Package: RNetLogo (via r-universe)

October 30, 2026

Version 1.0-4

Date 2017-06-10

Title Provides an Interface to the Agent-Based Modelling Platform 'NetLogo'

Author Jan C. Thiele

Maintainer Jan C. Thiele <rnetlogo@gmx.de>

Description Interface to use and access Wilensky's 'NetLogo' (Wilensky 1999) from R using either headless (no GUI) or interactive GUI mode. Provides functions to load models, execute commands, and get values from reporters. Mostly analogous to the 'NetLogo' 'Mathematica' Link <https://github.com/NetLogo/Mathematica-Link>.

Depends R (>= 3.3.2), rJava (>= 0.9-8), igraph

Suggests parallel

SystemRequirements Java (>= 8.0), NetLogo (>= 6.0)

License GPL-2

LazyLoad yes

URL http://rnetlogo.r-forge.r-project.org/

NeedsCompilation no

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2017-06-10 14:57:07 UTC

RemoteUrl https://github.com/cran/RNetLogo

RemoteRef HEAD

RemoteSha 37f374ff56fe6e6f6acfdae38480eb6d085d6d5d

Contents

RNetLogo-package . . . 2
NLCommand . . . 4
NLDfToList . . . 5
NLDoCommand . . . 6
NLDoCommandWhile . . . 7
NLDoReport . . . 8
NLDoReportWhile . . . 10
NLGetAgentSet . . . 11
NLGetGraph . . . 13
NLGetPatches . . . 14
NLLoadModel . . . 16
NLQuit . . . 17
NLReport . . . 19
NLSetAgentSet . . . 20
NLSetPatches . . . 21
NLSetPatchSet . . . 22
NLSourceFromString . . . 23
NLStart . . . 24

Index **28**

`RNetLogo-package` *Provides an interface to the agent-based modelling platform NetLogo*

Description

Interface to use and access Wilensky's NetLogo (Wilensky 1999) from R (R Core Team 2014) using either headless (no GUI) or interactive GUI mode. Provides functions to load models, execute commands, and get values from reporters. Mostly analogous to the NetLogo Mathematica Link `https://github.com/NetLogo/Mathematica-Link`.

Details

Package:	RNetLogo
Type:	Package
Version:	1.0-4
Date:	2017-06-10
License:	GNU GPL v2
LazyLoad:	yes

Start by creating a NetLogo instance by using `NLStart`. Then load a model with the function `NLLoadModel` and then use commands and reporters to do what you like.

It is possible to use NetLogo 3D. Just set the `is3d` argument in `NLStart` to `TRUE`. This functionality is experimental. All RNetLogo functions should work in NetLogo 3D as they do in conventional

2D NetLogo except NLSetPatches, which is not implemented to work with NetLogo 3D properly. NLSetPatchSet delivers a similar functionality usable also with NetLogo 3D but uses a data.frame instead of a matrix.

Note for MAC users: If you want to run RNetLogo in headless mode (without GUI, i.e. setting argument gui=FALSE in NLStart) you have to disable AWT before loading the package. Just execute Sys.setenv(NOAWT=1) before executing library(RNetLogo). If you want to run RNetLogo in GUI mode you have to start it from the JGR application (see https://cran.r-project.org/package=JGR and the note at http://groups.yahoo.com/group/netlogo-users/message/14817).
It can be necessary to run Sys.setenv(NOAWT=1) before loading the JGR package and run Sys.unsetenv("NOAWT") before starting JGR via JGR().

Note for Linux users: If you want to run RNetLogo in GUI mode you should start RNetLogo from JGR (see https://cran.r-project.org/package=JGR).

Note for Windows 32-bit users: Starting RNetLogo (in GUI mode) on 32-bit Windows (not 64-bit Windows running in 32-bit mode) may fail in R version 2.15.2 and 2.15.3 (see description here: https://stat.ethz.ch/pipermail/r-devel/2013-January/065576.html). The reason could be the increased C stack size in 2.15.2 and 2.15.3. If you execute Cstack_info() you can see how large the C stack size is. The problem seems to be resolved with 3.0.0. A workaround is to use R 2.15.1 or 3.x or to start RNetLogo from JGR (see https://cran.r-project.org/package=JGR) or RStudio (see http://www.rstudio.com/).

If you want to increase the Java Heap Space and set other parameters of the Java Virtual Machine (JVM) see notes at NLStart.

See the tutorial published as Thiele (2014) for an introduction. Example codes for all functions can be found in the folder "**examples**" in the installation path of the package. For **performance notes** see the vignette "performanceNotes.pdf" and for an introduction how to **run RNetLogo in parallel** on multicore computers or clusters/grids see the vignette "parallelProcessing.pdf".

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

References

For NetLogo see http://ccl.northwestern.edu/netlogo.

For R Extension for NetLogo see http://r-ext.sourceforge.net/.

For Rserve Extension for NetLogo see http://rserve-ext.sourceforge.net/.

The RNetLogo package is analogous to (and inspired by) the NetLogo Mathematica Link https://github.com/NetLogo/Mathematica-Link.

Thiele, J. (2014) R Marries NetLogo: Introduction to the RNetLogo Package. Journal of Statistical Software 58(2) 1-41. http://www.jstatsoft.org/v58/i02/

Thiele, J., Kurth, W., Grimm, V. (2012) RNetLogo: An R Package for Running and Exploring Individual-Based Models Implemented in NetLogo. Methods in Ecology and Evolution 3(3) 480-483.

R Core Team (2014) R: A Language and Environment for Statistical. R Foundation for Statistical Computing.Vienna, Austria. https://www.r-project.org.

Wilensky, U. (1999) NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

See Also

NLStart, NLLoadModel, NLQuit, rJava package

Examples

```
## Not run:
library(RNetLogo)
nl.path <- "C:/Program Files/NetLogo 6.0/app"
nl.jarname <- "netlogo-6.0.0.jar"
NLStart(nl.path, nl.jarname=nl.jarname)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
NLDoCommand(10, "go")
burned <- NLReport("burned-trees")
print(burned)
NLQuit()

## End(Not run)
```

NLCommand *Executes a command in the referenced NetLogo instance.*

Description

NLCommand executes a NetLogo command (submitted as a string) in the (submitted) NetLogo instance.

Usage

```
NLCommand(..., nl.obj=NULL)
```

Arguments

...	An undefined number of strings with the NetLogo command(s) to be executed. Vectors, lists and data.frames will be represented as NetLogo lists. To set a NetLogo list you can write 'set mylist',c(1,2,3) if the current NetLogo model knows a list named mylist. Furthermore, you can execute multiple commands in series, e.g. 'setup','go'
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

The command can be anything which can be submitted from the NetLogo Command Center. A command has no return value! If you want to return a value from NetLogo use NLReport and other report functions.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLDoCommand, NLDoCommandWhile, NLReport

Examples

```
## Not run:
NLStart("C:/Program Files/NetLogo 6.0/app")
NLCommand("create-turtles 10")

## End(Not run)
```

NLDfToList — *Transforms a data.frame into a NetLogo list or multiple NetLogo lists (one for each column of the data.frame).*

Description

NLDfToList pushes the values of a data.frame into NetLogo lists. The column names of the data.frame are used as names for the NetLogo lists (but the lists must already exist in the current NetLogo model).

Usage

```
NLDfToList(in.data.frame, nl.obj=NULL)
```

Arguments

in.data.frame	The data.frame to fill the NetLogo lists.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

Remember: There must be lists in the NetLogo model with the names of the columns of the submitted data.frame.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLDoCommand, NLDoCommandWhile, NLReport

Examples

```
## Not run:
NLStart("C:/Program Files/NetLogo 6.0/app")
df1 <- data.frame(x=c(1,2,3,4),y=c(5,6,7,8))
# the current NetLogo model must have two variables ('x' and 'y')
# add the variables
NLSourceFromString("globals [x y]", append.model=FALSE)
# set the variables to the data.frame
NLDfToList(df1)

## End(Not run)
```

NLDoCommand *Repeats execution of a command in the referenced NetLogo instance a defined number of times.*

Description

NLDoCommand executes a NetLogo command (submitted as a string) in the submitted NetLogo instance more than one time. It works like NLCommand.

Usage

```
NLDoCommand(iterations, ..., nl.obj=NULL)
```

Arguments

iterations	An integer defining the number of times the command is executed.
...	An undefined number of string(s) with the NetLogo command(s) to be executed. See NLCommand for details.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

This function is used to execute a command more than one time. It is usually used to call a procedure (e.g. "go") for a defined number of times.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLCommand, NLDoCommandWhile, NLReport

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
NLDoCommand(10, "go")

## End(Not run)
```

NLDoCommandWhile — *Repeats a command in the referenced NetLogo instance while a reporter returns* TRUE.

Description

NLDoCommandWhile function executes a NetLogo command (submitted as a string) in the submitted NetLogo instance more than one time. It works like NLCommand but will be repeated as long as the reporter returns TRUE.

Usage

```
NLDoCommandWhile(condition, ..., max.minutes=10, nl.obj=NULL)
```

Arguments

condition	A string with a NetLogo conditional reporter.
...	An undefined number of string(s) with the NetLogo command(s) to be executed. See NLCommand for details.
max.minutes	(optional) If max.minutes > 0 the execution stops after the defined number of minutes (with an error). By default, all executions are stopped after 10 minutes, to prevent the execution of endless loops. If you need more time, increase the value. If you're sure what you do, you can set this value to 0. Then, it will run while the condition is true (i.e. endlessly when the condition is never met. In

GUI mode, you can press "Tools –> Halt" in the NetLogo menu to interrupt a running process.). This can speed up the execution, because the time checking is not applied in this case.

`nl.obj` (optional) A string identifying a reference to a NetLogo instance created with `NLStart`.

Details

This function is used to execute a command for more than one time. It can be used, for example, to run a simulation (by calling "go") while a variable is below some limit.

The condition is evaluated before the submitted commands are executed. If the condition is `FALSE` at the first evaluation, the commands will never be executed.

Attention: Make sure that the condition switches from `TRUE` to `FALSE` sometime, otherwise you will run an endless loop (which is stopped after 10 minutes by default, see argument `max.minutes`).

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

`NLCommand`, `NLDoCommandWhile`, `NLReport`

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
NLDoCommandWhile("burned-trees < 500", "go")

## End(Not run)
```

NLDoReport

Repeats a command and a reporter in the referenced NetLogo instance a defined number of times.

Description

`NLDoReport` executes a NetLogo command (submitted as a string) in the NetLogo instance for more than one time, and executes the reporter after each iteration. It works like a combination of `NLReport` and `NLDoCommand`.

Usage

```
NLDoReport(iterations, command, reporter, as.data.frame=FALSE,
           df.col.names=NULL, nl.obj=NULL)
```

Arguments

iterations	An integer defining how many times the command is repeated.
command	A string with the NetLogo command to be executed.
reporter	A string containing a NetLogo reporter. This argument can also be an R vector containing multiple strings with different NetLogo reporters (separated by commas), like `c("count patches", "count turtles")`. (A similar effect can be reached by using a NetLogo reporter returning a NetLogo list, like `"(list count patches count agents)"` as a single string argument. But the result will not be an R list with nested R lists but an R list with nested R vectors because NetLogo lists are converted to R vectors.)
as.data.frame	(optional) If `TRUE` the function will return a data.frame instead of a list. Default is `FALSE`, which returns a list.
df.col.names	(optional) If `as.data.frame=TRUE`, contains the names of the columns of the returned data.frame. The argument is a vector containing the names as strings in the same order as the submitted reporters.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

This function is used to execute a command more than one time and report a value or a number of values after each iteration. It is often used to call a procedure (e.g. "go") for a defined number of times and save the value of a state variable each time.

Value

A list/nested list or data.frame with the value(s) of the reporter after each execution of the command.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLDoCommand, NLReport, NLDoReportWhile

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
```

```
NLCommand("setup")
burned10 <- NLDoReport(10, "go", "burned-trees")
initburned10 <- NLDoReport(10, "go", c("initial-trees","burned-trees"),
                as.data.frame=TRUE, df.col.names=c("initial","burned"))
str(initburned10)

## End(Not run)
```

`NLDoReportWhile` *Repeats execution of a command and a reporter in the referenced NetLogo instance while a conditional reporter returns* `TRUE`.

Description

`NLDoReportWhile` function executes a NetLogo command (submitted as a string) more than one time and executes the reporter after each iteration. It works like `NLDoReport` but will be repeated while the conditional reporter returns `TRUE`.

Usage

```
NLDoReportWhile(condition, command, reporter, as.data.frame=FALSE,
                df.col.names=NULL, max.minutes=10, nl.obj=NULL)
```

Arguments

`condition`	A string with a NetLogo conditional reporter.
`command`	A string with the NetLogo command to be executed.
`reporter`	A string containing a NetLogo reporter. This argument can also be an R vector containing multiple strings with different NetLogo reporters (separated by commas), like `c("count patches", "count turtles")`. (A similar effect can be reached by using a NetLogo reporter returning a NetLogo list, like `"(list count patches count agents)"` as a single string argument. But the result will not be an R list with nested R lists but an R list with nested R vectors because NetLogo lists are converted to R vectors.)
`as.data.frame`	(optional) If `TRUE` the function will return a data.frame instead a list. Default is `FALSE` which returns a list.
`df.col.names`	(optional) If `as.data.frame=TRUE` defines the names of the columns of the returned data.frame. The argument is a vector containing the names as strings in the same order as the reporters.
`max.minutes`	(optional) If `max.minutes > 0` the execution stops after the defined number of minutes (with an error and no return value). By default, all executions are stopped after 10 minutes, to prevent the execution of endless loops. If you need more time, increase the value. If you're sure what you do, you can set this value to 0. Then, it will run while the condition is true (i.e. endlessly when the condition is never met. In GUI mode, you can press "Tools –> Halt" in the NetLogo menu to interrupt a running process.). This can speed up the execution, because the time checking is not applied in this case.

nl.obj (optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

This function executes a command more than one time and reports a value or a number of values after each iteration. It is usually used to call a procedure (e.g. "go") while a variable is below a boundary value and save the value of a state variable each time. Attention: Make sure that the condition switches from TRUE to FALSE sometime, otherwise you will run an endless loop (which is stopped after 10 minutes by default, see argument max.minutes).

Value

A list/nested list with the value(s) of the reporter after each execution of the command.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLDoCommandWhile, NLReport, NLDoReport

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
burnedLower2200 <- NLDoReportWhile("burned-trees < 2200", "go",
                                   "burned-trees")
str(burnedLower2200)

## End(Not run)
```

NLGetAgentSet *Reports variable value(s) of one or more agent(s) as a data.frame (optional as a list or vector)*

Description

NLGetAgentSet is an easy way to access variable value(s) of one or more agent(s) (in a sorted way) by specifying the name of the agent or the name of an agentset containing the agents. An agent is a turtle, breed, patch, or link. An agentset is a collection of agents.

Usage

```
NLGetAgentSet(agent.var, agentset, as.data.frame=TRUE,
              agents.by.row=FALSE, as.vector=FALSE, nl.obj=NULL)
```

Arguments

agent.var	A string or vector/list of strings with the variable names of the agent(s).
agentset	A string specifying the agent or agentset to be queried.
as.data.frame	(optional) If TRUE (default) the function will return a data.frame with a column for each agent.var and a row for each agent. The column names are taken from the names of the agent.var argument. If FALSE the function will return a list instead of a data.frame (little bit faster when not using agents.by.row=TRUE).
agents.by.row	(optional) This argument has an effect only in combination with as.data.frame=FALSE, i.e. when a list is returned. If agents.by.row=FALSE (default) the returned list contains one list element for each agent.var. Each list element contains a vector with the values of the different agents (agentset). If agents.by.row=TRUE the returned list contains one list element for each agent. Each list element contains a vector with the values of the different requested agent variables (agent.var). Attention: agents.by.row=TRUE makes the function very slow, especially when many agents are requested.
as.vector	(optional) Set this argument to TRUE for getting the result as a simple vector in case of requesting only one agent variable. This is the fastest way to access one agent variable. It does not make sense to set this variable to TRUE together with as.data.frame=TRUE, but as.vector is processed first and will win the race if you accidentally set as.data.frame to TRUE as well. By default as.vector is FALSE.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

It's possible to use all variables of an agent, which can be found in NetLogo's Agent Monitors. It isn't possible to get values from different types of agents (i.e. turtles, patches, links) with one call of NLGetAgentSet.

Value

Returns a data.frame (optional a list) with the variable value(s) of an agent/agents of an agentset. One row for each agent and one column for each agent variable. The result is sorted in the same manner as using sort agentset in NetLogo, i.e. turtles are sorted by their who variable and patches from upper left to lower right.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLReport, NLGetPatches, NLGetGraph

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
# NLLoadModel(...)
NLCommand("create-turtles 10")

colors <- NLGetAgentSet(c("who","xcor","ycor","color"),
                        "turtles with [who < 5]")
str(colors)

# or as a list (slightly faster):
colors.list <- NLGetAgentSet(c("who","xcor","ycor","color"),
                        "turtles with [who < 5]", as.data.frame=FALSE)
str(colors.list)

# or as a list with one list element for each agent
# (very slow!, not recommended especially for large agentsets)
colors.list2 <- NLGetAgentSet(c("who","xcor","ycor","color"),
                        "turtles with [who < 5]", as.data.frame=FALSE,
                        agents.by.row=TRUE)
str(colors.list2)

# getting the ends of links is a little bit more tricky, because they store only the
# reference to the turtles and turtles cannot directly be requested.
# A way to go is:
# create some links
NLCommand("ask turtles [ create-links-with n-of 2 other turtles ]")
link.test <- NLGetAgentSet(c("[who] of end1","[who] of end2"),"links")
str(link.test)

## End(Not run)
```

NLGetGraph *Captures a network.*

Description

NLGetGraph converts a set of NetLogo Link agents into an igraph graph object (see package igraph for details on graph objects).

Usage

```
NLGetGraph(link.agentset="links", nl.obj=NULL)
```

Arguments

link.agentset	(optional) A string defining an agentset of NetLogo Links. Default is "links", which are all links.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

Saves a link network in a graph object of package igraph for network analysis.

Value

Returns a graph object of package igraph.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLGetAgentSet

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <-
"/models/Sample Models/Networks/Preferential Attachment.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
NLDoCommand(4, "go")
graph1 <- NLGetGraph()
plot(graph1, layout=layout.kamada.kawai, vertex.label=V(graph1)$name,
     vertex.shape="rectangle", vertex.size=20, asp=FALSE)

## End(Not run)
```

NLGetPatches — *Reports the values of patch variables as a data.frame (optional as a list, matrix or simple vector)*

Description

NLGetPatches is an easy way to access variables of all patches (default) or of a subset of patches.

Usage

```
NLGetPatches(patch.var, patchset="patches", as.matrix=FALSE,
             as.data.frame=TRUE, patches.by.row=FALSE,
             as.vector=FALSE, nl.obj=NULL)
```

Arguments

patch.var	A string or vector/list of strings with the names of patch variables to report.
patchset	(optional) A string defining which patches to request. By default, values of all patches are returned.
as.matrix	(optional) If this variable is TRUE (default is FALSE), the function will return the result as a matrix representing the NetLogo world. (This option is Only available, if the argument patchset is not used, i.e. if you request all patches and only one patch variable, i.e. length of patch.var is 1.)
as.data.frame	(optional) If TRUE (default) the function returns a data.frame with one column for each patch.var and one row for each patch. The column names are taken from the names of the patch.var argument. If FALSE the function will return a list instead of a data.frame (little bit faster, when not using patches.by.row=TRUE).
patches.by.row	(optional) This argument has an effect only in combination with as.data.frame=FALSE, i.e. when a list is returned. If patches.by.row=FALSE (default) the returned list contains one list element for each patch.var. Each list element contains a vector with the values of the different patches (patchset). If patches.by.row=TRUE the returned list contains one list element for each patch. Each list element contains a vector with the values of the different requested patch variables (patch.var). Attention: patches.by.row=TRUE makes the function very slow, especially when many patches are requested.
as.vector	(optional) Set this argument to TRUE for getting the result as a simple vector in case of requesting only one patch variable. This is the fastest way to access one patch variable. It does not make sense to set this variable to TRUE together with as.data.frame=TRUE or as.matrix=TRUE, but as.vector is processed first and will win the race if you accidentally set as.data.frame or as.matrix to TRUE as well. By default as.vector is FALSE.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

It's possible to use all the variables of a patch, which can be found in NetLogo's Agent Monitors.

Value

Returns a data.frame (optional a list) with the variable value(s) of a patch/patches of a patchset. One row for each patch and one column for each patch variable. The result is sorted (like using sort patchset in NetLogo), e.g. patches are sorted from upper left to lower right.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLReport, NLGetAgentSet, NLGetGraph

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
# NLLoadModel(...)

allpatches <- NLGetPatches(c("pxcor","pycor","pcolor"))
str(allpatches)

# only a subset of patches
subsetpatches <- NLGetPatches(c("pxcor","pycor","pcolor"),
                              "patches with [pxcor < 5]")
str(subsetpatches)

# or as a list (slightly faster):
colors.list <- NLGetPatches(c("pxcor","pycor","pcolor"),
                        "patches with [pxcor < 5]", as.data.frame=FALSE)
str(colors.list)

# or as a list with one list element for each patch
# (very slow!, not recommended especially for large patchsets)
colors.list2 <- NLGetPatches(c("pxcor","pycor","pcolor"),
                        "patches with [pxcor < 5]", as.data.frame=FALSE,
                        patches.by.row=TRUE)
str(colors.list2)

## End(Not run)
```

NLLoadModel *Loads a model into the NetLogo instance.*

Description

NLLoadModel loads a model (*.nlogo file) into the submitted NetLogo instance.

Usage

```
NLLoadModel(model.path, nl.obj=NULL)
```

Arguments

model.path	A string containing either the absolute path to the model file (*.nlogo file) or a relative path to the model file starting from the NetLogo installation directory specified in NLStart.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLStart, NLQuit

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
absolute.model.path <- paste(nl.path,model.path,sep="")
NLLoadModel(absolute.model.path)

relative.model.path <- "models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(relative.model.path)

## End(Not run)
```

NLQuit *Quits a NetLogo instance.*

Description

Quits the NetLogo workspace and closes the GUI window (if started with GUI).

Usage

```
NLQuit(nl.obj=NULL, all=FALSE)
```

Arguments

nl.obj	(optional) A string identifying a reference to the NetLogo instance defined in nl.obj of NLStart.
all	(optional) A boolean variable: If TRUE all active instances of NetLogo created with NLStart are closed. Then, nl.obj argument is not used.

Value

No return value.

Warning

please note that you will not be ask to save changes when closing NetLogo. Furthermore, there is currently no way to kill a NetLogo instance with GUI completely. After executing NLQuit on a GUI instance, you can't run NLStart again. You have to quit your R session first and start a new one. The reason is that NetLogo quits via System.exit (and has no functionality to quit all threads manually) but executing System.exit will terminate the whole JVM which will also terminate rJava and finally R. But there is a trick to run RNetLogo in GUI mode multiple times described in the document parallelProcessing.pdf in directory parallelProcessing in the installation directory of the package. It can happen that some memory is not released although you have executed NLQuit, because shutting down the running JVM via rJava and unloading the required libraries is not possible. Therefore, it is a good idea to start a new R session if possible when you load a new model.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLStart

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
NLQuit()

## End(Not run)
```

NLReport *Reports a value or list of values*

Description

NLReport reports NetLogo data back to R.

Usage

```
NLReport(reporter, nl.obj=NULL)
```

Arguments

reporter	A string containing a NetLogo reporter. (Or a vector of strings.)
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

Every reporter (commands which return a value) that can be called in the NetLogo Command Center can be called with NLReport.

Value

A vector of length one if only one value is returned. Otherwise it is a list or, if necessary, a nested list with the reported values.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLDoReport, NLDoReportWhile, NLGetPatches, NLGetAgentSet

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""))
NLCommand("setup")
NLDoCommand(10, "go")
noburned <- NLReport("burned-trees")
str(noburned)

## End(Not run)
```

NLSetAgentSet *Sets a variable of one or more agent(s) to value(s) in a data.frame or vector.*

Description

NLSetAgentSet is an easy way to set the variable value(s) of one or more agent(s) (by specifying the name of the agent or the name of an agentset containing the agents) to the value(s) of a data.frame or vector.

Usage

```
NLSetAgentSet(agentset, input, var.name=NULL, nl.obj=NULL)
```

Arguments

agentset	A string specifying the agent or agentset for which values should be changed.
input	A data.frame or vector. If a data.frame, it must have one column with the corresponding agent variable name for each agent variable to be set and one row for each agent. The rows have to be sorted in the order NetLogo is processing the agentset with sort agentset (e.g. turtles are sorted by their who value). If a vector, only one agent variable can be set and the name has to be given by the optional argument var.name.
var.name	If input is a simple vector instead of a data.frame it gives the name of the agent variable as a string which should be set with the values of the vector submitted in input. With a vector you can only set one agent variable at a time.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

The agent variable values contained as columns in the input data.frame are changed. The columns of the data.frame have to be named exactly like the agent variable which should get the values. The rows have to be sorted as NetLogo would process the agentset using the sort reporter.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLSetPatches, NLGetAgentSet, NLGetGraph, NLDfToList

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
# NLLoadModel(...)
ag <- NLGetAgentSet(c("xcor","ycor"),"turtles")
ag2 <- data.frame(xcor=ag$xcor, ycor=ag$xcor)
NLSetAgentSet("turtles", ag2)

## End(Not run)
```

NLSetPatches *Sets a variable of all patches in the NetLogo world to the values in a matrix.*

Description

NLSetPatches is an easy way to set the values of all patches to the values of a matrix.

Usage

```
NLSetPatches(patch.var, in.matrix, nl.obj=NULL)
```

Arguments

patch.var	The name of the patch variable to set.
in.matrix	A matrix that represents the NetLogo world (has the same dimensions).
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

The matrix must have the same x- and y-dimensions as the NetLogo world, indices beginning with (1,1). The upper-left cell (1,1) of the matrix represents the upper-left patch of the NetLogo world, no matter where the origin of the NetLogo world is set. This function is not available when running NetLogo 3D. Use NLSetPatchSet instead.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLReport, NLGetAgentSet, NLGetGraph, NLDfToList

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
m1 <- matrix(1:1089 , 33)
NLSetPatches("pcolor", m1)

## End(Not run)
```

NLSetPatchSet *Sets the variable value of one or more patch(es) to value(s) in a data.frame.*

Description

NLSetPatchSet is an easy way to set the variable value of one or more patch(es) to the value(s) of a data.frame.

Usage

```
NLSetPatchSet(patch.var, input, nl.obj=NULL)
```

Arguments

patch.var	This argument gives the name of the patch variable as a string which should be set to the values of the third (for NetLogo 2D) or fourth column (for NetLogo 3D) of the data.frame submitted in input.
input	A data.frame with columns giving the coordinates of a patch and the values for the patch variable to be changed. For conventional 2D NetLogo there has to be a pxcor and a pycor column , for NetLogo 3D there has to be a pxcor, a pycor, and a pzcor column. name of the column that contains the new values for the patch variable has to be equal to the argument patch.var.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

This function is used to update one patch variable for patches identified by their pxcor, pycor (and pzcor in case of NetLogo 3D) values based on values given in a data.frame.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLSetPatches, NLGetAgentSet, NLGetGraph, NLDfToList

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
# NLLoadModel(...)

# for NetLogo 2D:
input <- NLGetPatches(c("pxcor","pycor","pcolor"))
str(input)
# for NetLogo 3D:
input <- NLGetPatches(c("pxcor","pycor","pzcor","pcolor"))
str(input)

input$pcolor <- floor(abs(rnorm(nrow(input))*100))
patch.var <- "pcolor"
NLSetPatchSet(patch.var, input)

## End(Not run)
```

NLSourceFromString *Creates or appends NetLogo code from R.*

Description

NLSourceFromString is a way to create/append a NetLogo model's source code dynamically from R.

Usage

```
NLSourceFromString(..., append.model=TRUE, nl.obj=NULL)
```

Arguments

...	An undefined number of strings containing NetLogo model source code to be printed into the procedures tab. Line breaks within a string can be represented as \n.
append.model	(optional) Determines whether existing code in the procedures tab (i.e. a loaded model) will be appended by the new code or will be replaced. By default, all existing code will be appended.
nl.obj	(optional) A string identifying a reference to a NetLogo instance created with NLStart.

Details

This function only works with NetLogo instances with GUI. It doesn't work in headless mode.

Value

No return value.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

See Also

NLReport, NLGetAgentSet, NLGetGraph, NLDfToList

Examples

```
## Not run:
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
setup <- "to setup\n ca\n crt 10\nend \n"
go <- "to go\n ask turtles [\n  set xcor random-xcor\n
       set ycor random-ycor\n ]\nend \n"
reporter1 <- "to-report noturtles\n report count turtles\n end \n"
NLSourceFromString(setup,go,reporter1, append.model=FALSE)
NLCommand("setup")
NLCommand("go")
noturtles <- NLReport("noturtles")
print(noturtles)

## End(Not run)
```

NLStart *Creates an instance of NetLogo*

Description

NLStart creates a new instance of NetLogo in either headless (without the Graphical User Interface) or GUI mode.

Usage

```
NLStart(nl.path, gui=TRUE, nl.obj=NULL, is3d=FALSE, nl.jarname='netlogo-6.0.0.jar')
```

Arguments

nl.path	An absolute path to your NetLogo installation (the folder where the NetLogo.jar is) starting from the root. On Windows, for example, something like "C:/Program Files/NetLogo 6.0/app".
gui	(optional) A boolean value: if TRUE, NetLogo will be started with GUI (only one instance with GUI can be created currently!). FALSE will start NetLogo in headless mode.

`nl.obj` (optional) A string which is used to identify the created NetLogo instance reference internally (in `.rnetlogo` environment). To refer to this instance just use the same name in the other functions of this package. If `nl.obj=NULL` (default), the internal name to the reference is `_nl.intern_` and is not needed to be submitted to the other functions of this package. After using NLQuit, the identical name can be used again for a new instance.

`is3d` (optional) A boolean value: if `TRUE`, NetLogo 3D will be started. `FALSE` will start the conventional 2D NetLogo. This functionality is experimental. All RNetLogo functions should work in NetLogo 3D as they do in conventional 2D NetLogo except NLSetPatches, which is currently not implemented to work in NetLogo 3D properly.

`nl.jarname` (optional) The name of the NetLogo jar file. Since NetLogo 6.0 the jar file includes the version number. Default value is netlogo-6.0.0.jar. For other version the name has to be set here.

Details

You can start multiple instances of NetLogo in headless mode and store each in another variable (using `nl.obj`) but it is not possible to start multiple instances in GUI mode. (It would result in an crash of R since there is no way to detach the Java Virtual Machine via rJava.) But there is a trick to run RNetLogo in GUI mode multiple times described in the document `parallelProcessing.pdf` in directory `parallelProcessing` in the installation directory of the package.

Note for Mac OS users: If you want to run RNetLogo in headless mode (without GUI, i.e. setting argument `gui=FALSE`) you have to disable AWT before loading the package. Just execute `Sys.setenv(NOAWT=1)` before executing `library(RNetLogo)`. If you want to run RNetLogo in GUI mode you have to start it from the JGR application (see https://cran.r-project.org/package=JGR and the note at http://groups.yahoo.com/group/netlogo-users/message/14817).
It can be necessary to run `Sys.setenv(NOAWT=1)` before loading the JGR package and run `Sys.unsetenv("NOAWT")` before starting JGR via `JGR()`.

Note for Linux users: If you want to run RNetLogo in GUI mode you should start RNetLogo in the JGR application (see https://cran.r-project.org/package=JGR).

Note for Windows 32-bit users: Starting RNetLogo (in GUI mode) on 32-bit Windows (not 64-bit Windows running in 32-bit mode) may fail in R version 2.15.2 and 2.15.3 (see description here: https://stat.ethz.ch/pipermail/r-devel/2013-January/065576.html). The reason could be the increased C stack size in 2.15.2 and 2.15.3. If you execute `Cstack_info()` you can see how large the C stack size is. The problem seems to be resolved with 3.0.0. A workaround is to use R 2.15.1 or 3.x or to start RNetLogo from JGR (see https://cran.r-project.org/package=JGR) or RStudio (see http://www.rstudio.com/).

Avoid manually changing the working directory of R, because NetLogo needs to have the working directory pointed to its installation path. As the R working directory and the Java working directory depend on each other, changing the R working directory can result in unexpected behavior of NetLogo. Therefore, you should use absolute paths for I/O processes in R instead of submitting `setwd(...)`. Note that the RNetLogo package changes the working directory automatically when loading NetLogo and changes back to the former working directory closing the last active instance of NetLogo with `NLQuit`.

As mentioned in `NLQuit`, it is currently not possible to quit NetLogo completely.

If you want to specify options for the underlying Java Virtual Machine (JVM), like increasing the Java Heap Space for large models, execute `options(java.parameters="...")` before loading the RNetLogo package with `library(RNetLogo)` or `require(RNetLogo)`. For increasing the Java Heap Space it can be `options(java.parameters="-Xmx1024m")`, for example. Use a vector of strings for setting multiple options, for example
`options(java.parameters=c("-server","-Xmx1300m"))`. See also
http://ccl.northwestern.edu/netlogo/docs/faq.html#howbig and rJava manual.

See the directory `examples` in the installation directory of the package for example codes to all RNetLogo functions.

See Thiele (2014) (also included in directory `tutorial` in the installation directory of the package) for a step-by-step usage tutorial.

See the vignette `performanceNotes.pdf` for performance notes.

See the vignette `parallelProcessing.pdf` on how to run RNetLogo on multiple processors/clusters in parallel.

Value

No return value.

Warning

It's not possible to run multiple instances of NetLogo in GUI mode! Closing NetLogo from the NetLogo Window is blocked, because it would quit the whole R process. To close the NetLogo call `NLQuit`. If you use the headless mode you should first load a model with `NLLoadModel` before executing other commands or reporters. In GUI mode you can execute commands and reporters already with the initial empty model without loading a specific one.

Author(s)

Jan C. Thiele <rnetlogo@gmx.de>

References

Thiele, J. (2014) R Marries NetLogo: Introduction to the RNetLogo Package. Journal of Statistical Software 58(2) 1-41. http://www.jstatsoft.org/v58/i02/

See Also

`NLQuit`

Examples

```
## Not run:
library(RNetLogo)
nl.path <- "C:/Program Files/NetLogo 6.0/app"
NLStart(nl.path)
NLCommand("create-turtles 10")
noturtles <- NLReport("count turtles")
print(noturtles)
```

```
# create a second NetLogo instance in headless mode (= without GUI)
# stored in a variable
nlheadless1 <- "nlheadless1"
NLStart(nl.path, gui=F, nl.obj=nlheadless1)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""), nl.obj=nlheadless1)
NLCommand("setup", nl.obj=nlheadless1)
burned1 <- NLDoReport(20, "go", c("ticks","burned-trees"),
                      as.data.frame=TRUE,df.col.names=c("tick","burned"),
                      nl.obj=nlheadless1)
print(burned1)

# create a third NetLogo instance in headless mode (= without GUI)
# with explicit name of stored object
nlheadless2 <- "nlheadless2"
NLStart(nl.path, gui=F, nl.obj=nlheadless2)
model.path <- "/models/Sample Models/Earth Science/Fire.nlogo"
NLLoadModel(paste(nl.path,model.path,sep=""), nl.obj=nlheadless2)
NLCommand("setup", nl.obj=nlheadless2)
burned2 <- NLDoReport(10, "go", c("ticks","burned-trees"),
                      as.data.frame=TRUE,df.col.names=c("tick","burned"),
                      nl.obj=nlheadless2)
print(burned2)

## End(Not run)
```

Index

∗ **NLCommand**
NLCommand, 4
∗ **NLDfToList**
NLDfToList, 5
∗ **NLDoCommandWhile**
NLDoCommandWhile, 7
∗ **NLDoCommand**
NLDoCommand, 6
∗ **NLDoReportWhile**
NLDoReportWhile, 10
∗ **NLDoReport**
NLDoReport, 8
∗ **NLGetAgentSet**
NLGetAgentSet, 11
∗ **NLGetGraph**
NLGetGraph, 13
∗ **NLGetPatches**
NLGetPatches, 14
∗ **NLLoadModel**
NLLoadModel, 16
∗ **NLQuit**
NLQuit, 17
∗ **NLReport**
NLReport, 19
∗ **NLSetAgentSet**
NLSetAgentSet, 20
NLSetPatchSet, 22
∗ **NLSetPatches**
NLSetPatches, 21
∗ **NLSourceFromString**
NLSourceFromString, 23
∗ **NLStart**
NLStart, 24
∗ **NetLogo**
RNetLogo-package, 2
∗ **RNetLogo**
NLCommand, 4
NLDfToList, 5
NLDoCommand, 6
NLDoCommandWhile, 7
NLDoReport, 8
NLDoReportWhile, 10
NLGetAgentSet, 11
NLGetGraph, 13
NLGetPatches, 14
NLLoadModel, 16
NLQuit, 17
NLReport, 19
NLSetAgentSet, 20
NLSetPatches, 21
NLSetPatchSet, 22
NLSourceFromString, 23
NLStart, 24
∗ **agent-based**
RNetLogo-package, 2
∗ **individual-based**
RNetLogo-package, 2
∗ **interface**
NLCommand, 4
NLDfToList, 5
NLDoCommand, 6
NLDoCommandWhile, 7
NLDoReport, 8
NLDoReportWhile, 10
NLGetAgentSet, 11
NLGetGraph, 13
NLGetPatches, 14
NLLoadModel, 16
NLQuit, 17
NLReport, 19
NLSetAgentSet, 20
NLSetPatches, 21
NLSetPatchSet, 22
NLSourceFromString, 23
NLStart, 24

NLCommand, 4, *6–8*
NLDfToList, 5, *20, 21*, *23, 24*
NLDoCommand, *5*, 6, *6*, *8, 9*

NLDoCommandWhile, *5, 6*, 7, *7, 8*, *11*
NLDoReport, 8, *10, 11*, *19*
NLDoReportWhile, *9*, 10, *19*
NLGetAgentSet, 11, *14*, *16*, *19–21*, *23, 24*
NLGetGraph, 13, *13*, *16*, *20, 21*, *23, 24*
NLGetPatches, *13*, 14, *19*
NLLoadModel, *2*, *4*, 16, *26*
NLQuit, *4*, 17, *17*, *26*
NLReport, *4–9*, *11*, *13*, *16*, 19, *21*, *24*
NLSetAgentSet, 20
NLSetPatches, *3*, *20*, 21, *23*
NLSetPatchSet, *3*, *21*, 22
NLSourceFromString, 23
NLStart, *2–6*, *8, 9*, *11, 12*, *14, 15*, *17–23*, 24

RNetLogo *(*RNetLogo-package*)*, 2
RNetLogo-package, 2