Package 'rsyncrosim'

March 15, 2017

Type Package

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

Version 0.1.0

Author ApexRMS

Maintainer Josie Hughes <josie.s.hughes@gmail.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 modules and take advantage of general features common to many kinds of simulation models, such as defining scenarios of model inputs, running Monte Carlo simulations, and viewing charts and maps of outputs.

License ?

LazyData TRUE

Imports DBI,
RSQLite,
raster,
parallel,
Rcpp,
rgdal

Suggests plyr,
ggplot2,
rasterVis,
knitr,
rmarkdown,
testthat

Collate 'addRows.R'
'generics.R'
'session.R'
'ssimLibrary.R'
'project.R'
'scenario.R'
'breakpointSession.R'
'breakpoint.R'
'command.R'
'dataframeFromSSim.R'
'fullFilename.R'
'getFromXProjScn.R'

'internalHelpers.R'
'sqlStatements.R'
'internalWrappers.R'
'rasterAttributes.R'
'rsyncrosim.R'

RoxygenNote 5.0.1

VignetteBuilder knitr

R topics documented:

addModules<- *Add modules*

addons *addons of an SSimLibrary*

addModules<-

Add modules

Description

Add module or modules to this version of SyncroSim

Usage

```
addModules(x) <- value
```

Arguments

x	A SyncroSim Session object.
value	The path to an .ssimpkg file on disk, or a vector of filepaths

addons

addons of an SSimLibrary

Description

The addons of an SSimLibrary.

Usage

```
addons(x, all = F)
```

Arguments

x	An SSimLibrary, or a Project/Scenario object associated with an SSimLibrary.
all	If T, all available addons are returned. Otherwise, only enabled addons.

Value

A dataframe of addons.

Examples

```
addons(ssimLibrary(model="stsim",name="stsim"))
```

addRows<- *Add rows to a datasheet.*

Description

Adds rows to a dataframe. Preserves the types and factor levels of x. Fills missing values if possible using factor levels.

Usage

```
addRows(x) <- value
```

Arguments

x	A dataframe.
value	A dataframe. Columns in value should be a subset of columns in x.

Value

A dataframe with new rows.

author *The author of a Scenario*

Description

The author of a Scenario

Usage

```
author(x)
```

Arguments

x	An Scenario object.

Value

The author name.

Breakpoint-class *Breakpoint class*

Description

Breakpoint class

Slots

`arguments` Timesteps or iterations e.g. "1,2"

`breakpointName` Breakpoint name

`name` Name

`transformerName` 'stsim:core-transformer' or?

`callback` The function to apply. See STSimBreakpointsTutorial.R for details.

breakpoints *The breakpoints of a Scenario*

Description

The breakpoints of a Scenario

Usage

```
breakpoints(x)
```

Arguments

`x`	A Scenario object.

Value

A list of Breakpoint objects.

BreakpointSession-class *BreakpointSession class*

Description

BreakpointSession class

Slots

`scenario` A SyncroSim scenario

`connection` A socket connection to the SyncroSim server.

`name` A name

command *SyncroSim console command*

Description

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

Usage

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

Arguments

args	A list of arguments to the SyncroSim console.
session	A SyncroSim session object. If NULL, a default session will be used.
printCmd	If T, the command string is printed.
silent	If NULL (default) use session@silent. If T suppress warnings from console.
wait	If TRUE (default) R will wait for the command to finish before proceeding.

Value

Output from the SyncroSim console.

Examples

```
# Use a default session to creat a new library
args = list(create=NULL,ssimLibrary=NULL,name=paste0(getwd(),"/temp.ssim",model="stsim:model-transformer")
output = command(args)
output
```

connection *Get or set a socket connection.*

Description

Get or set a socket connection.

Usage

```
connection(x, ...)

## S4 method for signature 'missingOrNULLOrChar'
connection(x = "127.0.0.1", port = 13000)

## S4 method for signature 'BreakpointSession'
connection(x)
```

Arguments

x	An ipAddress or BreakpointSession object. If NULL a default ip will be used.
port	For new connections only - a port number.

Methods (by class)

- missingOrNULLOrChar: Get a new connection.
- BreakpointSession: Get the connection of a BreakpointSession.

---

datasheet *Get a datasheet*

Description

Gets Syncrosim datasheet.

Usage

```
datasheet(x, name, project = NULL, scenario = NULL, optional = F,
  empty = F, lookupsAsFactors = T, sqlStatements = list(select =
  "SELECT *", groupBy = ""), includeKey = F, printCmd = F)
```

Arguments

x	An SSimLibrary, Project or Scenario object. Or the path to a library on disk. Or a list of Scenario or Project objects.
name	The sheet name
project	One or more Project names, id or objects.
scenario	One or more Scenario names, id or objects.
optional	If FALSE (default) returns only required columns. If TRUE returns optional columns also. Ignored if empty=F and lookupsAsFactors=F.
empty	If FALSE (default) returns data (if any). If TRUE returns empty dataframe.
lookupsAsFactors	If TRUE (default) dependencies returned as factors with allowed values (levels). Set FALSE to speed calculations.
sqlStatements	SELECT and GROUP BY SQL statements passed to SQLite database.
includeKey	If TRUE include primary key in output table.
printCmd	Set to TRUE to see SyncrSim command line arguments. Helpful for debugging.

Details

- 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 x is a list of Scenario or Project objects (output from run(), scenarios() or projects()): Adds ScenarioID/ProjectID column if appropriate.
- If length(scenario)>1: Adds ScenarioID/ProjectID column if appropriate.
- 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

A dataframe representing a SyncroSim datasheet.

`datasheets` *datasheets*

Description

Gets datasheets from an SSimLibrary, Project or Scenario.

Usage

```
datasheets(x, project = NULL, scenario = NULL, names = T, scope = NULL,
  optional = F, empty = F, lookupsAsFactors = T, refresh = F)
```

Arguments

`x`	An SSimLibrary, Project or Scenario object. Or a path to a SyncroSim library on disk.
`project`	Project name or id. Ignored if x is a Project.
`scenario`	Scenario name or id. Ignored if x is a Scenario.
`names`	If TRUE (default) returns dataframe of sheet names, ignoring remaining arguments. If FALSE returns a named list of dataframes representing each datasheet.
`scope`	"scenario","project", "library", "all", or NULL.
`optional`	If FALSE (default) returns only required columns. If TRUE returns optional columns also. Ignored if empty=F and lookupsAsFactors=F.
`empty`	If FALSE (default) returns data (if any). If TRUE returns empty dataframe.
`lookupsAsFactors`	If TRUE (default) lookups are returned as factors with allowed values (levels). Set FALSE to speed calculations.
`refresh`	If FALSE (default) names are retrieved from x@datasheetNames. If TRUE names are retrieved using a console call (slower).

Details

See `datasheet` for discussion of optional/empty/sheetName/lookupsAsFactors arguments.

- If x/project/scenario identify a scenario: Returns library, project, and scenario scope datasheets.
- If x/project/scenario identify a project (but not a scenario): Returns library and project scope datasheets.
- If x/project/scenario identify a library (but not a project or scenario): Returns library scope datasheets.

Value

A dataframe of datasheet names, or list of datasheets represented by dataframes.

`definitions` *definitions*

Description

Alias for `datasheets` function

Usage

```
datasheets(x, project = NULL, scenario = NULL, names = T, scope = NULL,
  optional = F, empty = F, lookupsAsFactors = T, refresh = F)
```

`deleteProjects` *Delete projects from a Library*

Description

Deletes one or more projects from a SyncroSim library.

Usage

```
deleteProjects(x, project = NULL, force = F)
```

Arguments

`x`	An SSimLibrary, Project or Scenario associated with a library.
`project`	One or more project names or ids.

Value

A list of "Success!" or failure messages for each project.

Examples

```
myLibrary = ssimLibrary(model="stsim",session=devSession)
myProject = project(myLibrary)
projects(myLibrary,names=T)
deleteProjects(myLibrary,project="Project1")
projects(myLibrary,names=T)
```

deleteScenarios *Delete scenarios*

Description

Deletes one or more scenarios from a SyncroSim library.

Usage

```
deleteScenarios(x, scenario = NULL, force = FALSE)
```

Arguments

x	An SSimLibrary, Project or Scenario.
scenario	One or more scenario names or ids.
force	If FALSE (default) user is prompted to confirm deletions.

Value

A list of "Success!" or failure messages for each scenario.

Examples

```
myLibrary = ssimLibrary(model="stsim")
myScenario = scenario(project(myLibrary))
scenarios(myLibrary,names=T)
deleteScenarios(myLibrary,scenario="Scenario")
scenarios(myLibrary,names=T)
```

description *The description of a Scenario*

Description

The description of a Scenario

Usage

```
description(x)
```

Arguments

x	An Scenario object.

Value

The description.

disableAddons<- *Disable addons.*

Description

Disable addons an SSimLibrary, or Project/Scenario with an associated SSimLibrary.

Usage

```
disableAddons(x) <- value
```

Arguments

x=	A SSimLibrary, Project or Scenario.

Value

x

Examples

```
myLibrary = ssimLibrary()
enableAddons(myLibrary)=c("stsim-ecological-departure")
addons(myLibrary)
disableAddons(myLibrary)=c("stsim-ecological-departure")
addons(myLibrary)
```

enableAddons<- *Enable addons.*

Description

Enable addons of an SSimLibrary, or Project/Scenario with an associated SSimLibrary.

Usage

```
enableAddons(x) <- value
```

Arguments

x=	A SSimLibrary, Project or Scenario.

Value

x

Examples

```
myLibrary = ssimLibrary()
enableAddons(myLibrary)=c("stsim-ecological-departure", "stsim-stock-flow")
addons(myLibrary)
```

filepath *The path to a SyncroSim object on disk*

Description

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

Usage

```
filepath(x)
```

Arguments

x	An object containing a filepath.

hello *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

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

Description

The id of a SyncroSim Project or Scenario.

Usage

```
id(x)
```

Arguments

x	An object with an id.

info *Information about an object*

Description

Get basic information about a SyncroSim Session, SSimLibarary, Project or Scenario

Usage

```
info(x)
```

Arguments

x	An object containing info.

loadDatasheets *Set datasheets*

Description

Loads datasheets into the SyncroSim library.

Usage

```
loadDatasheets(x, data, name, project = NULL, scenario = NULL,
  breakpoint = F, printCmd = F)
```

Arguments

x	An SSimLibrary, Project or Scenario object. Or the path to a library on disk.
data	A dataframe or named list of dataframes to load.
name	The sheet name - required if data is a dataframe, ignored otherwise.
project	Project name or id.
scenario	Scenario name or id.
breakpoint	Set to TRUE when modifying datasheets in a breakpoint function.
printCmd	Set to TRUE to see the SyncroSim command line arguments. Helpful for debugging.

Value

A named list of success or failure reports.

loadSpatialData *Set spatial data*

Description

Loads spatial data into the SyncroSim library.

Usage

```
loadSpatialData(x, data, metadata = NULL, project = NULL, scenario = NULL,
  breakpoint = F, check = T)
```

Arguments

x	An SSimLibrary, Project or Scenario object. Or the path to a library on disk.
data	A RasterLayer or RasterStack to load.
metadata	A dataframe that can be appended to datasheet(x,metadata$SheetName[1]), containing 1 row for each layer - see details. If NULL, use names(data) metadata.
project	Project name or id.
scenario	Scenario name or id.
breakpoint	Set to TRUE when setting spatial data in a breakpoint function.
check	Default is TRUE. Set FALSE to speed calculations my assuming metadata is valid.

Details

If metadata=NULL or sheetName=NULL assume each raster layer has names() metadata - see spatialData() for details.

Otherwise, metadata should be a dataframe that can be appended to datasheet(x,metadata$SheetName[1]), containing 1 row for each layer of data. "SheetName" and "RasterLayerName" columns are expected in metadata, and will be removed before appending to the datasheet.

INCOMPLETE: this method has only been implemented for breakpoint=T and sheet name is STSim_TransitionSpatialMu

Value

A named list of success or failure reports.

modelName *The name of the primary model associate with a SyncroSim object*

Description

The name of the primary model associated with a SSimLibarary, Project or Scenario.

Usage

```
modelName(x)
```

Arguments

x	An object with an associated primary model.

Value

A model name

---

models *Installed models*

Description

Models installed with this version of SyncroSim

Usage

```
models(x)
```

Arguments

x	A SyncroSim Session object.

---

modelVersion *The version of the primary model associated with a SyncroSim object*

Description

The version of the primary model associated with a SSimLibarary, Project or Scenario.

Usage

```
modelVersion(x)
```

Arguments

x	An object with an associated primary model.

Value

A model version.

modules *Installed modules*

Description

Modules installed with this version of SyncroSim

Usage

```
modules(x)
```

Arguments

x	A SyncroSim Session object.

multiband *Modify the grouping of spatial layers.*

Description

Modify the grouping of spatial output layers in a SyncroSim results scenario.

Usage

```
multiband(x, action, grouping = NULL)
```

Arguments

x	A SyncroSim results Scenario or list of SyncroSim result Scenarios.
action	Options are: apply, remove, rebuild
grouping	Only used if action=apply. If NULL use datasheet(myLibrary,name="STime_Options"). Options are: Iteration,Timestep,All

Value

"Success!" or an error message from SyncroSim.

Examples

```
# Update an old scenario to allow rsyncrosim to access spatial output
multiband(myResultScenario,action="rebuild")

# Combine spatial outputs into multi-band rasters containing a layer for each timetep.
multiband(myResultScenario,action="apply",grouping="Timestep")

# Combine spatial outputs into multi-band rasters containing a layer for each iteration.
multiband(myResultScenario,action="apply",grouping="Iteration")

# Combine spatial outputs into multi-band rasters containing a layer for each timestep and iteration.
multiband(myResultScenario,action="apply",grouping="All")
```

```
# Remove multi-banding
multiband(myResultScenario,action="remove")
```

`name` *The name of a SyncroSim project or scenario.*

Description

The name of a SyncroSim Project or Scenario.

Usage

```
name(x)
```

Arguments

`x`	An object with a name.

`name<-` *Set the project or scenario names.*

Description

Set the name of a SyncroSim Project or Scenario.

Usage

```
name(x) <- value
```

Arguments

`x`	A SyncroSim `Project` or `Scenario` object.
`value`	The new name.

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

Description

The id of the parent of a SyncroSim results scenario. 0 if x is not a results scenario.

Usage

```
parentId(x)
```

Arguments

`x`	A Scenario object.

pid *The pid of a SyncroSim Scenario.*

Description

The project id of a SyncroSim Scenario

Usage

```
pid(x)
```

Arguments

x	An Scenario object.

project *Create or open a project.*

Description

Creates or opens an Project object representing a SyncroSim project.

Usage

```
project(ssimLibrary, name = NULL, id = NULL, create = T,
  projects = NULL)
```

Arguments

ssimLibrary	An SSimLibrary object, representing the library that contains the project.
name	The project name.
id	The project id.
create	If TRUE, create project if one does not exist. If FALSE, only return an existing project
projects	A dataframe of existing projects produced by projects(). Use to speed processing.

Details

- If name/id uniquely identify an existing project: Returns the existing Project
- If name/id identify more than one project: Error
- If name/id don't identify an existing project, and name is not specified: Creates a new Project called "Project". The id argument is ignored, as SyncroSim automatically assigns an id.
- If name/id don't identify an existing project, and name is specified: Creates a new Project called <name>. The id argument is ignored, as SyncroSim automatically assigns an id.

Value

A `Project` object representing a SyncroSim project.

Examples

```
# Create a new project
myLibrary = ssimLibrary(model="stsim",name="stsim")
myProject = project(myLibrary) #If no name is given, creates a project named "Project<ID>".
myProject = project(ssimLibrary=mySsimLibrary, name="My new project name")

# Get a named list of existing projects
myProjects = projects(myLibrary) # Each element in the list is named by a character version of the project ID
names(myProjects)   # vector of the project names (using base R names function)
#TO DO: base R function names returns project id's, not names. Do we want to overwrite the base function?

# Get an existing project. Assume that name uniquely identifies a single project - give error if not
myProject = myProjects[[1]]
myProject = project(myLibrary, name="My new project name")

# Get/set the project properties - for now we can only set the name
name(myProject)
name(myProject) = "New project name" #  - committed to db immediately
ssimLibrary(myProject) # Returns a SyncroSimLibrary object for the project
```

Project-class *SyncroSim Project class*

Description

`Project` object representing a SyncroSim Project.

Slots

`session` The session associated with the library.

`filepath` The path to the library on disk.

`datasheetNames` Names and scopes of datasheets in the library.

`name` The project name

`id` The project id

See Also

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

projectId — *The project id of a SyncroSim Scenario.*

Description

The project id of a SyncroSim Scenario

Usage

```
pid(x)
```

Arguments

x	An Scenario object.

projects — *The projects in a SyncroSim library.*

Description

Get a list of projects in a SyncroSim library.

Usage

```
projects(x, names = F)
```

Arguments

x	An SSimLibrary object, or a Project or Scenario associated with a Library
names	If FALSE, a list of Project objects is returned. If TRUE returns a dataframe containing the name and id of each project.

Value

By default returns a list of projects identified by the project id. Each element of the list contains a SyncroSim Project object. If names=T, returns a dataframe containing the name and id of each project.

Examples

```
myProjects = projects(ssimLibrary(model="stsim",name="stsim"))
```

rasterAttributes<- *Set attributes and colors of a RasterLayer object.*

Description

Set attributes and colors of a Raster object. This is a wrapper around ratify() and colortable() functions from the raster package.

Usage

```
rasterAttributes(x) <- value
```

Arguments

x	A Raster object.
rat	A raster attribute table. This is a dataframe with ID, (optional) Color, and other columns. See raster::ratify() for details.

Details

The (optional) Color column of a rat table should have one of these formats:

- R,G,B,alpha: 4 numbers representing red, green, blue and alpha, separated by commas, and scaled between 0 and 255. See rgb() for details.
- R colour names: See colors() for options.
- hexadecimal colors: As returned by R functions such as rainbow(), heat.colors(), terrain.colors(), topo.colors(), gray(), etc.

Examples

```
levels(myRaster) #retrieve raster attribute table
colortable(myRaster) #retrieve colortable
```

readOnly *The write status of a Scenario*

Description

Whether or not the scenario is readOnly

Usage

```
readOnly(x)
```

Arguments

x	An Scenario object.

Value

TRUE or FALSE

`removeModules<-` *Remove modules*

Description

Remove module or modules to this version of SyncroSim. Note that removing a module can be difficult to undo. To restore the module the user will need to provide a .ssimpkg file or reinstall SyncroSim. Thus, `removeModules` requires confirmation from the user.

Usage

```
removeModules(x) <- value
```

Arguments

`x`	A SyncroSim `Session` object.
`value`	A module or vector of modules to remove. `modules()` for options.

`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 modules and take advantage of general features common to many kinds of simulation models, such as defining scenarios of model 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(x, scenario = NULL, onlyIds = F, jobs = 1)
```

Arguments

x	One or more SSimLibrary, Projects or Scenario objects. Or the path to a library on disk.
scenario	One or more scenario objects, names or ids.
onlyIds	If FALSE (default) result Scenario objects are returned. If TRUE (faster) result scenario ids are returned.
jobs	The number of jobs to run. Passed to SyncroSim where multithreading is handled.

Value

A named list of result Scenario objects or ids. The name is the parent scenario for each result.

scenario *Create or open a scenario*

Description

Creates or opens a Scenario object representing a SyncroSim scenario.

Usage

```
scenario(ssimLibrary = NULL, project = NULL, name = NULL, id = NULL,
  create = T, scenarios = NULL, sourceScenario = NULL, author = NULL,
  description = NULL, readOnly = NULL)
```

Arguments

ssimLibrary	An SSimLibrary object or name, or an object that contains an SSimLibrary. If a name is given, the library will be opened using the default session.
project	A Project object, project name, or project id.
name	The scenario name.
id	The scenario id.
create	If TRUE, create scenario if one does not exist. If FALSE, only return an existing scenario

scenarios	A dataframe of existing scenarios produced by scenarios(). Use to speed processing.
sourceScenario	The name or id of a scenario to copy.
author	Optional.
description	Optional.
readOnly	By default scenarios are not readOnly.

Details

- If name/id/project uniquely identifies an existing scenario: Returns the existing Scenario
- If name/id/project uniquely identifies more than one existing scenario: Error
- If project is NULL, and name/id do not uniquely idenfity an existing scenario: Error
- If project is not NULL, name is NULL, and id/project do not idenfity an existing scenario: Creates a new Scenario called "Scenario". The id argument is ignored, as SyncroSim automatically assigns an id. If sourceScenario is not NULL the new scenario will be a copy of sourceScenario.
- If project is not NULL, name is not NULL, and name/id/project do not idenfity an existing scenario: Creates a new Scenario called <name>. The id argument is ignored, as 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.

Examples

```
# Create a new default scenario
myLibrary = ssimLibrary(model="stsim",name="stsim")
myProject = project(myLibrary) #If no name is given, creates a project named "Project".
myScenario = scenario(myProject)
```

Scenario-class *SyncroSim Scenario class*

Description

`Scenario` object representing a SyncroSim Project.

Slots

`session` The session associated with the library.

`filepath` The path to the library on disk.

`datasheetNames` Names and scope of all datasheets in library.

`pid` The project id.

`name` The scenario name.

`id` 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 ?breakpoints for details.

See Also

See scenario for options when creating or loading an SyncroSim Scenario.

---

scenarios *The scenarios in a SyncroSim library or project.*

Description

Get a list of scenarios in a SSimLibrary or Project.

Usage

```
scenarios(x, project = NULL, names = F, results = NULL, select = NULL)
```

Arguments

x	An SSimLibrary or Project object, or an SSimLibrary name.
project	An optional project name, id, or object.
names	If FALSE, a list of Scenario objects is returned. If TRUE returns a dataframe containing the name,id and project id of each scenario.
results	If TRUE only return result scenarios.
select	An (optional) vector of scenario ids or names to include

Examples

```
myScenarios = scenarios(ssimLibrary(model="stsim",name="stsim"))
```

---

session *Start or get a SyncroSim session.*

Description

Methods to create a Syncrosim session or fetch one from a SSimLibrary, Project or Scenario object.

Usage

```
session(x = NULL, ...)

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

## S4 method for signature 'SSimLibrary'
session(x)
```

Arguments

x	A path to SyncroSim.Console.exe or an object containing a Session. If NULL the usual locations are searched.
silent	Applies only if x is a path or NULL. If TRUE, warnings from the console are ignored. Otherwise they are printed.

Value

An SyncroSim Session object containing a valid console path.

Methods (by class)

- `missingOrNULLOrChar`: Create a SyncroSim Session from a filepath or get default Session.
- `SSimLibrary`: Get the Session associated with a SSimLibrary.

Examples

```
# Look for SyncroSim in the usual places
mySession = session()
path(mySession)

# Specify a SyncroSim version
mySession = session("C:/Program Files/SyncroSim/1/SyncroSim.Console.exe")

# Get the session from an SSimLibrary
myLib = ssimLibrary(name="stsim",model="stsim")
session(myLib)

# Assign a session to a SyncroSim library
session(myLib)=session()
```

`Session-class` *SyncroSim Session class*

Description

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

Slots

`filepath` The path to SyncroSim

`silent` If TRUE (default), warnings from the console are ignored. Otherwise they are printed.

Examples

```
# Create or load a library using a non-default Session
mySession = session("C:/Program Files/SyncroSim/1/SyncroSim.Console.exe")
myLib = ssimLibrary(name="stsim",model="st-sim",session=mySession)
session(myLib)

showMethods(class="Session",where=loadNamespace("rsyncrosim")) #Methods for the Session
filepath(mySession)   # Lists the folder location of syncrosim session
version(mySession)   # Lists the version of syncrosim session
modules(mySession)   # Dataframe of the modules installed with this version of syncrosim.
models(mySsim) # Dataframe of the models installed with this version of syncrosim.

# Add and remove modules
removeModules(mySsim) = "stsim-stock-flow"
is.element("stsim-stock-flow",modules(mySsim)$shortName)
```

```
addModules(mySsim) = "C:/Program Files/SyncroSim/1/CorePackages/stockflow.ssimpkg"
addModules(mySsim) = c("C:/Program Files/SyncroSim/1/CorePackages/stockflow.ssimpkg","C:/Program Files/Syncr
is.element("stsim-stock-flow",modules(mySsim)$shortName)

# Create or load a library using a default Session
myLib = ssimLibrary(name="stsim",model="stsim")
session(myLib)
```

`session<-` *Set a SyncroSim session.*

Description

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

Usage

```
session(x) <- value
```

Arguments

x=	A SyncroSim Session.

Value

An SyncroSim object containing a Session.

Examples

```
myLibrary = ssimLibrary()
session(myLibrary)=session()
session(myLibrary)
```

`setBreakpoint` *Set breakpoint of a Scenario.*

Description

Add a Breakpoint object to breakpoints of a Scenario.

Usage

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

Arguments

x	A SyncroSim Scenario
breakpointType	bi: before iteration; ai: after iteration; bt:before timestep; at: aftertimestep
transformerName	'stsim:core-transformer' or?
arguments	A vector of timesteps or iterations e.g. c(1,2)
callback	The function to apply. See STSimBreakpointsTutorial.R for details.

Value

An SyncroSim Scenario object containing breakpoints

`setProperties` *Set the properties of a scenario.*

Description

Set the author, description and/or readOnly status of a scenario.

Usage

```
setProperties(x, author = NULL, description = NULL, readOnly = NULL)
```

Arguments

`x`	An Scenario object.
`author`	An author name.
`description`	A description.
`readOnly`	TRUE or FALSE.

Value

Success or a failure message

`silent` *Check if a Session is silent*

Description

Checks whether a SyncroSim Session is silent or not.

Usage

```
silent(x)
```

Arguments

`x`	A SyncroSim `Session` object.

spatialData *Get spatial inputs or outputs from a SyncroSim scenario.*

Description

Get spatial inputs or outputs from a SyncroSim scenario.

Usage

```
spatialData(x, sheet, iterations = NULL, timesteps = NULL,
  nameFilters = NULL, rat = NULL)
```

Arguments

x	A SyncroSim results Scenario or list of SyncroSim result Scenarios.
sheet	The name of a spatial datasheet. See subset(datasheets(myResultScenario),isSpatial)$name for options.
iterations	A vector of iterations. If NULL(default) all available iterations will be included
timesteps	A vector of timesteps. If NULL(default) all available timesteps will be included.
nameFilters	A vector of strings. Only layer name that include these terms will be returned.
rat	An (optional) raster attribute table. This is dataframe with ID, (optional) Color, and other columns. See raster::ratify() for details.

Details

The Color column of a rat table should have one of these formats:

- R,G,B,alpha: 4 numbers representing red, green, blue and alpha, separated by commas, and scaled between 0 and 255. See rgb() for details.
- R colour names: See colors() for options.
- hexadecimal colors: As returned by R functions such as rainbow(), heat.colors(), terrain.colors(), topo.colors(), gray(), etc.

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 RasterStack or RasterBrick object. See raster package documentation for details.

sqlStatements *Get SELECT and GROUP BY Statements*

Description

Creates SELECT and GROUP BY SQL Satements. Variables are column names. Variables not included in groupBy or aggregate will be dropped from the table.

Usage

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

Arguments

groupBy	Vector of variables to GROUP BY.
aggregate	Vector of variables to aggregate using aggregateFunction
aggregateFunction	An SQL aggregate function (e.g. SUM, COUNT)
where	A list of subset variables.

Value

A list of SQL SELECT and GROUP BY statements.

ssimLibrary *Create or open a library.*

Description

Creates or opens an SSimLibrary object representing a SyncroSim library.

Usage

```
ssimLibrary(name = NULL, ...)

## S4 method for signature 'missingOrNULLOrChar'
ssimLibrary(name = NULL, model = NULL,
  session = NULL, addons = NULL, backup = F, backupName = "backup",
  backupOverwrite = T, forceUpdate = F)

## S4 method for signature 'Project'
ssimLibrary(name)

## S4 method for signature 'Scenario'
ssimLibrary(name)
```

Arguments

name	A file name, model type, SyncroSim Project or Scenario. Optional.
model	The model type. Optional when loading an existing library.
session	A SyncroSim Session. If NULL, the default SyncroSim Session will be used.
addons	One or more addons. See addons() for options.
backup	If TRUE, a backup copy is made when an existing library is opened.
backupName	Added to a library filepath to create a backup library.
backupOverwrite	If TRUE, the existing backup of a library (if any) will be overwritten.
forceUpdate	If FALSE (default) user will be prompted to approve any required updates. If TRUE, required updates will be applied silently.

Details

- If name is SyncroSim Project or Scenario: Returns the SSimLibrary associated with the Project or Scenario.
- If given no name and no model: Opens an existing SyncroSim library in the current working directory - returns an error if more than one library exists. If library does not exist and only one model is installed - creates a library of that type.
- If given a model but no name: Opens or creates a library called <model>.ssim in the current working directory.
- If given a name but no model and name is a valid model type: Attempts to open a library of that name. If library does not exist creates a library of type <name> in the current working directory.
- If given a name but no model and name is not a valid model type: Attempts to open a library of that name. Returns an error if that library does not already exist.
- If given a name and a model: Opens or creates a library called <name>.ssim. Returns an error if the library already exists but is a different type of model.

Value

An SSimLibrary object representing a SyncroSim library.

Methods (by class)

- Project: Get the SSimLibrary associated with a SyncroSim Project.
- Scenario: Get the SSimLibrary associated with a SyncroSim Scenario.

Examples

```
# See the installed models
models(session())

# Create a library called <model>.ssim in the current working directory.
myLibrary = ssimLibrary(model="stsim")
session(myLibrary) #The SycroSim session
filepath(myLibrary) #Path to the file on disk.
info(myLibrary) #Model type and other library information.

# Open an existing SyncroSim library in the current working directory - don't make a backup copy.
```

```
myLibrary = ssimLibrary()

# Create a library with a name in the current working directory
mySecondLibrary = ssimLibrary(name="Lib2",model="stsim")

# Create a library with a name in another directory
myThirdLibrary = ssimLibrary(name=paste0(getwd(),"/Temp/Lib3"),model="stsim")

# Create or load a library using a specific session
mySession = session("C:/Program Files/SyncroSim/1/SyncroSim.Console.exe")
myLibrary = ssimLibrary(name="Lib2",session=mySession)

# Add a project and get the library associated with that project
myProject = project(myLibrary)
myLibrary = ssimLibrary(myProject)
```

`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 names and scope of all datasheets in the library. Used to speed calculations.

See Also

See `ssimLibrary` for options when creating or loading an SyncroSim library.

Examples

```
# Create or load and query a SyncroSim Library.
myLibrary = ssimLibrary(model="stsim")
session(myLibrary)
filepath(myLibrary)
info(myLibrary)

# Add or load a project, then get the SyncroSim Library associated with that Project
myProject = project(myLibrary)
myLibrary = ssimLibrary(myProject)
```

update *Apply updates.*

Description

Apply updates to a SyncroSim Library.

Usage

```
update(x)
```

Arguments

x	An SSimLibrary object, or a Project or Scenario associated with a Library

Value

Success or a failure message from the console.

version *The SyncroSim version*

Description

The version of a SyncroSim Session.

Usage

```
version(x)
```

Arguments

x	A SyncroSim Session object.