vector or list, it will be converted to a single row dataframe. If value is an unnamed vector or list, the number of elements should equal the number of columns in the targetDataframe; elements are assumed to be in same order as dataframe columns.

Value

A dataframe with new rows.

---

backup *Backup an SsimLibrary.*

Description

Backup an SsimLibrary.

Usage

```
backup(ssimObject)

## S4 method for signature 'character'
backup(ssimObject)

## S4 method for signature 'SsimObject'
backup(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.

---

basePackage *Installed base packages*

Description

Base packages installed with this version of SyncroSim

Usage

```
basePackage(ssimObject = NULL)

## S4 method for signature 'character'
basePackage(ssimObject = NULL)

## S4 method for signature 'missingOrNULL'
basePackage(ssimObject = NULL)

## S4 method for signature 'Session'
basePackage(ssimObject = NULL)

## S4 method for signature 'SsimLibrary'
basePackage(ssimObject = NULL)
```

Arguments

ssimObject	Session or SsimLibrary.

Value

A dataframe of base packages (for Session) or named vector of character strings (for SsimLibrary)

---

breakpoint *Lists the breakpoints for a Scenario.*

Description

Lists the breakpoints for a Scenario.

Usage

```
breakpoint(x)

## S4 method for signature 'Scenario'
breakpoint(x)
```

Arguments

x	A SyncroSim Scenario

command *SyncroSim console command*

Description

Issues a command to the SyncroSim console and returns the output.

Usage

```
command(args, session = NULL, program = "SyncroSim.Console.exe", wait = T)
```

Arguments

args	Character string, named list, named vector, unnamed list, or unnamed vector. Arguments for the SyncroSim console. See details.
session	Session. If NULL, a default session will be used.
program	Character. The name of the target SyncroSim executable. Options include SyncroSim.Console.exe (default), SyncroSim.Server.exe, SyncroSim.PackageManager.exe and SyncroSim.Multiband.exe.
wait	Logical. If TRUE (default) R will wait for the command to finish before proceeding. Note that silent(session) is ignored if wait=F.

Details

Example args, and the resulting character string passed to the SyncroSim console:

- Character string e.g. "–create –help": "–create –help"
- Named list or named vector e.g. list(name1=NULL,name2=value2): "–name1 –name2=value2"
- Unnamed list or unnamed vector e.g. c("create","help"): "–create –help"

Value

Output from the SyncroSim program.

Examples

```
#Use a default session to create a new library in the current working directory.
args = list(create=NULL,library=NULL,name=paste0(getwd(),"/temp.ssim"),package="stsim")
output = command(args,session=session(printCmd=TRUE))
output

#Three different ways to provide args to command
command(c("create","help"))
command("--create --help")
command(list(create=NULL,help=NULL))
```

datasheet *Get a datasheet*

Description

Gets SyncroSim datasheet.

Usage

```
datasheet(ssimObject, name = NULL, project = NULL, scenario = NULL,
  summary = NULL, optional = F, empty = F, lookupsAsFactors = T,
  sqlStatement = list(select = "SELECT *", groupBy = ""), forceElements = F)

## S4 method for signature 'list'
datasheet(ssimObject, name = NULL, project = NULL,
  scenario = NULL, summary = NULL, optional = F, empty = F,
  lookupsAsFactors = T, sqlStatement = list(select = "SELECT *", groupBy =
  ""), forceElements = F)

## S4 method for signature 'character'
datasheet(ssimObject, name = NULL, project = NULL,
  scenario = NULL, summary = NULL, optional = F, empty = F,
  lookupsAsFactors = T, sqlStatement = list(select = "SELECT *", groupBy =
  ""), forceElements = F)

## S4 method for signature 'SsimObject'
datasheet(ssimObject, name = NULL, project = NULL,
  scenario = NULL, summary = NULL, optional = F, empty = F,
  lookupsAsFactors = T, sqlStatement = list(select = "SELECT *", groupBy =
  ""), forceElements = F)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario, or list of objects. Note that all objects in a list must be of the same type, and belong to the same library.
name	Character or vector of these. Sheet name(s). If NULL, all datasheets in the ssimObject will be returned. Note that setting summary=F and name=NULL pulls all datasheets, which is timeconsuming and not generally recommended.
project	Character, numeric, or vector of these. One or more Project names, ids or objects. Note that integer ids are slightly faster.
scenario	Character, numeric, or vector of these. One or more Scenario names, ids or objects. Note that integer ids are slightly faster.
summary	Logical. If TRUE returns a dataframe of sheet names and other info. If FALSE returns dataframe or list of dataframes.
optional	Logical. If summary=TRUE and optional=TRUE returns only scope, name and displayName. If summary=FALSE and optional=TRUE returns all of the datasheet's columns, including the optional columns. If summary=TRUE, optional=FALSE, returns only those columns that are mandatory and contain data (if empty=F). Ignored if summary=F, empty=F and lookupsAsFactors=F.

`empty`	Logical. If TRUE returns empty dataframes for each datasheet. Ignored if summary=TRUE.
`lookupsAsFactors`	Logical. If TRUE (default) dependencies returned as factors with allowed values (levels). Set FALSE to speed calculations. Ignored if summary=TRUE.
`sqlStatement`	List returned by sqlStatement(). SELECT and GROUP BY SQL statements passed to SQLite database. Ignored if summary=TRUE.
`forceElements`	Logical. If FALSE and name has a single element returns a dataframe; otherwise a list of dataframes. Ignored if summary=TRUE.

Details

If summary=T or summary=NULL and name=NULL a dataframe describing the datasheets is returned: If optional=T columns include: scope, package, name, displayName, isSingle, isOutput, data. data only displayed for scenarios. dataInherited and dataSource columns added if a scenario has dependencies. If optional=F columns include: scope, name, displayName. All other arguments are ignored.

Otherwise, for each element in name a datasheet is returned as follows:

- If lookupsAsFactors=T (default): Each column is given the correct data type, and dependencies returned as factors with allowed values (levels). A warning is issued if the lookup has not yet been set.
- If empty=T: Each column is given the correct data type. Fast (1 less console command)
- If empty=F and lookupsAsFactors=F: Column types are not checked, and the optional argument is ignored. Fast (1 less console command).
- If ssimObject is a list of Scenario or Project objects (output from run(), scenario() or project()): Adds ScenarioID/ProjectID column if appropriate.
- If scenario/project is a vector: Adds ScenarioID/ProjectID column as necessary.
- If requested datasheet has scenario scope and contains info from more than one scenario: ScenarioID/ScenarioName/ScenarioParent columns identify the scenario by name, id, and parent (if a result scenario)
- If requested datasheet has project scope and contains info from more than one project: ProjectID/ProjectName columns identify the project by name and id.

Value

If summary=T returns a dataframe of datasheet names and other info, otherwise returns a dataframe or list of these.

`datasheetRaster` *Get spatial inputs or outputs from a Scenario(s).*

Description

Get spatial inputs or outputs from one or more SyncroSim scenarios.

Usage

```
datasheetRaster(ssimObject, datasheet, column = NULL, scenario = NULL,
  iteration = NULL, timestep = NULL, subset = NULL, forceElements = F)

## S4 method for signature 'character'
datasheetRaster(ssimObject, datasheet, column = NULL,
  scenario = NULL, iteration = NULL, timestep = NULL, subset = NULL,
  forceElements = F)

## S4 method for signature 'list'
datasheetRaster(ssimObject, datasheet, column = NULL,
  scenario = NULL, iteration = NULL, timestep = NULL, subset = NULL,
  forceElements = F)

## S4 method for signature 'SsimObject'
datasheetRaster(ssimObject, datasheet, column = NULL,
  scenario = NULL, iteration = NULL, timestep = NULL, subset = NULL,
  forceElements = F)

## S4 method for signature 'Scenario'
datasheetRaster(ssimObject, datasheet, column = NULL,
  scenario = NULL, iteration = NULL, timestep = NULL, subset = NULL,
  forceElements = F)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario or list of Scenarios. If SsimLibrary/Project, then scenario argument is required.
datasheet	character string. The name of the datasheet containing the raster data.
column	character string. The name of the column in the datasheet containing the filenames for raster data. If NULL then use the first column that contains raster filenames.
scenario	character string, integer, or vector of these. The scenarios to include. Required if ssimObject is an SsimLibrary/Project, ignored if ssimObject is a list of Scenarios.
iteration	integer, character string, or vector of integer/character strings. Iteration(s) to include. If NULL then all iterations are included. If no Iteration column in the datasheet, then ignored.
timestep	integer, character string, or vector of integer/character string. Timestep(s) to include. If NULL then all timesteps are included. If no Timestep column in the datasheet, then ignored.
subset	logical expression. logical expression indicating datasheet rows to return. e.g. expression(grepl("Ts0001",Filename,fixed=T)). See subset() for details.
forceElements	logical. If TRUE then returns a single raster as a RasterStack; otherwise returns a single raster as a RasterLayer directly.

Details

The names() of the returned raster stack contain metadata. For datasheets without Filename this is: paste0(<datasheet name>,".Scn",<scenario id>,".",<tif name>) For datasheets containing Filename this is: paste0(<datasheet name>,".Scn",<scenario id>,".It",<iteration>,".Ts",<timestep>)

Value

A RasterLayer, RasterStack or RasterBrick object. See raster package documentation for details.

Examples

```
## Not run:
datasheetRaster(myResult,datasheet="OutputSpatialState",
  subset=expression(grepl("Ts0001",Filename,fixed=T)))

## End(Not run)
```

---

dateModified *The last date a SsimLibrary/Project/Scenario was modified.*

Description

The most recent modification date of an SsimLibrary/Project/Scenario

Usage

```
dateModified(ssimObject)

## S4 method for signature 'character'
dateModified(ssimObject)

## S4 method for signature 'SsimLibrary'
dateModified(ssimObject)

## S4 method for signature 'Project'
dateModified(ssimObject)

## S4 method for signature 'Scenario'
dateModified(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.

---

delete *Delete library, project, scenario, datasheet*

Description

Deletes one or more items. Note this is irreversible

Usage

```
delete(ssimObject, project = NULL, scenario = NULL, datasheet = NULL,
  force = F)

## S4 method for signature 'character'
delete(ssimObject, project = NULL, scenario = NULL,
  datasheet = NULL, force = F)

## S4 method for signature 'SsimObject'
delete(ssimObject, project = NULL, scenario = NULL,
  datasheet = NULL, force = F)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario, or path to a library.
project	character string, numeric, or vector of these. One or more project names or ids. Note that project argument is ignored if ssimObject is a list. Note that integer ids are slightly faster.
scenario	character string, numeric, or vector of these. One or more scenario names or ids. Note that scenario argument is ignored if ssimObject is a list. Note that integer ids are slightly faster.
datasheet	character string or vector of these. One or more datasheet names.
force	logical. If FALSE (default), user will be prompted to approve removal of each item.

Value

A list of "saved" or failure messages for each item.

Examples

```
## Not run:
myLibrary = ssimLibrary("mylib", create=T)
myProject = project(myLibrary, project="a project", create=T)
project(myLibrary)
delete(myLibrary, project="a project")
project(myLibrary)

## End(Not run)
```

---

deleteBreakpoint *Delete a Scenario breakpoint.*

Description

This function will delete a Scenario breakpoint.

Usage

```
deleteBreakpoint(x, transformerName = NULL, breakpointType = NULL)

## S4 method for signature 'Scenario'
deleteBreakpoint(x, transformerName = NULL,
  breakpointType = NULL)
```

Arguments

x	A SyncroSim Scenario
transformerName	A Stochastic Time Transformer (e.g. stsim_Runtime). Optional.
breakpointType	bi: before iteration; ai: after iteration; bt:before timestep; at: after timestep. Optional.

Value

A SyncroSim Scenario with an updated list of breakpoints

Examples

```
## Not run:
myScenario = deleteBreakpoint(myScenario)
myScenario = deleteBreakpoint(myScenario, transformerName="stsim_Runtime")

## End(Not run)
```

---

deleteModule *Delete module or modules*

Description

Deprecated. See: deletePackage

Usage

```
deleteModule(name, session = NULL, force = F)

## S4 method for signature 'ANY,missingOrNULLOrChar'
deleteModule(name, session = NULL,
  force = F)

## S4 method for signature 'ANY,Session'
deleteModule(name, session = NULL, force = F)
```

Arguments

name	Character string or vector of these. A module or vector of modules to remove. See modules() for options.
session	Session.
force	logical. If T, delete without requiring confirmation from user.

Value

"saved" or error message.

`deletePackage` *Deletes a package*

Description

Deletes a package

Usage

```
deletePackage(name, session = NULL, force = F)

## S4 method for signature 'ANY,character'
deletePackage(name, session = NULL, force = F)

## S4 method for signature 'ANY,missingOrNULL'
deletePackage(name, session = NULL, force = F)

## S4 method for signature 'ANY,Session'
deletePackage(name, session = NULL, force = F)
```

Arguments

`name`	Character. The name of the package to delete.
`session`	Session.
`force`	logical. If T, delete without requiring confirmation from user.

Value

"saved" or error message.

`dependency` *Set or remove Scenario dependency(s), or get existing dependencies.*

Description

Set or remove Scenario dependency(s), or get existing dependencies.

Usage

```
dependency(scenario, dependency = NULL, remove = F, force = F)

## S4 method for signature 'character'
dependency(scenario, dependency = NULL, remove = F,
  force = F)

## S4 method for signature 'Scenario'
dependency(scenario, dependency = NULL, remove = F,
  force = F)
```

Arguments

scenario	Scenario. The scenario to which a dependency is to be added (or has already been added if remove=TRUE).
dependency	Scenario, character string, integer, or list/vector of these. The scenario(s) that are the source of the dependency, in order from lowest to highest precedence. If NULL other arguments are ingored and the list of existing dependencies is returned.
remove	logical. If F (default) dependencies are added. If T, dependencies are removed.
force	logical. If F (default) prompt before removing dependencies.

Details

If dependency==NULL, other arguments are ignored, and set of existing dependencies is returned in order of precedence (from highest to lowest precedence). Otherwise, returns list of saved or error messages for each dependency of each scenario.

Note that the order of dependencies can be important - dependencies added most recently take precedence over existing dependencies. So, dependencies included in the dependency argument take precedence over any other existing dependencies. If the dependency argument includes more than one element, elements are ordered from lowest to highest precedence.

Value

If dependency!=NULL, character string (saved or error message) or list of these. Otherwise, a dataframe of existing dependencies, or list of these.

---

description *Description of an SsimLibrary/Project/Scenario.*

Description

The description of an SsimLibrary/ProjectScenario.

Usage

```
description(ssimObject)

## S4 method for signature 'character'
description(ssimObject)

## S4 method for signature 'SsimObject'
description(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.

description<- *Set the description of an SsimLibrary/Project/Scenario.*

Description

Set the description of an SsimLibrary/ProjectScenario.

Usage

```
description(ssimObject) <- value

## S4 replacement method for signature 'character'
description(ssimObject) <- value

## S4 replacement method for signature 'SsimObject'
description(ssimObject) <- value
```

Arguments

ssimObject	Scenario/Project/SsimLibrary
value	The new description.

disableAddon *Disable addon or addons.*

Description

Disable addon or addons of an SsimLibrary, or Project/Scenario with an associated SsimLibrary.

Usage

```
disableAddon(ssimLibrary, name)

## S4 method for signature 'character'
disableAddon(ssimLibrary, name)

## S4 method for signature 'SsimLibrary'
disableAddon(ssimLibrary, name)
```

Arguments

ssimLibrary	SsimLibrary
name	Character string or vector of these.

Value

saved or error message.

Examples

```
## Not run:
myLibrary = ssimLibrary("mylib", create=T)
enableAddon(myLibrary,c("stsim-ecodep"))
addon(myLibrary)
disableAddon(myLibrary,c("stsim-ecodep"))
addon(myLibrary)

## End(Not run)
```

`enableAddon` *Enable addon or addons.*

Description

Enable addon or addons of an SsimLibrary.

Usage

```
enableAddon(ssimLibrary, name)

## S4 method for signature 'character'
enableAddon(ssimLibrary, name)

## S4 method for signature 'SsimLibrary'
enableAddon(ssimLibrary, name)
```

Arguments

`ssimLibrary`	SsimLibrary
`name`	Character string or vector of these.

Value

saved or error message for each addon.

Examples

```
## Not run:
myLibrary = ssimLibrary()
enableAddon(myLibrary,c("stsim-ecological-departure"))
addon(myLibrary)
disableAddon(myLibrary,c("stsim-ecological-departure"))
addon(myLibrary)

## End(Not run)
```

envInputFolder *SyncroSim DataSheet Input Folder*

Description

Creates and returns a SyncroSim DataSheet Input Folder

Usage

```
envInputFolder(scenario, datasheetName)
```

Arguments

scenario	Scenario. A SyncroSim result scenario.
datasheetName	character. The input datasheet name.

Value

a folder name for the specified data sheet

envOutputFolder *SyncroSim DataSheet Output Folder*

Description

Creates and returns a SyncroSim DataSheet Output Folder

Usage

```
envOutputFolder(scenario, datasheetName)
```

Arguments

scenario	Scenario. A SyncroSim result scenario.
datasheetName	character. The output datasheet name.

Value

a folder name for the specified data sheet

envReportProgress *Reports progress for a SyncroSim simulation*

Description

Reports progress for a SyncroSim simulation

Begins a SyncroSim simulation

Steps a SyncroSim simulation

Ends a SyncroSim simulation

Usage

```
envReportProgress(iteration, timestep)

envBeginSimulation(totalSteps)

envStepSimulation()

envEndSimulation()
```

Arguments

iteration	integer. The current iteration.
timestep	integer. The current timestep.
totalSteps	integer. The total number of steps in the simulation.

envTempFolder *SyncroSim Temporary Folder*

Description

Creates and returns a SyncroSim Temporary Folder

Usage

```
envTempFolder(folderName)
```

Arguments

folderName	character. The folder name

Value

a temporary folder name

filepath *The path to a SyncroSim object on disk*

Description

The path to a SyncroSim Session, SSimLibarary, Project or Scenario on disk.

Usage

```
filepath(ssimObject)

## S4 method for signature 'character'
filepath(ssimObject)

## S4 method for signature 'Session'
filepath(ssimObject)

## S4 method for signature 'SsimObject'
filepath(ssimObject)
```

Arguments

ssimObject	An object containing a filepath.

mergeDependencies *Merge Dependencies for a Scenario.*

Description

Whether or not a Scenario is configured to merge dependencies at run time.

Usage

```
mergeDependencies(ssimObject)

## S4 method for signature 'character'
mergeDependencies(ssimObject)

## S4 method for signature 'Scenario'
mergeDependencies(ssimObject)
```

Arguments

ssimObject	Scenario

Value

logical.

mergeDependencies<- *Merge Dependencies for a Scenario.*

Description

Whether or not a Scenario is configured to merge dependencies at run time.

Usage

```
mergeDependencies(ssimObject) <- value

## S4 replacement method for signature 'character'
mergeDependencies(ssimObject) <- value

## S4 replacement method for signature 'Scenario'
mergeDependencies(ssimObject) <- value
```

Arguments

ssimObject	Scenario
value	Logical. If T the Scenario will be set to merge dependencies at runtime.

model *Installed models*

Description

Deprecated. See: package

Usage

```
model(ssimObject = NULL)

## S4 method for signature 'character'
model(ssimObject = NULL)

## S4 method for signature 'missingOrNULL'
model(ssimObject = NULL)

## S4 method for signature 'Session'
model(ssimObject = NULL)

## S4 method for signature 'SsimLibrary'
model(ssimObject = NULL)
```

Arguments

ssimObject	Session or SsimLibrary.

Value

A dataframe of models (for Session) or named vector of character strings (for SsimLibrary)

---

module *Installed modules*

Description

Deprecated. See: package

Usage

```
module(session)

## S4 method for signature 'missingOrNULL'
module(session)

## S4 method for signature 'character'
module(session)

## S4 method for signature 'Session'
module(session)
```

Arguments

session	Session.

Value

A dataframe of modules

---

name *The name of a SyncroSim library, project or scenario.*

Description

The name of an SsimLibrary, Project or Scenario.

Usage

```
name(ssimObject)

## S4 method for signature 'character'
name(ssimObject)

## S4 method for signature 'SsimLibrary'
name(ssimObject)

## S4 method for signature 'Scenario'
```

```
name(ssimObject)

## S4 method for signature 'Project'
name(ssimObject)
```

Arguments

ssimObject	SsimLibrary, Project, or Scenario.

Value

character string

---

name<- *Set ssimObject name.*

Description

Set the name of a SyncroSim Project, Scenario or Library

Usage

```
name(ssimObject) <- value

## S4 replacement method for signature 'character'
name(ssimObject) <- value

## S4 replacement method for signature 'SsimLibrary'
name(ssimObject) <- value

## S4 replacement method for signature 'Project'
name(ssimObject) <- value

## S4 replacement method for signature 'Scenario'
name(ssimObject) <- value
```

Arguments

ssimObject	Scenario/Project/SsimLibrary
value	The new name.

owner *The owner of a SsimLibrary/Project/Scenario.*

Description

The owner of an SsimLibrary/ProjectScenario

Usage

```
owner(ssimObject)

## S4 method for signature 'character'
owner(ssimObject)

## S4 method for signature 'SsimLibrary'
owner(ssimObject)

## S4 method for signature 'Project'
owner(ssimObject)

## S4 method for signature 'Scenario'
owner(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.

owner<- *Set the owner of an SsimLibrary/Project/Scenario.*

Description

Set the owner of an SsimLibrary/Project/Scenario.

Usage

```
owner(ssimObject) <- value

## S4 replacement method for signature 'character'
owner(ssimObject) <- value

## S4 replacement method for signature 'SsimObject'
owner(ssimObject) <- value
```

Arguments

ssimObject	Scenario/Project/SsimLibrary
value	The new owner.

package *Installed or available packages*

Description

Packages or installed or available for this version of SyncroSim.

Usage

```
package(session, installed = T)

## S4 method for signature 'missingOrNULL'
package(session, installed = T)

## S4 method for signature 'character'
package(session, installed = T)

## S4 method for signature 'Session'
package(session, installed = T)
```

Arguments

session	Session.
installed	Logical. True to list installed packages and False to list available pacakges.

Value

A dataframe of packages

parentId *The parent scenario id of a SyncroSim Scenario.*

Description

The id of the parent of a SyncroSim results scenario. NA if scenario is not a results scenario.

Usage

```
parentId(scenario)

## S4 method for signature 'character'
parentId(scenario)

## S4 method for signature 'Scenario'
parentId(scenario)
```

Arguments

scenario	A Scenario object.

Value

An integer id of the parent scenario.

`printCmd` *Get printCmd of a Session.*

Description

Get printCmd setting of a Session object.

Usage

```
printCmd(session = NULL)

## S4 method for signature 'Session'
printCmd(session = NULL)

## S4 method for signature 'missingOrNULLOrChar'
printCmd(session = NULL)
```

Arguments

`session`	Session or character. A Session object or path to a session. If NULL, the default session will be used.

Value

logical.

`project` *Create or open a project or projects.*

Description

If summary = FALSE, returns one or more `Project` objects representing a SyncroSim projects. If summary = TRUE, returns project summary info.

Usage

```
project(ssimObject = NULL, project = NULL, sourceProject = NULL,
  create = F, summary = NULL, forceElements = F, overwrite = F)
```

Arguments

ssimObject	SsimLibrary/Scenario or character. An ssimObject containing a filepath to a library, or a filepath.
project	Character, integer, or vector of these. Names or ids of one or more projects. Note that integer ids are slightly faster.
sourceProject	Character, integer, or Project object. If not NULL, new projects will be copies of the sourceProject.
summary	Logical. If TRUE then return the project(s) in a dataframe with the projectId, name, description, owner, dateModified, readOnly. Default is TRUE if project=NULL and ssimObject is not Scenario/Project, FALSE otherwise.
forceElements	Logical. If TRUE then returns a single project as a named list; otherwise returns a single project as a Project object. Applies only when summary=FALSE.
overwrite	Logical. If TRUE an existing Project will be overwritten.

Details

For each element of project:

- If element identifies an existing project: Returns the existing Project
- If element identifies more than one project: Error
- If element does not identify an existing project: Creates a new Project named element. Note that SyncroSim automatically assign an id to a new project.

Value

A Project object representing a SyncroSim project, or a dataframe of project names and descriptions.

Examples

```
## Not run:
#Load a Library and create a new Project
myLibrary = ssimLibrary(name="stsim")
myProject = project(ssimLibrary=myLibrary, project="My new project name")

#Get a named list of existing Projects.
#Each element in the list is named by a character version of the Project ID.
myProjects = project(myLibrary,summary=F)
names(myProjects)   # vector of the project ids

#Get an existing Project.
myProject = myProjects[[1]]
myProject = project(myLibrary, project="My new project name")

#Get/set the project properties
name(myProject)
name(myProject) = "New project name"

## End(Not run)
```

Project-class *SyncroSim Project class*

Description

`Project` object representing a SyncroSim Project.

Slots

`session` The Session associated with the Project's Library.

`filepath` The path to the Project's Library on disk.

`datasheetNames` Names and scopes of datasheets in the Project's Library

`projectId` The Project id

See Also

See `project` for options when creating or loading a SyncroSim Project.

projectId *The projectId of a SyncroSim project or scenario.*

Description

The projectId of a SyncroSim Project or Scenario.

Usage

```
projectId(ssimObject)

## S4 method for signature 'character'
projectId(ssimObject)

## S4 method for signature 'Project'
projectId(ssimObject)

## S4 method for signature 'Scenario'
projectId(ssimObject)
```

Arguments

`ssimObject`	Project/Scenario.

Value

An integer project id.

readOnly *Read-only status of an SsimLibrary/Project/Scenario.*

Description

Whether or not an SsimLibrary/ProjectScenario is read-only.

Usage

```
readOnly(ssimObject)

## S4 method for signature 'character'
readOnly(ssimObject)

## S4 method for signature 'SsimLibrary'
readOnly(ssimObject)

## S4 method for signature 'Project'
readOnly(ssimObject)

## S4 method for signature 'Scenario'
readOnly(ssimObject)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.

Value

logical.

readOnly<- *Set the read/write status of an SsimLibrary/Project/Scenario.*

Description

Set the read-only status of an SsimLibrary/Project/Scenario. Applies to child objects if ssimObject is an SsimLibrary or Project.

Usage

```
readOnly(ssimObject) <- value

## S4 replacement method for signature 'character'
readOnly(ssimObject) <- value

## S4 replacement method for signature 'SsimObject'
readOnly(ssimObject) <- value
```

Arguments

ssimObject	Scenario/Project/SsimLibrary
value	Logical. If T the ssimObject will be read-only.

---

rsyncrosim *rsyncrosim: The R interface to SyncroSim: http://syncrosim.com/*

Description

rsyncrosim provides an interface to SyncroSim, a generalized framework for running and managing scenario-based stochastic simulations over space and time. Different kinds of simulation models can "plug-in" to SyncroSim as packages and take advantage of general features common to many kinds of simulation models, such as defining scenarios of inputs, running Monte Carlo simulations, and viewing charts and maps of outputs.

Details

To learn more about rsyncrosim, start with the vignette: TO DO

---

run *Run scenarios*

Description

Run one or more SyncroSim scenarios

Usage

```
run(ssimObject, scenario = NULL, summary = F, jobs = 1,
  forceElements = F)

## S4 method for signature 'character'
run(ssimObject, scenario = NULL, summary = F,
  jobs = 1, forceElements = F)

## S4 method for signature 'list'
run(ssimObject, scenario = NULL, summary = F, jobs = 1,
  forceElements = F)

## S4 method for signature 'SsimObject'
run(ssimObject, scenario = NULL, summary = F,
  jobs = 1, forceElements = F)

## S4 method for signature 'BreakpointSession'
run(ssimObject, scenario = NULL, summary = F,
  jobs = 1, forceElements = F)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario or a list of Scenarios. Or the path to a library on disk.
scenario	character, integer, or vector of these. Scenario names or ids. Or NULL. Note that integer ids are slightly faster.
summary	Logical. If FALSE (default) result Scenario objects are returned. If TRUE (faster) result scenario ids are returned.
jobs	Iteger. The number of jobs to run. Passed to SyncroSim where multithreading is handled.
forceElements	Logical. If TRUE then returns a single result scenario as a named list; otherwise returns a single result scenario as a Scenario object. Applies only when summary=FALSE.

Details

Note that breakpoints are ignored unless ssimObject is a single scenario.

Value

If summary=F a result Scenario object or a named list of result Scenarios. The name is the parent scenario for each result. If summary=T returns summary info for result scenarios.

---

runLog *The runLog of a result Scenario.*

Description

The runLog of a result Scenario

Usage

```
runLog(scenario)

## S4 method for signature 'character'
runLog(scenario)

## S4 method for signature 'Scenario'
runLog(scenario)
```

Arguments

scenario	A Scenario object.

Value

Character string of the run log.

saveDatasheet *Save datasheet(s)*

Description

Saves datasheets to a SsimLibrary/Project/Scenario.

Usage

```
saveDatasheet(ssimObject, data, name = NULL, fileData = NULL,
  append = NULL, forceElements = F, force = F, breakpoint = F,
  import = T, path = NULL)

## S4 method for signature 'character'
saveDatasheet(ssimObject, data, name = NULL,
  fileData = NULL, append = NULL, forceElements = F, force = F,
  breakpoint = F, import = T, path = NULL)

## S4 method for signature 'SsimObject'
saveDatasheet(ssimObject, data, name = NULL,
  fileData = NULL, append = NULL, forceElements = F, force = F,
  breakpoint = F, import = T, path = NULL)
```

Arguments

ssimObject	SsimLibrary/Project/Scenario.
data	A dataframe, named vector, or list of these. One or more datasheets to load.
name	character or vector of these. The name(s) of the datasheet(s) to be saved. If a vector of names is provided, then a list must be provided for the data argument. Names provided here will override those provided with data argument's list.
fileData	Named list or raster stack. Names are file names (without paths), corresponding to entries in data. The elements are objects containing the data associated with each name. Currently only supports Raster objects as elements.
append	logical. If TRUE, data will be appended to the datasheet if possible, otherwise current values will be overwritten by data. See details for behaviour when append=T. Default TRUE for project/library-scope datasheets, and FALSE for scenario-scope datasheets.
forceElements	logical. If FALSE (default) a single return message will be returns as a character string. Otherwise it will be returned in a list.
force	logical. If datasheet scope is project/library, and append=F, datasheet will be deleted before loading the new data. This can also delete other definitions and results, so user will be prompted for approval unless force=T.
breakpoint	Set to TRUE when modifying datasheets in a breakpoint function.
import	logical. Set to TRUE to import the data after saving.
path	character. An optional output path.

Details

Cautionary note re append=F: Deleting project and library level datasheets that contain lookups will also delete other definitions and results that rely on these lookups.

ssimObject/project/scenario should identify a single ssimObject.

If fileData !=NULL, each element of names(fileData) should correspond uniquely to at most one entry in data. If a name is not found in data the element will be ignored with a warning. If names(fileData) are full filepaths, rsyncrosim will write each object to the corresponding path for subsequent loading by SyncroSim. Note this is generally more time-consuming because the files must be written twice. If names(fileData) are not filepaths (faster, recommended), rsyncrosim will write each element directly to the appropriate SyncroSim input/output folders. rsyncrosim will write each element of fileData directly to the appropriate SyncroSim input/output folders. If fileData != NULL, data should be a dataframe, vector, or list of length 1, not a list of length >1.

There are 2 circumstances in which data will not be appended even if append=T:

- New data will not be appended if it is redundant with existing data, and the table does not allow redundancy.
- Old data will be replaced by new data if the datasheet allows only a single row.

Value

A success or failure message, or a list of these.

`scenario` *Create or open one or more Scenarios.*

Description

If summary = FALSE, returns one or more `Scenario` objects representing a SyncroSim scenarios. If summary = TRUE, returns scenario summary info.

Usage

```
scenario(ssimObject = NULL, scenario = NULL, sourceScenario = NULL,
  create = F, summary = NULL, results = F, forceElements = F,
  overwrite = F)
```

Arguments

`ssimObject`	SsimLibrary/Project or character. An ssimObject containing a filepath to a library, or a filepath.
`scenario`	Character, integer, or vector of these. Names or ids of one or more scenarios. Note integer ids are slightly faster.
`sourceScenario`	Character or integer. If not NULL, new scenarios will be copies of the sourceScenario.
`summary`	Logical. If TRUE then loads and returns the scenario(s) in a named vector/dataframe with the scenarioId, name, description, owner, dateModified, readOnly, parentID. Default is TRUE if scenario=NULL, FALSE otherwise.
`results`	Logical. If TRUE only return result scenarios.
`forceElements`	Logical. If TRUE then returns a single scenario as a named list; otherwise returns a single scenario as a Scenario object. Applies only when summary=FALSE.
`overwrite`	Logical. If TRUE an existing Scenario will be overwritten.

Details

For each element of scenario:

- If element/project/ssimObject uniquely identifies an existing scenario: Returns the existing Scenario
- If element/project/ssimObject uniquely identifies more than one existing scenario: Error
- If element/project/ssimObject do not identify an existing scenario or project: Error
- If element/project/ssimObject do not identify an existing scenario and element is numeric: Error - a name is required for new scenarios. SyncroSim will automatically assign an id when a scenario is created.
- If element/project/ssimObject do not identify an existing scenario and do identify a project, and element is a character string: Creates a new Scenario named element in the project. SyncroSim automatically assigns an id. If sourceScenario is not NULL the new scenario will be a copy of sourceScenario.

Value

A `Scenario` object representing a SyncroSim scenario, a list of Scenario objects, or a dataframe of scenario names and descriptions.

Examples

```
## Not run:
# Create a new scenario
myLibrary = ssimLibrary(name="stsim")
myProject = project(myLibrary,project="a project")
myScenario = scenario(myProject,scenario="a scenario",overwrite=T)

## End(Not run)
```

`Scenario-class` *SyncroSim Scenario class*

Description

`Scenario` object representing a SyncroSim Scnario.

Slots

`session` The Session associated with the Scenario.

`filepath` The path to the Scenario's Library on disk.

`datasheetNames` Names and scope of all datasheets in Scenario's Library.

`projectId` The project id.

`scenarioId` The scenario id.

`parentId` For a result scenario, this is the id of the parent scenario. 0 indicates this is not a result scenario.

`breakpoints` An (optional) list of Breakpoint objects.

See Also

See `scenario` for options when creating or loading a SyncroSim Scenario.

scenarioId *The scenarioId of a scenario.*

Description

The scenarioId of a Scenario.

Usage

```
scenarioId(scenario)

## S4 method for signature 'character'
scenarioId(scenario)

## S4 method for signature 'Scenario'
scenarioId(scenario)
```

Arguments

scenario	Scenario.

Value

integer id.

session *Creates or returns a SyncroSim session.*

Description

Methods to create or return a Syncrosim session.

Usage

```
session(x = NULL, silent = T, printCmd = F)

## S4 method for signature 'missingOrNULLOrChar'
session(x = NULL, silent = T,
  printCmd = F)

## S4 method for signature 'SsimObject'
session(x = NULL, silent = T, printCmd = F)
```

Arguments

x	Character or SsimObject. An optional path to the SyncroSim installation.
silent	Logical. Applies only if x is a path or NULL. If TRUE, warnings from the console are ignored. Otherwise they are printed.
printCmd	Logical. Applies only if x is a path or NULL. If TRUE, arguments passed to the SyncroSim console are also printed. Helpful for debugging. FALSE by default.

Value

A SyncroSim Session object.

Examples

```
## Not run:
#Create a library using a default Session and base package
myLib = ssimLibrary(name="mylib", create=T)

#Create a library using a non-default Session
mySession = session("C:/Downloads/SyncroSim")
myLib = ssimLibrary(name="mylib",session=mySession, create=T)

filepath(mySession)     # Lists the folder location of syncrosim session
version(mySession)      # Lists the version of syncrosim session
package(mySession)    # Dataframe of the packages installed with this version of syncrosim.
basePackage(mySession)  # Dataframe of the base packages installed with this version of syncrosim.

## End(Not run)
```

Session-class *SyncroSim Session class*

Description

A SyncroSim Session object contains a link to a SyncroSim installation. `SsimLibrary`, `Project` and `Scenario` objects contain a `Session` used to query and modify the object.

Slots

`filepath` The path to the SyncroSim installation.

`silent` If FALSE, all SyncroSim output with non-zero exit status is printed. Helpful for debugging. Default=TRUE.

`printCmd` If TRUE, arguments passed to the SyncroSim console are also printed. Helpful for debugging. Default=FALSE.

See Also

See `session` for options when creating a Session.

---

session<- *Set a SyncroSim session.*

Description

Set the Session of a SsimLibrary, Project or Scenario object.

Usage

```
session(ssimObject) <- value

## S4 replacement method for signature 'character'
session(ssimObject) <- value

## S4 replacement method for signature 'SsimObject'
session(ssimObject) <- value
```

Arguments

ssimObject	SsimObject/Project/Scenario.
value	A SyncroSim Session.

Details

In order to avoid problems with SyncroSim version compatibility and library updating, the new session must have the same filepath as the session of the SsimObject e.g. filepath(value)==filepath(session(ssimObject))

Value

An SyncroSim object containing a Session.

---

silent *Check if a Session is silent*

Description

Checks whether a SyncroSim Session is silent or not.

Usage

```
silent(session)

## S4 method for signature 'Session'
silent(session)

## S4 method for signature 'missingOrNULLOrChar'
silent(session)
```

Arguments

session	Session or character. A SyncroSim Session object or path to a session. If NULL, the default session will be used.

Value

logical.

silent<- *Set silent property of a Session*

Description

Set silent property of a sessio to TRUE or FALSE

Usage

```
silent(session) <- value

## S4 replacement method for signature 'character'
silent(session) <- value

## S4 replacement method for signature 'Session'
silent(session) <- value
```

Arguments

session	Session
value	logical

sqlStatement *Construct an SQLite query*

Description

Creates SELECT, GROUP BY and WHERE SQL statements. The resulting list of SQL statements will be converted to an SQLite database query by the datasheet() function.

Usage

```
sqlStatement(groupBy = NULL, aggregate = NULL, aggregateFunction = "SUM",
  where = NULL)
```

Arguments

groupBy	character string or vector of these. Vector of variables (column names) to GROUP BY.
aggregate	character string of vector of these. Vector of variables (column names) to aggregate using aggregateFunction
aggregateFunction	character string. An SQL aggregate function (e.g. SUM, COUNT)
where	named list. A list of subset variables. Names are column names, and elements are the values to be selected from each column.

Details

Variables are column names of the datasheet. See column names using datasheet(,empty=T). Variables not included in groupBy, aggregate or where will be dropped from the table. Note that it is not possible to construct a complete SQL query at this stage, because the datasheet() function may add ScenarioID and/or ProjectID to the query.

Value

A list of SELECT, GROUP BY and WHERE SQL statements used by datasheet() to construct an SQLite database query.

Examples

```
#Query the total Amount for each combination of ScenarioID, Iteration, Timestep and StateLabelXID,
#including only Timesteps 0,1 and 2, and Iterations 3 and 4.
mySQL = sqlStatement(groupBy=c("ScenarioID","Iteration","Timestep","StateLabelXID"),
  aggregate=c("Amount"),where=list(Timestep=c(0,1,2),Iteration=c(3,4)))
mySQL
```

---

ssimEnvironment *SyncroSim Environment.*

Description

Retrieves SyncroSim specific environment variables.

Usage

```
ssimEnvironment()
```

Value

a data.frame of SyncroSim specific environment variables.

---

ssimLibrary *Create or open a library.*

Description

Creates or opens an SsimLibrary object. If summary = T, returns library summary info. If summary = NULL, returns library summary info if ssimObject is an SsimLibrary, SsimLibrary object otherwise.

Usage

```
ssimLibrary(name = NULL, create = F, summary = NULL, package = NULL,
  session = NULL, addon = NULL, forceUpdate = F, overwrite = F)

## S4 method for signature 'SsimObject'
ssimLibrary(name = NULL, create = F,
  summary = NULL, package = NULL, session = NULL, addon = NULL,
  forceUpdate = F, overwrite = F)

## S4 method for signature 'missingOrNULLOrChar'
ssimLibrary(name = NULL, create = F,
  summary = NULL, package = NULL, session = NULL, addon = NULL,
  forceUpdate = F, overwrite = F)
```

Arguments

name	Character string, Project/Scenario/SsimLibrary. The path to a library or SsimObject.
summary	logical. Default T
package	Character. The package type. The default is "stsim".
session	Session. If NULL, session() will be used.
addon	Character or character vector. One or more addons. See addon() for options.
forceUpdate	Logical. If FALSE (default) user will be prompted to approve any required updates. If TRUE, required updates will be applied silently.
overwrite	Logical. If TRUE an existing Library will be overwritten.

Details

- If name is SyncroSim Project or Scenario: Returns the SsimLibrary associated with the Project or Scenario.
- If name is NULL: Create/open a SsimLibrary in the current working directory with the filename SsimLibrary.ssim.
- If name is a string: If string is not a valid path treat as filename in working directory. If no file suffix provided in string then add .ssim. Attempts to open a library of that name. If library does not exist creates a library of type package in the current working directory.
- If given a name and a package: Create/open a library called <name>.ssim. Returns an error if the library already exists but is a different type of package.

Value

An SsimLibrary object.

Examples

```
## Not run:
#Create a library using the default session
myLibrary = ssimLibrary(name="myLib", overwrite=T)

#Open a library using the default session
myLibrary = ssimLibrary(name="myLib")
```

```
#Create library using a specific session
mySession = session("C:/Downloads/SyncroSim")
myLibrary = ssimLibrary(name="myLib",session=mySession, session=T)

session(myLibrary)
filepath(myLibrary)
info(myLibrary)

## End(Not run)
```

`SsimLibrary-class` *SyncroSim Library class*

Description

`SsimLibrary` object representing a SyncroSim Library.

Slots

`session` The SyncroSim Session.

`filepath` The path to the Library on disk.

`datasheetNames` The name and scope of all datasheets in the Library.

See Also

See `ssimLibrary` for options when creating or loading a SyncroSim Library.

`ssimUpdate` *Apply updates.*

Description

Apply updates to a SyncroSim Library,or a Project or Scenario associated with a Library.

Usage

```
ssimUpdate(ssimObject)

## S4 method for signature 'character'
ssimUpdate(ssimObject)

## S4 method for signature 'SsimObject'
ssimUpdate(ssimObject)
```

Arguments

`ssimObject`	SsimLibrary/Project/Scenario

Value

"saved" or a failure message from the console.

tempfilepath *The temporary file path to a SyncroSim object on disk*

Description

The temporary file path to a SyncroSim Session, SSimLibarary, Project or Scenario on disk.

Usage

```
tempfilepath(ssimObject)

## S4 method for signature 'character'
tempfilepath(ssimObject)

## S4 method for signature 'Session'
tempfilepath(ssimObject)

## S4 method for signature 'SsimObject'
tempfilepath(ssimObject)
```

Arguments

ssimObject	An object containing a filepath.

updatePackage *Update Package*

Description

Updates a SyncroSim package

Usage

```
updatePackage(name = NULL, session = NULL, listonly = F)

## S4 method for signature 'ANY,character'
updatePackage(name = NULL, session = NULL,
  listonly = F)

## S4 method for signature 'ANY,missingOrNULL'
updatePackage(name = NULL, session = NULL,
  listonly = F)

## S4 method for signature 'ANY,Session'
updatePackage(name = NULL, session = NULL,
  listonly = F)
```

Arguments

name	Character string. The name of the package to update. If NULL, all packages will be updated.
session	Session.
listonly	Logical. If TRUE, available updates are listed only.

---

version *The SyncroSim version*

Description

The version of a SyncroSim Session

Usage

```
version(session = NULL)

## S4 method for signature 'character'
version(session = NULL)

## S4 method for signature 'missingOrNULL'
version(session = NULL)

## S4 method for signature 'Session'
version(session = NULL)
```

Arguments

session	Session.

Index