Package ‘simlandr’

August 16, 2021

Type Package
Title Simulation-Based Landscape Construction for Dynamical Systems
Version 0.1.1
Description A set of tools for constructing potential landscapes for dynamical systems using Monte-Carlo simulation. Especially suitable for psychological formal models.
License GPL (>= 3)
Encoding UTF-8
Imports dplyr, magrittr, purrr, tibble, ggplot2, scales, MASS, plotly,
htmlwidgets, bigmemory, digest, methods, ks, gganimate,
forcats, rlang
RoxygenNote 7.1.1
URL https://github.com/Sciurus365/simlandr
BugReports https://github.com/Sciurus365/simlandr/issues
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Jingmeng Cui [aut, cre]
Maintainer Jingmeng Cui <jingmeng.cui@outlook.com>
Repository CRAN
Date/Publication 2021-08-16 07:50:02 UTC

R topics documented:

attach_all_matrices .............................................. 2
calculate_barrier ................................................. 3
calculate_barrier_2d ............................................. 4
calculate_barrier_2d_batch ..................................... 4
calculate_barrier_3d ............................................. 5
calculate_barrier_3d_batch ................................... 6
check_conv ....................................................... 7
attach_all_matrices

Attach all matrices in a batch simulation

Usage

attach_all_matrices(bs, backingpath = "bp")

Arguments

bs A batch_simulation object.
backingpath Passed to as.big.matrix.
calculate_barrier

General function for calculating energy barrier

Description

General function for calculating energy barrier

Usage

calculate_barrier(l, ...)

## S3 method for class '2d_density_landscape'
calculate_barrier(l, ...)

## S3 method for class 'density'
calculate_barrier(l, ...)

## S3 method for class '2d_density_landscape'
calculate_barrier(l, ...)

## S3 method for class '3d_static_landscape'
calculate_barrier(l, ...)

## S3 method for class 'list'
calculate_barrier(l, ...)

## S3 method for class 'barrier'
plot(x, ...)

## S3 method for class '3d_animation_landscape'
calculate_barrier(l, ...)

## S3 method for class '3d_matrix_landscape'
calculate_barrier(l, ...)

## S3 method for class '2d_matrix_landscape'
calculate_barrier(l, ...)

Arguments

l       A landscape or related project.
...
    Not in use.
x       A ‘barrier’ object.

Methods (by class)

- barrier: Plot a ‘barrier’
**calculate_barrier_2d_batch**

**See Also**

*calculate_barrier_2d, calculate_barrier_3d, calculate_barrier_3d_batch*

---

**calculate_barrier_2d  Calculate barrier from a 2D landscape**

**Description**

Calculate barrier from a 2D landscape

**Usage**

```r
calculate_barrier_2d(
  l,
  start_location_value = 0,
  start_r = 0.1, 
  end_location_value = 0.7,
  end_r = 0.15, 
  base = exp(1)
)
```

**Arguments**

- `l`: A `2d_density_landscape` object (recommended) or a density distribution.
- `start_location_value, end_location_value`: The initial position (in value) for searching the start/end point.
- `start_r, end_r`: The searching radius for searching the start/end point.
- `base`: The base of the log function.

---

**calculate_barrier_2d_batch  Calculate barrier from a 2D landscape with multiple simulations**

**Description**

Calculate barrier from a 2D landscape with multiple simulations
**Usage**

```r
calculate_barrier_2d_batch(
  l,
  bg = NULL,
  start_location_value = 0,
  start_r = 0.1,
  end_location_value = 0.7,
  end_r = 0.15,
  base = exp(1)
)
```

**Arguments**

- **l**: A `2d_animation_landscape` (not implemented yet) or a `2d_matrix_landscape`.
- **bg**: A `barrier_grid_3d` object if you want to use different parameters for each condition. Otherwise `NULL`.
- **start_location_value, end_location_value**: The initial position (in value) for searching the start/end point.
- **start_r, end_r**: The searching (L1) radius for searching the start/end point.
- **base**: The base of the log function.

---

**Description**

Calculate barrier from a 3D landscape

**Usage**

```r
calculate_barrier_3d(  
  l,
  start_location_value = c(0, 0),
  start_r = 0.1,
  end_location_value = c(0.7, 0.6),
  end_r = 0.15,
  Umax,
  expand = TRUE,
  omit_unstable = FALSE,
  base = exp(1)
)
```
calculate_barrier_3d_batch

Arguments

- `l`: A 3d_static_landscape object (recommended) or a kde2d distribution.
- `start_location_value, end_location_value`: The initial position (in value) for searching the start/end point.
- `start_r, end_r`: The searching (L1) radius for searching the start/end point.
- `Umax`: The highest possible value of the potential function.
- `expand`: If the values in the range all equal to Umax, expand the range or not?
- `omit_unstable`: If a state is not stable (the "local minimum" overlaps with the saddle point), omit that state or not?
- `base`: The base of the log function.

Description

Calculate barrier from a 3D landscape with multiple simulations

Usage

```r
calculate_barrier_3d_batch(
  l,
  bg = NULL,
  start_location_value = c(0, 0),
  start_r = 0.1,
  end_location_value = c(0.7, 0.6),
  end_r = 0.15,
  Umax,
  expand = TRUE,
  omit_unstable = FALSE,
  base = exp(1)
)
```

Arguments

- `l`: A 3d_static_landscape or a 3d_matrix_landscape.
- `bg`: A barrier_grid_3d object if you want to use different parameters for each condition. Otherwise NULL.
- `start_location_value, end_location_value`: The initial position (in value) for searching the start/end point.
- `start_r, end_r`: The searching (L1) radius for searching the start/end point.
- `Umax`: The highest possible value of the potential function.
check_conv

expand If the values in the range all equal to $U_{\text{max}}$, expand the range or not?
omit_unstable If a state is not stable (the "local minimum" overlaps with the saddle point), omit that state or not?
base The base of the log function.

check_conv Check density convergence of simulation

Description
Check density convergence of simulation

Usage
check_conv(output, vars, sample_perc = 0.2, plot_type = "bin")

Arguments
output A matrix of simulation output.
vars The names of variables to check.
sample_perc The percentage of data sample for the initial, middle, and final stage of the simulation.
plot_type Which type of plots should be generated? ("bin" or "density")

fill_in_struct Fill a vector of values into a structure list.

Description
Fill a vector of values into a structure list.

Usage
fill_in_struct(vec, struct)

Arguments
vec A vector of values.
struct A list with a certain structure.
find_local_min_3d

Find local minimum of a 3d distribution

Description
Find local minimum of a 3d distribution

Usage
find_local_min_3d(dist, localmin, r, Umax, expand = TRUE, first_called = TRUE)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dist</td>
<td>An kde2d distribution object.</td>
</tr>
<tr>
<td>localmin</td>
<td>Starting value of finding local minimum.</td>
</tr>
<tr>
<td>r</td>
<td>Searching (L1) radius.</td>
</tr>
<tr>
<td>Umax</td>
<td>The highest possible value of the potential function.</td>
</tr>
<tr>
<td>expand</td>
<td>If the values in the range all equal to $U_{max}$, expand the range or not?</td>
</tr>
<tr>
<td>first_called</td>
<td>Is this function first called by another function?</td>
</tr>
</tbody>
</table>
get_barrier_height

Get the barrier height from a 'barrier' object.

Description
Get the barrier height from a 'barrier' object.

Usage
get_barrier_height(b)

Arguments
b A barrier object.

get_dist
Get the probability distribution from a landscape object

Description
Get the probability distribution from a landscape object

Usage
get_dist(l, index = 1)

Arguments
l A landscape project.
index 1 to get the distribution in tidy format; 2 or "raw" to get the raw simulation result (batch_simulation).
Description

This layer can show the saddle point (2d) and the minimal path (3d) on the landscape.

Usage

get_geom(b, path = TRUE)

Arguments

b A barrier object.
path Show the lowest elevation path in the graph?

hash_big.matrix-class

Class "hash_big.matrix": big matrix with a md5 hash reference

Description

hash_big.matrix class is a modified class from big.matrix-class. Its purpose is to help users operate big matrices within hard disk in a reusable way, so that the large matrices do not consume too much memory, and the matrices can be reused for the next time. Comparing with big.matrix-class, the major enhancement of hash_big.matrix class is that the backing files are, by default, stored in a permanent place, with the md5 of the object as the file name. With this explicit name, hash_big.matrix objects can be easily reloaded into workspace every time.

Usage

as.hash_big.matrix(x, backingpath = "bp", silence = TRUE, ...)
attach.hash_big.matrix(x, backingpath = "bp")

Arguments

x A matrix, vector, or data.frame for as.big.matrix.
backingpath, ... Passed to as.big.matrix.
silence Suppress messages?
make_2d_density

Functions

- `as.hash_big.matrix`: Create a `hash_big.matrix` object from a matrix.
- `attach.hash_big.matrix`: Attach a `hash_big.matrix` object from the backing file to the workspace.

Slots

- `md5`: The md5 value of the matrix.
- `address`: Inherited from `big.matrix`.

---

**make_2d_density**  
*Make 2D density-based landscape plot for a single simulation output*

Description

Make 2D density-based landscape plot for a single simulation output

Usage

```r
make_2d_density(output, x, adjust = 50, from = -0.1, to = 1, Umax = 5)
```

Arguments

- `output`: A matrix of simulation output.
- `x`: The name of the target variable.
- `adjust, from, to`: Passed to `density`.
- `Umax`: The maximum displayed value of potential.

Value

A `2d_density_landscape` object.
make_2d_kernel_dist  Make 2D kernel smooth distribution

Description

Make 2D kernel smooth distribution

Usage

```r
make_2d_kernel_dist(
  output,
  x,
  y,
  n = 200,
  lims = c(-0.1, 1.1, -0.1, 1.1),
  h,
  kde_fun = "ks"
)
```

Arguments

- **output**: A matrix of simulation output.
- **x, y**: The name of the target variable.
- **n, lims, h**: Passed to `kde` or `kde2d`. If using `ks::kde`, `H = diag(h, 2, 2)`. Note: the definition of bandwidth ('h') is different in two functions. To get a similar output, the 'h' is about 50 to 5000 times smaller for `kde` than `kde2d`.
- **kde_fun**: Which to use? Choices: "ks" `ks::kde` (default; faster and taking less memory); "MASS" `MASS::kde2d`.

Value

A `kde2d`-type list.

make_2d_matrix  Make a matrix of 2d graphs for two parameters

Description

Make a matrix of 2d graphs for two parameters
Usage

```r
make_2d_matrix(
  bs,
  x,
  rows = NULL,
  cols,
  adjust = 50,
  from = -0.1,
  to = 1,
  Umax = 5,
  individual_landscape = FALSE
)
```

Arguments

- `bs` A `batch_simulation` object created by `batch_simulation`.
- `x, rows, cols` The names of the target variables. If `rows` is `NULL`, only a vector of graphs will be generated.
- `adjust, from, to` Passed to `density`.
- `Umax` The maximum displayed value of potential.
- `individual_landscape` Make individual landscape for each simulation?

Description

Make a tidy data frame from smooth 2d distribution matrix

Usage

```r
make_2d_tidy_dist(dist_2d, value = NULL, var_name = NULL)
```

Arguments

- `dist_2d` kde2d distribution.
- `value` The value of the variable of interest.
- `var_name` The name of the variable.

Value

A tidy data frame.
**make_3d_animation**  
*Make 3d animations from multiple simulations*

**Description**
Make 3d animations from multiple simulations

**Usage**
```r
make_3d_animation(
  bs,
  x, y, fr,
  Umax = 5,
  n = 200,
  lims = c(-0.1, 1.1, -0.1, 1.1),
  h = 0.001,
  kde_fun = "ks",
  individual_landscape = FALSE,
  mat_3d = TRUE
)
```

**Arguments**
- **bs**: A `batch_simulation` object created by `batch_simulation`.
- **x, y, fr**: The names of the target variables. `fr` corresponds to the `frame` parameter in plotly.
- **Umax**: The maximum displayed value of potential.
- **n, lims, h, kde_fun**: Passed to `make_2d_kernel_dist`
- **individual_landscape**: Make individual landscape for each simulation?
- **mat_3d**: Also make heatmap matrix?

**make_3d_kernel_dist**  
*Make 3D kernel smooth distribution*

**Description**
Make 3D kernel smooth distribution
Usage

```r
make_3d_kernel_dist(
  output,
  x,
  y,
  z,
  n = 200,
  lims = c(-0.1, 1.1, -0.1, 1.1, -0.1, 1.1),
  h
)
```

Arguments

- `output`: A matrix of simulation output.
- `x, y, z`: The name of the target variable.
- `n, lims, h`: Passed to `kde` (but using the format of `kde2d` to make it consistent across functions). For `ks::kde`, \( H = \text{diag}(h, 2, 2) \).

Value

A `MASS::kde2d`-type list.

---

`make_3d_matrix`  
Make a matrix or vector of 3d heatmap graphs for two parameters

Description

(Note: a matrix of interactive maps is currently not supported.)

Usage

```r
make_3d_matrix(
  bs,
  x,
  y,
  rows = NULL,
  cols,
  Umax = 5,
  n = 200,
  lims = c(-0.1, 1.1, -0.1, 1.1),
  h = 0.001,
  kde_fun = "ks",
  individual_landscape = FALSE
)
```
Arguments

bs  
A batch_simulation object created by batch_simulation.
x, y, rows, cols  
The names of the target variables. If rows is NULL, only a vector of graphs will be generated.

Umax  
The maximum displayed value of potential.
n, lims, h, kde_fun  
Passed to make_2d_kernel_dist

individual_landscape  
Make individual landscape for each simulation?

Description

Make 3D static landscape plots from simulation output

Usage

make_3d_static(
  output,
  x,
  y,
  Umax = 5,
  n = 200,
  lims = c(-0.1, 1.1, -0.1, 1.1),
  h = 0.001,
  kde_fun = "ks"
)

Arguments

output  
A matrix of simulation output.
x, y  
The name of the target variable.

Umax  
The maximum displayed value of potential.
n, lims, h, kde_fun  
Passed to make_2d_kernel_dist

Value

A 3d_static_landscape, landscape object.
make_3d_tidy_dist

**Description**

Make a tidy data frame from smooth 3d distribution matrix

**Usage**

```r
make_3d_tidy_dist(dist_3d, value = NULL, var_name = NULL)
```

**Arguments**

- `dist_3d` : kde2d-type distribution.
- `value` : The value of the variable of interest.
- `var_name` : The name of the variable.

**Value**

A tidy data frame.

make_4d_static

**Description**

Make 4D static space-color plots from simulation output

**Usage**

```r
make_4d_static(  
  output,  
  x,  
  y,  
  z,  
  Umax = 5,  
  n = 50,  
  lims = c(-0.1, 1.1, -0.1, 1.1, -0.1, 1.1),  
  h = 0.001
)
```
make_barrier_grid_2d

Arguments

- **output**: A matrix of simulation output.
- **x, y, z**: The name of the target variable.
- **Umax**: The maximum displayed value of potential.
- **n, lims, h**: Passed to `make_3d_kernel_dist`

Value

A `4d_static_landscape, landscape` object.

Description

Make a grid for calculating barriers for 2d landscapes

Usage

```r
make_barrier_grid_2d(
  vg,
  start_location_value = 0,
  start_r = 0.1,
  end_location_value = 0.7,
  end_r = 0.15,
  df = NULL,
  print_template = FALSE
)
```

Arguments

- **vg**: A `var_grid` object.
- **start_location_value, start_r, end_location_value, end_r**: Default values for finding local minimum. See `calculate_barrier_3d_batch`.
- **df**: A data frame for the variables. Use `print_template = TRUE` to get a template.
- **print_template**: Print a template for df.
make_barrier_grid_3d  Make a grid for calculating barriers for 3d landscapes

Description

Make a grid for calculating barriers for 3d landscapes

Usage

make_barrier_grid_3d(
  vg,
  start_location_value = c(0, 0),
  start_r = 0.1,
  end_location_value = c(0.7, 0.6),
  end_r = 0.15,
  df = NULL,
  print_template = FALSE
)

Arguments

vg  A var_grid object.
start_location_value, start_r, end_location_value, end_r
  Default values for finding local minimum. See calculate_barrier_3d_batch.
df  A data frame for the variables. Use print_template = TRUE to get a template.
print_template  Print a template for df.

make_var_grid  Make variable grids for batch simulation

Description

This is the main function for making the variable grids.

Usage

make_var_grid(var_set)

Arguments

var_set  A var_set object. See new_var_set and add_var.

Value

A var_grid object.
See Also

*batch_simulation* for a concrete example.

---

**modified_simulation**  *Do the batch simulation*

**Description**

This is the main function for the batch simulation.

**Usage**

```r
modified_simulation(sim_fun, var_list, default_list, bigmemory = TRUE)
```

```r
batch_simulation(var_grid, sim_fun, default_list, bigmemory = TRUE)
```

**Arguments**

- `sim_fun` The simulation function. See *sim_fun_test* for an example.
- `var_list` An `var_list` object generated by `fill_in_struct`.
- `default_list` A list of default values for `sim_fun`.
- `bigmemory` Use `hash_big.matrix-class` to store large matrices?
- `var_grid` A `var_grid` object. See `make_var_grid`.

**Value**

A `batch_simulation` object, also a data frame. The first column, `var`, is a list of `var_list` that contains all the variables; the second to the second last columns are the values of the variables; the last column is the output of the simulation function.

**Functions**

- `modified_simulation`: Modify a single simulation.

**Examples**

```r
test <- new_var_set()
test <- test %>%
  add_var("par1", "var1", 1, 2, 0.1) %>%
  add_var("par2", "var2", 1, 2, 0.1)
test_grid <- make_var_grid(test)
test_result <- batch_simulation(test_grid, sim_fun_test, 
  default_list = list(
    par1 = list(var1 = 0),
    par2 = list(var2 = 0, var3 = 0)
  ), bigmemory = FALSE
)
test_result
```
new_var_set

Create and modify variable sets for batch simulation

Description

A variable set contains the descriptions of the relevant variables in a batch simulation. Use new_var_set to create an S3 var_set object, and use add_var to add descriptions of variables.

Usage

new_var_set()

add_var(var_set, par_name, var_name, start, end, by)

nvar(var_set)

npar(var_set)

Arguments

var_set A var_set object.
par_name, var_name
The name of the parameter and variable in the simulation function
start, end, by The data points where you want to test the variables. Passed to seq.

Functions

• new_var_set: Create a var_set.
• add_var: Add a variable to the var_set.
• nvar: The number of variables.
• npar: The number of parameters.

See Also

make_var_grid for making grids from variable sets; batch_simulation for running batch simulation and a concrete example.

Examples

test <- new_var_set()
test <- test %>%
  add_var("par1", "var1", 1, 2, 0.1) %>%
  add_var("par2", "var2", 1, 2, 0.1)
plot.landscape  

Make plots from landscape objects

Description

Make plots from landscape objects

Usage

```r
## S3 method for class 'landscape'
plot(x, index = 1, ...)
```

Arguments

- `x`: A landscape object
- `index`: Default is 1. For some landscape objects, there is a second plot (usually 2d heatmaps for 3d landscapes) or a third plot (usually 3d matrices for 3d animations). Use `index = 2` to plot that one.
- `...`: Not in use.

print.batch_simulation

Print a batch_simulation

Description

Print a batch_simulation

Usage

```r
## S3 method for class 'batch_simulation'
print(x, detail = FALSE, ...)
```

Arguments

- `x`: The object.
- `detail`: Do you want to print the object details as a full list?
- `...`: Not in use.
print.check_conv

**Print a `check_conv`**

### Description

Print a `check_conv`

### Usage

```r
## S3 method for class 'check_conv'
print(x, ask = TRUE, ...)
```

### Arguments

- `x`: The object.
- `ask`: Ask to press enter to see the next plot?
- `...`: Not in use.

print.var_grid

**Print a var_grid**

### Description

Print a var_grid

### Usage

```r
## S3 method for class 'var_grid'
print(x, detail = FALSE, ...)
```

### Arguments

- `x`: The object.
- `detail`: Do you want to print the object details as a full list?
- `...`: Not in use.
print.var_set  
\textit{Print a var\_set}

\textbf{Description}

Print a var\_set

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'var\_set'
print(x, detail = FALSE, ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} The object.
  \item \texttt{detail} \hspace{1cm} Do you want to print the object details as a full list?
  \item \texttt{...} \hspace{1cm} Not in use.
\end{itemize}

reverselog_trans  
\textit{A function for reversed log transformation}

\textbf{Description}

A function for reversed log transformation

\textbf{Usage}

\begin{verbatim}
reverselog_trans(base = exp(1))
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{base} \hspace{1cm} The base of logarithm
\end{itemize}
save_landscape

Description
Save landscape plots

Usage
save_landscape(l, path = NULL, selfcontained = F, ...)

Arguments
- l: A landscape object
- path: The path to save the output. Default: "/pics/x_y.html".
- selfcontained: For plotly plots, save the output as a self-contained html file? Default: FALSE.
- ...: Other parameters passed to saveWidget or ggsave

sim_fun_test
A simple simulation function for testing

Description
A simple simulation function for testing

Usage
sim_fun_test(par1, par2, length = 1000)

Arguments
- par1, par2: Two parameters. par1 contains var1; par2 contains var2 and var3.
- length: The length of simulation.

Value
A matrix of simulation results.

See Also
batch_simulation for a concrete example.
Index

add_var, 19
add_var (new_var_set), 21
as.big.matrix, 2, 10
as.hash_big.matrix
  (hash_big.matrix-class), 10
attach.hash_big.matrix
  (hash_big.matrix-class), 10
attach_all_matrices, 2

batch_simulation, 2, 13, 14, 16, 20, 21, 25
batch_simulation (modified_simulation), 20

calculate_barrier, 3
calculate_barrier_2d, 4, 4
calculate_barrier_2d_batch, 4
calculate_barrier_3d, 4, 5
calculate_barrier_3d_batch, 4, 6, 18, 19
calculate_conv, 7

fill_in_struct, 7, 20
find_local_min_2d, 8
find_local_min_3d, 8

get_barrier_height, 9
gdist, 9
ggeom, 10
ggsave, 25

hash_big.matrix
  (hash_big.matrix-class), 10
hash_big.matrix-class, 10

kde, 12, 15
kde2d, 12, 15

make_2d_density, 11
make_2d_kernel_dist, 12, 16
make_2d_matrix, 12
make_2d_tidy_dist, 13
make_3d Animation, 14
make_3d kernel_dist, 14, 18
make_3d_matrix, 15
make_3d static, 16
make_3d tidy dist, 17
make_4d static, 17
make_barrier_grid_2d, 18
make_barrier_grid_3d, 19
make_var_grid, 19, 20, 21
modified_simulation, 20

new_var_set, 19, 21
np (new_var_set), 21
nvar (new_var_set), 21

plot.barrier (calculate_barrier), 3
plot. landscape, 22
print.batch_simulation, 22
print.check_conv, 23
print.var_grid, 23
print.var_set, 24

reverselog_trans, 24

save_landscape, 25
saveWidget, 25
sim_fun_test, 20, 25