Package 'logolink'

October 9, 2025

Title An Interface for Running 'NetLogo' Simulations

Version 0.1.0

Description An interface for 'NetLogo' <https://www.netlogo.org> that enables programmatic setup and execution of simulations. Designed to facilitate integrating 'NetLogo' models into reproducible workflows by creating and running 'BehaviorSpace' experiments and retrieving their results.

License GPL (>= 3)

URL https://danielvartan.github.io/logolink/, https://github.com/danielvartan/logolink/

BugReports https://github.com/danielvartan/logolink/issues/

Depends R (>= 4.4)

Imports checkmate (>= 2.3.3), cli (>= 3.6.5), dplyr (>= 1.1.4), fs (>= 1.6.6), glue (>= 1.8.0), janitor (>= 2.2.1), magrittr (>= 2.0.4), purrr (>= 1.1.0), readr (>= 2.1.5), stringr (>= 1.5.2), xml2 (>= 1.4.0)

Suggests bslib (>= 0.9.0), covr (>= 3.6.4), knitr (>= 1.50), spelling (>= 2.3.2), testthat (>= 3.2.3), tibble (>= 3.3.0),

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation no

Author Daniel Vartanian [aut, cre, ccp, cph] (ORCID: <https://orcid.org/0000-0001-7782-759X>)

Maintainer Daniel Vartanian <danielvartan@proton.me>

Repository CRAN

Date/Publication 2025-10-09 08:00:02 UTC

Contents

`create_experiment` *Create a NetLogo BehaviorSpace experiment XML file*

Description

`create_experiment()` creates a NetLogo BehaviorSpace experiment XML in a temporary file that can be used to run headless experiments with the `run_experiment()` function.

Please refer to the BehaviorSpace Guide for complete guidance on how to set and run experiments in NetLogo.

Usage

```
create_experiment(
  name = "",
  repetitions = 1,
  sequential_run_order = TRUE,
  run_metrics_every_step = FALSE,
  pre_experiment = NULL,
  setup = "setup",
  go = "go",
  post_run = NULL,
  post_experiment = NULL,
  time_limit = 1,
  exit_condition = NULL,
  metrics = c("count turtles", "count patches"),
  run_metrics_condition = NULL,
  constants = NULL
)
```

Arguments

`name`	(optional) A string specifying the name of the experiment (default: `""`).
`repetitions`	(optional) An integer number specifying the number of times to repeat the experiment (default: `1`).
`sequential_run_order`	(optional) A `logical` flag indicating whether to run the experiments in sequential order (default: `TRUE`).

`run_metrics_every_step`
: (optional) A `logical` flag indicating whether to record metrics at every step (default: `FALSE`).

`pre_experiment`
: (optional) A string specifying the NetLogo command to run before the experiment starts (default: `NULL`).

`setup`
: A string specifying the NetLogo command to set up the model (default: `"setup"`).

`go`
: A string specifying the NetLogo command to run the model (default: `"go"`).

`post_run`
: (optional) A string specifying the NetLogo command to run after each run (default: `NULL`).

`post_experiment`
: (optional) A string specifying the NetLogo command to run after the experiment ends (default: `NULL`).

`time_limit`
: (optional) An integer number specifying the maximum number of steps to run for each repetition (default: `1`).

`exit_condition`
: (optional) A string specifying the NetLogo command that defines the exit condition for the experiment (default: `NULL`).

`metrics`
: A `character` vector specifying the NetLogo commands to record as metrics (default: `c('count turtles', 'count patches')`).

`run_metrics_condition`
: (optional) A string specifying the NetLogo command that defines the condition to record metrics (default: `NULL`).

`constants`
: (optional) A named `list` specifying the constants to vary in the experiment. Each element can be either a single value (for enumerated values) or a `list` with `first`, `step`, and `last` elements (for stepped values). See the *Details* and *Examples* sections to learn more (default: `NULL`).

Details

Constants:

The `constants` argument allows you to specify the parameters to vary in the experiment. It should be a named `list` where each name corresponds to a NetLogo variable. The value for each name can be either:

- A single value (for enumerated values). For example, to set the variable `initial-number-of-turtles` to `10`, you would use `list("initial-number-of-turtles" = 10)`.
- A `list` with `first`, `step`, and `last` elements (for stepped values). For example, to vary the variable `initial-number-of-turtles` from `10` to `50` in steps of `10`, you would use `list("initial-number-of-turtles" = list(first = 10, step = 10, last = 50))`.

Please note that any mistake in the constants names will cause the experiment to return an empty result set. Be careful when changing them.

Also, enclose commands with single quotes (e.g., `'n-values 10 ["N/A"]'`), since NetLogo only accepts double quotes for strings.

Value

A string with the path to the created XML file.

See Also

Other NetLogo functions: `run_experiment()`

Examples

```
setup_file <- create_experiment(
  name = "Wolf Sheep Simple Model Analysis",
  repetitions = 10,
  sequential_run_order = TRUE,
  run_metrics_every_step = TRUE,
  setup = "setup",
  go = "go",
  time_limit = 1000,
  metrics = c(
    'count wolves',
    'count sheep'
  ),
  run_metrics_condition = NULL,
  constants = list(
    "number-of-sheep" = 500,
    "number-of-wolves" = list(
      first = 5,
      step = 1,
      last = 15
    ),
    "movement-cost" = 0.5,
    "grass-regrowth-rate" = 0.3,
    "energy-gain-from-grass" = 2,
    "energy-gain-from-sheep" = 5
  )
)

setup_file

setup_file |> inspect_experiment_file()
```

`inspect_experiment_file` *Inspect a BehaviorSpace experiment XML file*

Description

`inspect_experiment_file()` reads and prints the content of a BehaviorSpace experiment XML file. This is useful for debugging and understanding the structure of the experiment file.

Usage

```
inspect_experiment_file(file)
```

Arguments

file	A string specifying the path to the BehaviorSpace experiment XML file.

Value

An invisible NULL. This function is used for its side effect.

See Also

Other Utility functions: parse_netlogo_list()

Examples

```
file <- create_experiment(name = "My Experiment")

file |> inspect_experiment_file()
```

---

parse_netlogo_list *Parse NetLogo Lists*

Description

parse_netlogo_list() parses NetLogo-style lists represented as strings (e.g., "[1 2 3]") into R lists. It automatically detects numeric, integer, logical, and character types within the lists and converts them accordingly.

If the input does not contain NetLogo-style lists, it returns the original vector unchanged.

Usage

```
parse_netlogo_list(x)
```

Arguments

x	An atomic object potentially containing NetLogo-style lists.

Value

A list of parsed elements if the input contains NetLogo-style lists; otherwise, returns the original vector.

See Also

Other Utility functions: inspect_experiment_file()

Examples

```
# Scalar Examples -----

'["a" "b" "c"]' |> parse_netlogo_list()

'[1 2 3]' |> parse_netlogo_list()

'[1.1 2.1 3.1]' |> parse_netlogo_list()

'[true false true]' |> parse_netlogo_list()

# Vector Examples -----

c('["a" "b" "c"]', '["d" "e" "f"]') |> parse_netlogo_list()

c('[1 2 3]', '[4 5 6]') |> parse_netlogo_list()

c('[1.1 2.1 3.1]', '[4.1 5.1 6.1]') |> parse_netlogo_list()

c('[true false true]', '[false true false]') |> parse_netlogo_list()
```

---

run_experiment *Run a NetLogo BehaviorSpace experiment*

Description

`run_experiment()` runs a NetLogo BehaviorSpace experiment in headless mode and returns the results as a tidy data frame. It can be used with `create_experiment()` to create the experiment XML file on the fly, or with an existing experiment stored in the NetLogo model file.

Please refer to the BehaviorSpace Guide for complete guidance on how to set and run experiments in NetLogo.

Usage

```
run_experiment(
  netlogo_path,
  model_path,
  experiment = NULL,
  setup_file = NULL,
  other_arguments = NULL,
  parse = TRUE
)
```

Arguments

`netlogo_path`	A string specifying the path to the NetLogo executable. In Windows, this is usually something like `C:\Program Files\NetLogo 7.0.0\NetLogo.exe`.

`model_path` A string specifying the path to the NetLogo model file (with extension `.nlogo`, `.nlogo3d`, `.nlogox`, or `.nlogox3d`).

`experiment` (optional) A string specifying the name of the experiment defined in the NetLogo model file (default: `NULL`).

`setup_file` (optional) A string specifying the path to an XML file containing the experiment definition. This file can be created using `create_experiment()` or exported from the NetLogo BehaviorSpace interface (default: `NULL`).

`other_arguments` (optional) A `character` vector specifying any additional command-line arguments to pass to the NetLogo executable. For example, you can use `c("--threads 4")` to specify the number of threads to use (default: `NULL`).

`parse` (optional) A `logical` flag indicating whether to parse NetLogo lists in the output data frame. If `TRUE`, columns containing NetLogo lists (e.g., `[1 2 3]`) will be converted to R lists. If `FALSE`, the columns will remain as `character` strings (default: `TRUE`).

Value

A `tibble` containing the results of the experiment.

See Also

Other NetLogo functions: `create_experiment()`

Examples

```
# Set the Environment -----

## Change the path below to point to your NetLogo executable.
netlogo_path <- file.path("", "opt", "netlogo-7-0-0", "bin", "NetLogo")

## Change the path below to point to the 'Wolf Sheep Simple 5' NetLogo
## model file in the Model Library.
model_path <- file.path(
  "", "opt", "netlogo-7-0-0", "models", "IABM Textbook", "chapter 4",
  "Wolf Sheep Simple 5.nlogox"
)

# Using `create_experiment()` to Create the Experiment XML File -----

## Not run:
  setup_file <- create_experiment(
    name = "Wolf Sheep Simple Model Analysis",
    repetitions = 10,
    sequential_run_order = TRUE,
    run_metrics_every_step = TRUE,
    setup = "setup",
    go = "go",
    time_limit = 1000,
    metrics = c(
```

```
    'count wolves',
    'count sheep'
  ),
  run_metrics_condition = NULL,
  constants = list(
    "number-of-sheep" = 500,
    "number-of-wolves" = list(
      first = 5,
      step = 1,
      last = 15
    ),
    "movement-cost" = 0.5,
    "grass-regrowth-rate" = 0.3,
    "energy-gain-from-grass" = 2,
    "energy-gain-from-sheep" = 5
  )
)

run_experiment(
  netlogo_path = netlogo_path,
  model_path = model_path,
  setup_file = setup_file
)
## Expected output:
#> # A tibble: 110,110 × 10
#>   run_number number_of_sheep number_of_wolves movement_cost
#>        <dbl>           <dbl>            <dbl>         <dbl>
#>  1          3             500                5           0.5
#>  2          9             500                5           0.5
#>  3          4             500                5           0.5
#>  4          1             500                5           0.5
#>  5          6             500                5           0.5
#>  6          8             500                5           0.5
#>  7          7             500                5           0.5
#>  8          2             500                5           0.5
#>  9          5             500                5           0.5
#> 10          2             500                5           0.5
#>  # 110,100 more rows
#>  # 6 more variables: grass_regrowth_rate <dbl>,
#>  # energy_gain_from_grass <dbl>, energy_gain_from_sheep <dbl>,
#>  # step <dbl>, count_wolves <dbl>, count_sheep <dbl>
#>  # Use `print(n = ...)` to see more rows

## End(Not run)

# Using an Experiment Defined in the NetLogo Model File -----

## Not run:
  run_experiment(
    netlogo_path = netlogo_path,
    model_path = model_path,
    experiment = "Wolf Sheep Simple model analysis"
  )
```

```
## Expected output:
#> # A tibble: 110 × 11
#>    run_number energy_gain_from_grass number_of_wolves movement_cost
#>         <dbl>                  <dbl>            <dbl>         <dbl>
#>  1          4                      2                5           0.5
#>  2          8                      2                5           0.5
#>  3          2                      2                5           0.5
#>  4          9                      2                5           0.5
#>  5          1                      2                5           0.5
#>  6          5                      2                5           0.5
#>  7          7                      2                5           0.5
#>  8          3                      2                5           0.5
#>  9          6                      2                5           0.5
#> 10         12                      2                6           0.5
#> # 100 more rows
#> # 7 more variables: energy_gain_from_sheep <dbl>,
#> # number_of_sheep <dbl>, grass_regrowth_rate <dbl>, step <dbl>,
#> # count_wolves <dbl>, count_sheep <dbl>,
#> # sum_grass_amount_of_patches <dbl>
#> # Use `print(n = ...)` to see more rows

## End(Not run)
```

Index