Package ‘nlexperiment’

September 14, 2015

Type Package

Title Exploration of NetLogo Agent Based Models

Version 0.1.5

Date 2015-09-14

Author Darko Bergant

Maintainer Darko Bergant <darko.bergant@gmail.com>

BugReports https://github.com/bergant/nlexperiment/issues

URL http://bergant.github.io/nlexperiment/

Description A tool for NetLogo experiment definition,
exploring simulation results and model optimization.
Makes it easy to turn the cycle of experiment definition,
data analysis, visualisations and
parameter fitting into readable and reproducible documents.

Depends R (>= 3.1)

License GPL-2

Imports RNetLogo,
digest

Suggests knitr,
dplyr,
png,
ggplot2,
testthat,
tgp,
fast

LazyData TRUE

R topics documented:

`nlexperiment-package` *nlexperiment: NetLogo experiments*

Description

Exploration of NetLogo (Wilensky 1999) agent based models.

Details

A tool for NetLogo experiment definition, exploring simulation results and model optimization. Makes it easy to turn the cycle of experiment definition, data analysis, visualisations and parameter fitting into readable and reproducible documents.

RNetLogo package (Thiele 2014) is used as an interface to NetLogo environment.

Functions in **nlexperiment** assume the following steps:

- Define NetLogo experiment object with parameter sets, measures and simulation options (see `nl_experiment` function).
- Run experiment (see `nl_run`). The result of running an experiment keeps original experiment definition along with the simulation results and makes the process of model analysis more concise and reproducible. To run the simulation in parallel working processes use the `parallel` attribute in `nl_run` function.
- Analyse and present results of simulation(s). See `nl_get_result` for getting different data from the result and `nl_show_step`, `nl_show_patches` for pre-defined plots.
- When additional questions pop out, changes to experiment will be needed. Refine the original definition of the experiment by changing only parameter sets (`nl_set_param_values`), set different measures (`nl_set_measures`) or set other simulation options (`nl_set_run_options`).

References

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

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/

The ideas and principles of NetLogo experiment definition is taken from the NetLogo's Behavior Space tool http://ccl.northwestern.edu/netlogo/docs/behaviorspace.html and BehaviorSearch tool http://www.behaviorsearch.org/

Examples

```
## Not run:
# Set the path to your NetLogo installation
nl_netlogo_path("c:/Program Files (x86)/NetLogo 5.1.0/")

# Create NetLogo experiment of Net Logo Fire model
experiment <- nl_experiment(
  model_file = "models/Sample Models/Earth Science/Fire.nlogo",
  while_condition = "any? turtles",
  repetitions = 10,
  run_measures = measures(
    percent_burned = "(burned-trees / initial-trees) * 100",
    progress = "max [pxcor] of patches with [pcolor > 0 and pcolor < 55]"
  ),
  param_values = list(
    density = seq(from = 55, to = 62, by = 1)
  )
)

# Run the experiment using multi-core processing
result <- nl_run(experiment, parallel = TRUE)

# Get observations data frame
dat <- nl_get_run_result(result)

# plot percent burned by density
library(ggplot2)
ggplot(dat, mapping = aes(x = factor(density), y = percent_burned) ) +
  geom_violin()

## End(Not run)
```

`nl_default_mapping` *Default mapping from R names to NetLogo variables*

Description

Creates mapping with simple rule: changes every character `_.` to ? and `_` to `-`.

Usage

```
nl_default_mapping(param_values)
```

Arguments

`param_values`	Parameter values in list or data frame

Value

Named vector with default mapping. Use as function argument in `nl_experiment` mapping.

Examples

```
param_values = list(
  world_size = 50,
  population = 80,
  max_align_turn = c(1, 5, 20),
  max_cohere_turn = c(1, 3, 20),
  max_separate_turn = c(1, 1.5, 20),
  vision = c(1, 3, 10),
  minimum_separation = c(1, 3, 10),
  .dummy = c(1:0)
)

nl_default_mapping(param_values)

# Define experiment mapping with a function instead of named vector:
experiment <- nl_experiment(
  model_file = "models/Sample Models/Biology/Flocking.nlogo",

  param_values = list(
    world_size = 50,
    population = 80,
    max_align_turn = c(1, 5, 20),
    max_cohere_turn = c(1, 3, 20),
    max_separate_turn = c(1, 1.5, 20),
    vision = c(1, 3, 10),
    minimum_separation = c(1, 3, 10),
    .dummy = c(1:0)
  ),
  mapping = nl_default_mapping
)

# check experiment parameter names mapping
cbind(experiment$mapping)
```

`nl_eval_run` *Evaluate experiment with specific parameters*

Description

Function `nl_eval_run` runs experiment as with `nl_run` but requires started NetLogo instance with loaded model.

Function `nl_eval_init` starts NetLogo instance and loads the NetLogo model. When using parallel version it initializes several processes and returns cluster objects

Function `nl_eval_close` stops NetLogo instance

Function `nl_get_eval_fun` returns a function wich calls `nl_eval_run` but does not need additional parameters.

Usage

```
nl_eval_run(param_set, experiment, criteria = NULL, print_progress = FALSE,
  call_back = NULL, parallel = FALSE, cluster = NULL,
  param_names = NULL)

nl_eval_init(experiment, parallel = FALSE, max_cores = NULL)

nl_eval_close(parallel = FALSE, cluster = NULL)

nl_get_eval_fun(experiment, param_names, parallel = FALSE, cluster = NULL,
  criteria, call_back = NULL)
```

Arguments

`param_set`	parameter set (a list of parameters with values)
`experiment`	NetLogo experiment object (see `nl_experiment`)
`criteria`	Which experiment evaluation criteria to be returned
`print_progress`	print evaluation progress
`call_back`	A call-back function for tracing result in optimization processes
`parallel`	If TRUE nl_eval_init returns cluster object which should be passed to nl_eval_run and nl_eval_close.
`cluster`	Required for parallel execution (nl_eval_init returns cluster object)
`param_names`	parameter names for parameter set
`max_cores`	If not defined all available cores are used.

Details

Use `nl_eval_run` when parameter set depend on previous evaluation (parameter fitting / callibration / optimization methods). It can use the same experiment object as `nl_run` function. Evaluation criteria should be defined. (see `nl_experiment` or `nl_set_measures`).

Examples

```
## Not run:

  experiment <- nl_experiment(
    model_file = "models/Sample Models/Biology/Flocking.nlogo",

    setup_commands = c("setup", "repeat 100 [go]"),
    iterations = 5,

    param_values = list(
      world_size = 50,
      population = 80,
      vision = 6,
      min_separation = seq(from = 0, to = 4, by = 0.5),
      max_align_turn = seq(from = 0, to = 20, by = 2.5)
    ),
    mapping = c(
      min_separation = "minimum-separation",
      max_align_turn = "max-align-turn"),
    step_measures = measures(
      converged = "1 -
      (standard-deviation [dx] of turtles +
      standard-deviation [dy] of turtles) / 2",
      mean_crowding =
        "mean [count flockmates + 1] of turtles"
    ),
    eval_criteria = criteria(
      c_converged = mean(step$converged),
      c_mcrowding = mean(step$mean_crowding)
    ),

    repetitions = 10,                 # repeat simulations 10 times

    eval_aggregate_fun = mean,        # aggregate over repetitions

    eval_mutate = criteria(           # evaluation criterium
      eval_value =
        sqrt((c_mcrowding - 8)^2 + 400*(c_converged - 1)^2)
    )
 )

 library(dfoptim)

 cl <- nl_eval_init(experiment, parallel = TRUE)
 trace <- nl_eval_tracer(verbose = FALSE)
 param_range <- nl_get_param_range(experiment)
 set.seed(1)

 o_result <- nmkb(
   par = (param_range$upper + param_range$lower)/2,
   fn = nl_eval_run,
     experiment = experiment,
     criteria = "eval_value",
     call_back = trace$add,
     parallel = TRUE, cluster = cl,
     param_names = names(param_range$lower),
   lower = param_range$lower,
   upper = param_range$upper,
   control = list(maxfeval = 200)
 )
 nl_eval_close(parallel = TRUE, cl)

## End(Not run)
```

`nl_eval_tracer` *Iterations call-back factory*

Description

Iterations call-back factory

Usage

```
nl_eval_tracer(verbose = TRUE)
```

Arguments

`verbose`	When TRUE adding new data will print the line

`nl_experiment` *Create NetLogo experiment object*

Description

Use this function to create NetLogo experiment object.

Usage

```
nl_experiment(model_file, iterations = NULL, while_condition = NULL,
  repetitions = 1, random_seed = NULL, step_measures = NULL,
  run_measures = NULL, mapping = NULL, param_values = NULL,
  agents_after = NULL, agents_step = NULL, patches_after = NULL,
  export_view = FALSE, export_world = FALSE, setup_commands = "setup",
  go_command = "go", eval_criteria = NULL, eval_aggregate_fun = NULL,
  eval_mutate = NULL, data_handler = NULL)
```

Arguments

`model_file`	An absolute path to your NetLogo model file (.nlogo)
`iterations`	Number of iterations to run. Alternatively define while_condition to stop simulation.
`while_condition`	A string with a NetLogo conditional reporter. (for example: "ticks < 100")
`repetitions`	How many times to run the model with the same parameters. It is set to 1 by default. Result data sets will include run_id as additional variable to identify the specific runs. To change repetitions of existing experiment object use `nl_set_run_options`
`random_seed`	If defined, random seed will be set for each run. Note: using random seed and repetitions > 1 does not make sense.
`step_measures`	Measures per each simulation step in a named character vector. Use measures() function to construct measures in right format. To change step measures of existing experiment object use `nl_set_measures`
`run_measures`	Measures per each simulation run in a named character vector. Use measures() function to construct measures in right format. To change run measures of existing experiment object use `nl_set_measures`
`mapping`	Mapping between R and NetLogo parameters in named character vector. For example: c(diffusion_rate = "diffusion-rate", population = "population")
`param_values`	A data.frame with parameter values or a list of values to be expanded to all combinations of values
`agents_after`	Agents reporters see `nl_set_agent_reports`
`agents_step`	Agents reporters see `nl_set_agent_reports`
`patches_after`	Patches reporters see `nl_set_agent_reports`

`export_view`	If set to TRUE, the views will be exported to a png image files for each run (when running the experiment)
`export_world`	If set to TRUE, the world will be exported to a csv file for each run
`setup_commands`	NetLogo command strings to execute to setup the model
`go_command`	NetLogo command string to execute the step in the model
`eval_criteria`	A criteria calculation expressions. May use `step` or `run` data frames to calculate criteria. Elements from `step` should be aggregated. Must return named numeric vector.
`eval_aggregate_fun`	Aggregation function (used to aggregate criteria values when repetitions > 1)
`eval_mutate`	Add criteria based on aggregated values
`data_handler`	Function to handle observations. If handler is defined the observations will not be stored in result elements when running the experiment with ‘nl_run‘ function.

Value

NetLogo experiment object

See Also

To run experiment use `nl_run`. To change existing experiment object see `nl_set_measures`, `nl_set_run_options` and `nl_set_param_values`.

Examples

```
experiment <- nl_experiment(
  model_file = "models/Sample Models/Earth Science/Fire.nlogo",
  while_condition = "any? turtles",
  repetitions = 20,
  run_measures = measures(
    percent_burned = "(burned-trees / initial-trees) * 100",
    progress = "max [pxcor] of patches with [pcolor > 0 and pcolor < 55]"
  ),
  param_values = list(
    density = seq(from = 55, to = 62, by = 1)
  )
)
```

`nl_export_path` *Get and set export path*

Description

Get and set export path

Usage

```
nl_export_path(export_path = NULL)
```

Arguments

`export_path`	target folder to export files

Details

Setting export path is optional. If not set, running experiments with export options (view images and worlds) will create "export" folder in working directory. Option is defined per session.

`nl_get_fast_sensitivity` *Calculate sensitivity according to the FAST algorithm*

Description

Uses `sensitivity` from **fast** package to calculate a series of model outputs according to the FAST alogrithm

Usage

```
nl_get_fast_sensitivity(result, criteria)
```

Arguments

`result`	A nlexperiment result object
`criteria`	Name of evaluation criteria

Details

Only works when parameter value sets are defined with `nl_param_fast` function. Criteria must be defined in experiment (see `nl_experiment`, `eval_criteria` argument). Sensitivity is callculated for every simulation iteration (run_id).

Value

A data frame with sensitivity from simulation results for every simulation repetition (run_id)

Examples

```
## Not run:

experiment <- nl_experiment(
  model_file = "models/Sample Models/Biology/Flocking.nlogo",
  setup_commands = c("setup", "repeat 100 [go]"),
  iterations = 5,

  param_values = nl_param_fast(
    world_size = 50,
    population = 80,
    max_align_turn = c(1, 5, 20),
    max_cohere_turn = c(1, 3, 20),
    max_separate_turn = c(1, 1.5, 20),
    vision = c(1, 3, 10),
    minimum_separation = c(1, 3, 10)
  ),
  mapping = c(
    max_align_turn = "max-align-turn",
    max_cohere_turn = "max-cohere-turn",
    max_separate_turn = "max-separate-turn",
    minimum_separation = "minimum-separation",
    world_size = "world-size",
  ),
  step_measures = measures(
    converged = "1 -
      (standard-deviation [dx] of turtles +
       standard-deviation [dy] of turtles) / 2",
    mean_crowding =
      "mean [count flockmates + 1] of turtles"
  ),
  eval_criteria = criteria(                # aggregate over iterations
    c_converged = mean(step$converged),
    c_mcrowding = mean(step$mean_crowding)
  ),

  repetitions = 10,                        # repeat simulations 10 times
  random_seed = 1:10
)

#run experiment
result <- nl_run(experiment, parallel = TRUE)

#get sensitivity data
sensitivity_data <- nl_get_fast_sensitivity(result, "c_converged")

## End(Not run)
```

---

nl_get_param_range *Get ranges of experiment parameter sets*

Description

Upper and lower value for each parameter in experiment parameter sets

Usage

```
nl_get_param_range(experiment, diff_only = TRUE, as.data.frame = FALSE)
```

Arguments

experiment	NetLogo experiment object
diff_only	Uses only non-constant parameters
as.data.frame	Return in a data frame

Value

A list with lower and upper values for all parameters in experiment parameter set. When as.data.frame is specified a data frame with lower and upper columns.

`nl_get_result` *Get observations joined with parameter values*

Description

Observations are stored in result object only with references to parameter sets (param_set_id). `nl_get_result` joins the data with actual parameters used for each observation.

Usage

```
nl_get_result(result, add_parameters = TRUE, type = "run",
  sub_type = NULL, ...)

nl_get_run_result(result, add_parameters = TRUE, ...)

nl_get_step_result(result, add_parameters = TRUE, ...)

nl_get_criteria_result(result, add_parameters = TRUE, ...)
```

Arguments

`result`	A nlexperiment result object
`add_parameters`	Add parameter values from parameter space to the results
`type`	Observation type: "run", "step", "criteria", "agents_after", "patches_after" See `nl_run` for simulations result structure.
`sub_type`	Observation sub-type (in case of individual agents measures the sub type is a name of the measure)
`...`	expressions to transform resulting data frame

`nl_import_sliders` *Import sliders from NetLogo model file*

Description

Reads NetLogo model file and parses slider section

Usage

```
nl_import_sliders(experiment, max_values = 20)
```

Arguments

`experiment`	NetLogo experiment object
`max_values`	Maximum values per parameter

Details

Imports parameter names and ranges from sliders defined in NetLogo model file. Based on information from https://github.com/NetLogo/NetLogo/wiki/Model-file-format and https://github.com/NetLogo/NetLogo/wiki/V
Format

Value

A list with slider data, suggested parameter sets and mapping

---

`nl_map_parameter` *Internal: maps parameter*

Description

Internal: maps parameter

Usage

```
nl_map_parameter(experiment, parameter_name)
```

Arguments

`experiment`	Experiment object
`parameter_name`	Parameter name to map

Value

NetLogo variable name

`nl_netlogo_path` *Get and set netlogo path*

Description

Get and set netlogo path

Usage

```
nl_netlogo_path(nl_path = NULL)
```

Arguments

`nl_path`	An absolute path to your NetLogo installation On Windows, for example, something like "C:/Program Files/NetLogo 5.1.0".

nl_param_fast *Generate a parameter value sets for the FAST method*

Description

Uses `fast_parameters` from **fast** package to create parameter sets for Fourier Amplitute Sensitivity Test (FAST).

Usage

```
nl_param_fast(...)
```

Arguments

`...`	Named list of parameter ranges (numeric vectors)

Details

Uses only parameters with min != max values to create parameter sets. Adds dummy variable.

Value

A data frame with parameter value sets.

See Also

Use `nl_get_fast_sensitivity` to get sensitivity data. See fast package documentation for FAST algorithm details. from the simulation results. See `nl_param_lhs` for latin hypercube sampling.

Examples

```
param_values <- nl_param_fast(
  world_size = 50,
  population = 80,
  max_align_turn = c(1, 5, 20),
  max_cohere_turn = c(1, 3, 20),
  max_separate_turn = c(1, 1.5, 20),
  vision = c(1, 3, 10),
  minimum_separation = c(1, 3, 10)
)
```

nl_param_lhs *Create parameter sets with latin hypercube sampling*

Description

Parameter sets are created with `lhs` function from **tgp** package

Usage

```
nl_param_lhs(n, ...)
```

Arguments

n	Number of parameter sets
...	Named list of parameter ranges (numeric vectors)

Value

A data frame with parameter value sets

Examples

```
experiment <- nl_experiment(
  model_file = "models/Sample Models/Biology/Flocking.nlogo",
  setup_commands = c("setup", "repeat 100 [go]"),
  iterations = 5,

  param_values = nl_param_lhs(
    n = 100,                          # create 100 parameter value sets
    world_size = 50,
    population = 80,
    vision = 6,
    min_separation = c(0, 4),
    max_align_turn = c(0, 20)
  ),
  mapping = c(
    min_separation = "minimum-separation",
    max_align_turn = "max-align-turn"),

  step_measures = measures(
    converged = "1 -
    (standard-deviation [dx] of turtles +
    standard-deviation [dy] of turtles) / 2",
    mean_crowding =
      "mean [count flockmates + 1] of turtles"
  ),
  eval_criteria = criteria(
    c_converged = mean(step$converged),
    c_mcrowding = mean(step$mean_crowding)
  ),

  repetitions = 10,                     # repeat simulations 10 times
  random_seed = 1:10,

  eval_aggregate_fun = mean             # aggregate over repetitions
)
```

`nl_param_oat` *Create parameter sets with "one-at-a-time" (OAT) approach*

Description

Create parameter sets with "one-at-a-time" (OAT) approach

Usage

```
nl_param_oat(n, ...)
```

Arguments

`n`	Number of parameter sets per parameter
`...`	Named list of parameter ranges (numeric vectors) Minimum and maximum values are used as a range and median as the default value. Parameters with only 1 value are treated as constants.

Value

A data frame with parameter value sets

See Also

See also `nl_param_lhs` for latin cube and `nl_param_fast` for FAST parameter sampling.

Examples

```
# create 5 values for every parameter:
nl_param_oat(n = 5, P1 = c(1, 4, 10), P2 = c(4, 11, 20))

# using constant parameters:
nl_param_oat(n = 5, P1 = c(1, 4, 10), P2 = c(4, 11, 20), P3 = 6)

# define NetLogo experiment with OAT design:
experiment <- nl_experiment(
  model_file = "models/Sample Models/Biology/Flocking.nlogo",
  setup_commands = c("setup", "repeat 100 [go]"),
  iterations = 5,

  param_values = nl_param_oat(
    n = 25,                           # create 25 value sets per parameter
    max_align_turn = c(0, 5, 20),
    max_cohere_turn = c(0, 3, 20),
    max_separate_turn = c(0, 1.5, 20),
    vision = c(1, 3, 10),
    minimum_separation = c(0, 3, 10),
    .dummy = c(0,0.5,1),
    world_size = 50,
    population = 80
  ),
  mapping = nl_default_mapping,

  step_measures = measures(
    converged = "1 -
    (standard-deviation [dx] of turtles +
    standard-deviation [dy] of turtles) / 2",
    mean_crowding =
      "mean [count flockmates + 1] of turtles"
  ),
  eval_criteria = criteria(
    c_converged = mean(step$converged),
    c_mcrowding = mean(step$mean_crowding)
  ),

  repetitions = 10,                       # repeat simulations 10 times
  random_seed = 1:10

)
```

---

nl_run *Run NetLogo experiment*

Description

Runs NetLogo model for defined every parameter and repetitions. Returns a list of data frames for each measure defined in experiment.

Usage

```
nl_run(experiment, print_progress = FALSE, gui = FALSE, parallel = FALSE,
  max_cores = NULL)
```

Arguments

`experiment`	NetLogo experiment object
`print_progress`	Set to TRUE if you want to follow the progress in the console
`gui`	Start NetLogo with GUI (by default NetLogo is run in headless mode)
`parallel`	Runs experiment in parallel worker processes (requires parallel package)
`max_cores`	(optional) only relevant if parallel = TRUE. If not defined all available processors will be used

Details

Model is run for each parameter combination defined in parameter sets If `repetition` (defined in experiment) is greater than 1 then each run for a parameter set is repeated accordingly. Before each run the parameters are set and setup procedure(s) are called. After each run criteria function(s) are calculated (if defined)

Use parallel option if there are more than a few runs per processor core.

Value

Returns an object of class `nl_result`. It is a list containing at most the following components:

`step`	a data frame with observations based on temporal (step) measures. It includes at least param_set_id (id of parameter set), run_id (ID of simulation repetition), step_id (ID of simulation step), and columns named after the temporal measures
`run`	a data frame with observations based on final run measures. It includes at least param_set_id (id of parameter set), run_id (ID of simulation repetition), and columns named after the temporal measures
`agents_after`	a data frame with observations based on agents after each simulation run
`agents_before`	a data frame with observations based on agents before each simulation run

`patches_after`	a data frame with observations based on patches after each simulation run
`patches_before`	a data frame with observations based on patches before each simulation run
`criteria`	a data frame with values provided by criteria expressions (`eval_criteria` in experiment definition possibly aggregated by `eval_aggregate_fun`) and additional criteria defined by `eval_mutate` expressions
`export`	a filename list with reference to parameter sets and simulation repetitions
`duration`	time spent to complete the experiment (in `difftime`)
`experiment`	original NetLogo experiment object used

See Also

See `nl_experiment` for creating NetLogo experiment object.

`nl_set_agent_reports` *Set or change agent reports*

Description

Set reporting of variable value(s) of one or more agent(s) as a data.frame

Usage

```
nl_set_agent_reports(experiment, agents_before = NULL, agents_after = NULL,
  agents_step = NULL, patches_before = NULL, patches_after = NULL)
```

Arguments

`experiment`	NetLogo experiment object
`agents_before`	A list of agent reports to be accessed before each run.
`agents_after`	A list of agent reports to be accessed after each run.
`agents_step`	A list of agent reports to be accessed per each iteration (step).
`patches_before`	A list of patches reports to be accessed before each run
`patches_after`	A list of patches reports to be accessed after each run

Value

NetLogo experiment object

See Also

To create an experiment object use `nl_experiment`

nl_set_measures *Set or change measures of existing NetLogo experiment*

Description

Set or change measures of existing NetLogo experiment

Usage

```
nl_set_measures(experiment, step = NULL, run = NULL, eval_criteria = NULL,
  eval_aggregate_fun = NULL, eval_mutate = NULL, as.data.frame = TRUE,
  step_transform = NULL)
```

Arguments

`experiment`	NetLogo experiment object
`step`	NetLogo reporters for each step (reported at every tick). A list of named character vectors. Use `measures` function to get the correct structure.
`run`	NetLogo reporters for each run (reported at end of run). A list of named character vectors. Use `measures` function to get the correct structure.
`eval_criteria`	A criteria calculation expressions. May use `step` or `run` data frames to calculate criteria. Elements from `step` should be aggregated. Must return named numeric vector.
`eval_aggregate_fun`	Aggregate criteria. It makes sense when when repetitions > 1
`eval_mutate`	Add criteria based on aggregated values
`as.data.frame`	Reporting in data frame format (TRUE by default)
`step_transform`	A function to transform data frame result from step reporters. When simulation has many steps and only summary data is needed, step_transform can reduce memory requirements to run experiment.

Details

Values of experiment measures are NetLogo reporters. Names of measures will be used in the resulting data frames as column names.

Value

NetLogo experiment object

See Also

To create an experiment object use `nl_experiment`

`nl_set_param_values` *Define parameter sets for NetLogo experiment*

Description

Define parameter sets for NetLogo experiment

Usage

```
nl_set_param_values(experiment, param_values = NULL, mapping = NULL)
```

Arguments

`experiment`	NetLogo experiment object from nl_experiment() function
`param_values`	A data.frame with parameter values or a list of values to be expanded to all combinations of values
`mapping`	Mapping between R and NetLogo parameters in named character vector. For example: c(diffusion_rate = "diffusion-rate", population = "population")

Value

NetLogo experiment object

`nl_set_run_options` *Set run options of a NetLogo experiment object*

Description

You can set basic run options when creating experiment object with `nl_experiment`. To change these or add additional options use `nl_set_run_options`

Usage

```
nl_set_run_options(experiment, random_seed = NULL, repetitions = 1,
  max_minutes = 10, setup_commands = "setup", go_command = "go",
  data_handler = NULL)
```

Arguments

`experiment`	NetLogo experiment object from nl_experiment() function
`random_seed`	Random seed
`repetitions`	Number of repetitions (when random seed is not defined)
`max_minutes`	If max.minutes > 0 the execution stops after the defined number of minutes (with an error and no return value) Default value is 10.
`setup_commands`	NetLogo command strings to execute to setup the model
`go_command`	NetLogo command string to execute the step in the model
`data_handler`	Function to handle observations. If handler is defined the observations will not be stored in result elements when running the experiment with ‘nl_run‘ function.

Value

NetLogo experiment object

Examples

```
experiment <- nl_experiment(
  model_file = "my_model.nlogo",
  while_condition = "any? turtles"
)

experiment <- nl_set_run_options(
  experiment,
  repetitions = 3,
  setup_commands = c("setup", "change_something")
)
```

`nl_show_params` *Plots parameters with scatter plots*

Description

Plots parameters with scatter plots

Usage

```
nl_show_params(experiment, cex = 0.7, col = "#000000CC",
  lower.panel = NULL, ...)
```

Arguments

`experiment`	Experiment object
`cex`	Parameter passed to pairs function
`col`	Parameter passed to pairs function
`lower.panel`	Parameter passed to pairs function
`...`	Parameters passed to pairs function

`nl_show_patches` *Plot multiple patches result*

Description

Plot patches from simualations result

Usage

```
nl_show_patches(result, x_param, y_param = NULL, fill = "pcolor",
  type = "patches_after", sub_type = NULL)
```

Arguments

`result`	NetLogo experiment result object
`x_param`	row parameter
`y_param`	column parameter
`fill`	variable to control the color (default is pcolor)
`type`	as type from nl_get_result (default is "patches_after)
`sub_type`	as sub_type from nl_get_result (optional - if not the first patches set)

`nl_show_step` *Plot step measure observations*

Description

Plot observations for each simulation step

Usage

```
nl_show_step(result, x = "step_id", y, color = "run_id", x_param = ".",
  y_param = ".", title = NULL, data_filter = NULL, alpha = 1)
```

Arguments

`result`	NetLogo experiment result object
`x`	"step_id" or measure name (as string) to choose for x axis
`y`	measure name as string to plot on y axis
`color`	by default it is based on "run_id" (simulation repetition). Change to `NA` to plot every repetition in black
`x_param`	which parameter to use for faceting horizontally
`y_param`	which parameter to use for faceting vertically
`title`	plot title
`data_filter`	optional subset expression (not quoted) using parameters, `run_id` and `step_id`
`alpha`	lines opacity

See Also

To get only data and create custom plots see `nl_get_result`

`nl_show_views_grid` *Show exported views images in a grid*

Description

Show exported views images in a grid

Usage

```
nl_show_views_grid(result, x_param = NULL, y_param = NULL, img_gap = 0.03)
```

Arguments

`result`	Result from `nl_run` function
`x_param`	Name of parameter on x axis
`y_param`	Name of parameter on y axis
`img_gap`	A gap between the images

`print.nl_experiment` *Print NetLogo experiment object*

Description

Print NetLogo experiment object

Usage

```
## S3 method for class 'nl_experiment'
print(x, ...)
```

Arguments

`x`	NetLogo experiment object
`...`	further arguments passed to or from other methods.

Index