Package: Isinglandr (via r-universe)

October 10, 2024

Type Package

Title Landscape Construction and Simulation for Ising Networks

Version 0.1.1.9000

Description A toolbox for constructing potential landscapes for Ising networks. The parameters of the networks can be directly supplied by users or estimated by the 'IsingFit' package by van Borkulo and Epskamp (2016) <https://CRAN.R-project.org/package=IsingFit> from empirical data. The Ising model's Boltzmann distribution is preserved for the potential landscape function. The landscape functions can be used for quantifying and visualizing the stability of network states, as well as visualizing the simulation process.

License GPL (>= 3)

URL https://sciurus365.github.io/Isinglandr/, https://github.com/Sciurus365/Isinglandr

BugReports https://github.com/Sciurus365/Isinglandr/issues

Depends R (>= 2.10)

Imports boot, boot.pval, broom, cli, dplyr, gganimate, ggplot2, glue, magrittr, methods, plotly, purrr, rlang, shiny, shinycssloaders, shinythemes, simlandr, tibble, tidyr

Suggests gifski, IsingFit, IsingSampler, transformr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://sciurus365.r-universe.dev

RemoteUrl https://github.com/sciurus365/isinglandr

RemoteRef HEAD

RemoteSha e7c711c843e2840400a49654676f37fc83ac5054

Contents

`autolayer.stability` *Get ggplot2 layers of stability metrics to add to the landscape plots*

Description

Those layers can show how the stability metrics are calculated on the landscape.

Usage

```
## S3 method for class 'stability_2d_Isingland'
autolayer(
  object,
  point = TRUE,
  line = TRUE,
  split_value = TRUE,
  interval = TRUE,
  stability_value = TRUE,
  ...
)

## S3 method for class 'stability_2d_Isingland_matrix'
autolayer(
  object,
  point = TRUE,
  line = TRUE,
  split_value = TRUE,
  interval = TRUE,
  stability_value = TRUE,
  ...
)
```

Arguments

`object`	A `stability` object calculated by `calculate_stability()` or `calculate_stability.2d_Isingland(`
`point, line, split_value, interval, stability_value`	Show those elements on the layer? Default is `TRUE` for all of them.
`...`	Not in use.

Value

a ggplot layer

`calculate_barrier.Isingland`
Calculate energy barrier for Ising landscapes

Description

Calculate energy barrier for Ising landscapes

Usage

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

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

## S3 method for class 'barrier_2d_Isingland'
print(x, simplify = FALSE, ...)

## S3 method for class 'barrier_2d_Isingland'
summary(object, ...)

## S3 method for class 'barrier_2d_Isingland_matrix'
summary(object, ...)
```

Arguments

`l`	An `Isingland` object constructed with `make_2d_Isingland()` or `make_2d_Isingland_matrix()`.
`...`	Not in use.
`x`	a result of the *default* method of `summary()`.
`simplify`	Print a simplified version of the output? Default is `FALSE`.
`object`	an object for which a summary is desired.

Value

A `barrier_Isingland` object that contains the following components:

- `shape` A character describing the shape of the landscape.
- `local_min_start`,`local_min_end`,`saddle_point` The positions of the two local minimums and the saddle point, described each by a list containing:
  - `U` The potential value.
  - `location`
    - `x_index` The row index in `get_dist(l)`.
    - `x_value` The number of active nodes.
- `delta_U_start`,`delta_U_end` The barrier heights for both sides.

Functions

- `summary(barrier_2d_Isingland)`: Return a vector of barrier heights.
- `summary(barrier_2d_Isingland_matrix)`: Return a tibble of barrier heights and conditions.

`calculate_stability` *Calculate the stability metrics for Ising landscapes*

Description

The stability is calculated based on the shape of the potential landscape and the prior knowledge about the qualitatively different parts of the system. Two stability indicators are calculated separately, and their difference is used to represent a general stability of the system in favor of the first phase. Within each phase, the potential difference between the local maximum and the local minimum (if multiple minimums exist, use the one that is further from the other phase; and the local maximum should always be on the side to the other phase) is used to represent the stability of this phase.

Usage

```
calculate_stability(l, ...)

## S3 method for class '`2d_Isingland`'
calculate_stability(l, split_value = 0.5 * l$Nvar, ...)

## S3 method for class '`2d_Isingland_matrix`'
calculate_stability(l, split_value = 0.5 * l$Nvar, ...)
```

Arguments

`l`	An `Isingland` object constructed with `make_2d_Isingland()` or `make_2d_Isingland_matrix()`.
`...`	Not in use.
`split_value`	An integer to specify the number of active nodes used to split two stability ranges. Default is half of the number of nodes.

Value

`calculate_stability.2d_Isingland()` Returns a `calculate_stability.2d_Isingland` project, which contains the following elements:

dist The distribution tibble which is the same as in the input `l`.

effective_minindex1,effective_maxindex1,effective_minindex2,effective_maxindex2 The (row)indices in `dist` that were used as the positions of the local minimums and maximums in two parts.

stability1,stability2,stability_diff The stability measures for the first (left) part, the second part (right), and their difference.

`calculate_stability.2d_Isingland_matrix()` Returns a `stability_2d_Isingland_matrix` object, which is a tibble containing columns of the varying parameters and a column `stability` of the `calculate_stability.2d_Isingland` objects for each landscape.

When `print()`ed, a verbal description of the stability metrics is shown. Use the `summary()` method for a tidy version of the outputs.

`calculate_stability_se` *Calculate the standard error, confidence interval, and p-value for the stability metrics of an Ising landscape using bootstrapping*

Description

Note that the BCa method is used for stability differences, and the percentile method is used for stability measures of individual phases because the stability measures of individual phases are highly zero-inflated and may crash the BCa estimation procedure. **The range estimation of the stability measures of individual phases should be interpreted with caution**.

Usage

```
calculate_stability_se(
  data,
  split_value = 0.5 * ncol(data),
  R = 1000,
  IsingFit_options = list(plot = FALSE),
  Isingland_options = list(),
  ...
)

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

compare_stability(
  data1,
  data2,
  split_value = 0.5 * ncol(data),
  R = 1000,
  IsingFit_options = list(plot = FALSE),
  Isingland_options = list(),
  ...
)
```

Arguments

data	A matrix of binary data
split_value	An integer to specify the number of active nodes used to split two stability ranges. Default is half of the number of nodes.
R	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case `R` would be a vector of integers where each component gives the number of resamples from each of the rows of weights.
IsingFit_options	Parameters passed to `IsingFit::IsingFit()`
Isingland_options	Parameters passed to `make_2d_Isingland()`
...	Parameters passed to `boot::boot()`
x	An object of class `stability_se`
data1, data2	Two matrices of binary data

Details

Use `calculate_stability_se()` for a single dataset, and use `compare_stability()` for comparing the stability metrics of two groups.

If you encounter the error message "Error in if (any(ints)) out[inds[ints]] <- tstar[k[inds[ints]]] : missing value where TRUE/FALSE needed", you may need to install the patched version of the `boot.pval` package with `pak::pkg_install("Sciurus365/boot.pval@patch-1")`. See https://github.com/mthulin/boot.pval/issues/4.

If you encounter the error message "estimated adjustment 'a' is NA", that probably means you should increase the number of bootstrap samples (`R`). See https://stats.stackexchange.com/questions/37918/why-is-the-error-estimated-adjustment-a-is-na-generated-from-r-boot-package.

References

Puth, M.-T., Neuhäuser, M., & Ruxton, G. D. (2015). On the variety of methods for calculating confidence intervals by bootstrapping. Journal of Animal Ecology, 84(4), 892–897. https://doi.org/10.1111/1365-2656.12382

`chain_simulate_Isingland`   *Make Ising chains from (a series of) Ising grid(s) and perform a chain simulation.*

Description

First specify what is the network parameter in each time points, then perform a chain simulation based on it. An Ising chain can be generated from one or more Ising grid(s) with one changing condition each.

Usage

```
chain_simulate_Isingland(
  Ising_chain,
  transform = FALSE,
  initial = 0,
  beta2 = NULL
)

make_Ising_chain(...)
```

Arguments

`Ising_chain`	An `Ising_chain` object generated from `make_Ising_chain()`.
`transform`	By default, this function considers the Ising network to use `-1` and `1` for two states. Set `transform = TRUE` if the Ising network uses `0` and `1` for two states, *which is often the case for the Ising networks estimated using* `IsingFit::IsingFit()`.
`initial`	An integer indicating the initial number of active nodes for the simulation. Float numbers will be converted to an integer automatically.
`beta2`	The $beta$ value used for simulation. By default use the same value as for landscape construction. Manually setting this value can make the system easier or more difficult to transition to another state, but will alter the steady-state distribution as well.
`...`	Ising grid(s) created by `make_Ising_grid()`.

Value

`make_Ising_chain` returns an `Ising_chain` object, which is a tibble, and each row represents a set of parameters for an Ising network.

`chain_simulate_Isingland` returns a `chain_sim_Isingland` object, which is a tibble containing the parameters, the landscape, and the number of active nodes for each time step.

`make_2d_Isingland` *Make a 2D landscape for an Ising network*

Description

Calculate the potential value $U(n)$ for each system state, represented by the number of active nodes n. The potential value is determined so that the Boltzmann distribution is preserved. The Boltzmann distribution is the basis and the steady-state distribution of all dynamic methods for Ising models, including those used in `IsingSampler::IsingSampler()` and Glauber dynamics. This means that if you assume the real-life system has the same steady-state distribution as the Boltzmann distribution of the Ising model, then possibility that their are n active nodes in the system is proportional to $e^{U(n)}$. Because of this property of $e^{U(n)}$, it is aligned with the potential landscape definition by Wang et al. (2008) and can quantitatively represent the stability of different system states.

Usage

```
make_2d_Isingland(thresholds, weiadj, beta = 1, transform = FALSE)
```

Arguments

`thresholds, weiadj`	The thresholds and the weighted adjacency matrix of the Ising network. If you have an `IsingFit` object estimated using `IsingFit::IsingFit()`, you can find those two parameters in its components (`<IsingFit>$thresholds` and `<IsingFit>$weiadj`).
`beta`	The β value for calculating the Hamiltonian.
`transform`	By default, this function considers the Ising network to use `-1` and `1` for two states. Set `transform = TRUE` if the Ising network uses `0` and `1` for two states, *which is often the case for the Ising networks estimated using* `IsingFit::IsingFit()`.

Details

The potential function $U(n)$ is calculated by the following equation:

$$U(n) = -\log(\sum_{v}^{a(v)=n} e^{-\beta H(v)})/\beta,$$

where v represent a specific activation state of the network, $a(v)$ is the number of active nodes for v, and H is the Hamiltonian function for Ising networks.

Value

A `2d_Isingland` object that contains the following components:

- `dist_raw`,`dist` Two tibbles containing the probability distribution and the potential values for different states.
- `thresholds`,`weiadj`,`beta` The parameters supplied to the function.
- `Nvar` The number of variables (nodes) in the Ising network.

References

Wang, J., Xu, L., & Wang, E. (2008). Potential landscape and flux framework of nonequilibrium networks: Robustness, dissipation, and coherence of biochemical oscillations. Proceedings of the National Academy of Sciences, 105(34), 12271-12276. https://doi.org/10.1073/pnas.0800579105 Sacha Epskamp (2020). IsingSampler: Sampling methods and distribution functions for the Ising model. R package version 0.2.1. https://CRAN.R-project.org/package=IsingSampler Glauber, R. J. (1963). Time-dependent statistics of the Ising model. Journal of Mathematical Physics, 4(2), 294-307. https://doi.org/10.1063/1.1703954

See Also

`make_3d_Isingland()` if you have two groups of nodes that you want to count the number of active ones separately.

Examples

```
Nvar <- 10
m <- rep(0, Nvar)
w <- matrix(0.1, Nvar, Nvar)
diag(w) <- 0
result1 <- make_2d_Isingland(m, w)
plot(result1)
```

`make_2d_Isingland_matrix` *Make a matrix of landscapes for multiple Ising networks*

Description

Make multiple landscapes together for different parameters.

Usage

```
make_2d_Isingland_matrix(Ising_grid, transform = FALSE)
```

Arguments

`Ising_grid`	Parameter values generated by `make_Ising_grid()`.
`transform`	By default, this function considers the Ising network to use `-1` and `1` for two states. Set `transform = TRUE` if the Ising network uses `0` and `1` for two states, *which is often the case for the Ising networks estimated using* `IsingFit::IsingFit()`.

Value

A `2d_Isingland_matrix` object that contains the following components:

- `dist_raw`,`dist` Two tibbles containing the probability distribution and the potential values for different states.
- `Nvar` The number of variables (nodes) in the Ising network.

Examples

```
Nvar <- 10
m <- rep(0, Nvar)
w <- matrix(0.1, Nvar, Nvar)
diag(w) <- 0
result4 <- make_Ising_grid(
  all_thresholds(seq(-0.1, 0.1, 0.1), .f = `+`),
  whole_weiadj(seq(0.5, 1.5, 0.5)),
  m, w
) %>% make_2d_Isingland_matrix()
plot(result4)
```

`make_3d_Isingland` *Make a 3D landscape for an Ising network*

Description

Similar to `make_2d_Isingland()`, but two categories of nodes can be specified so the number of active nodes can be calculated separately.

Usage

```
make_3d_Isingland(thresholds, weiadj, x, y, beta = 1, transform = FALSE)
```

Arguments

`thresholds, weiadj`	The thresholds and the weighted adjacency matrix of the Ising network. If you have an `IsingFit` object estimated using `IsingFit::IsingFit()`, you can find those two parameters in its components (`<IsingFit>$thresholds` and `<IsingFit>$weiadj`).
`x, y`	Two vectors specifying the indices or the names of the nodes for two categories. If they are character vectors, the names should match the row names of the `thresholds` matrix.
`beta`	The β value for calculating the Hamiltonian.
`transform`	By default, this function considers the Ising network to use `-1` and `1` for two states. Set `transform = TRUE` if the Ising network uses `0` and `1` for two states, *which is often the case for the Ising networks estimated using* `IsingFit::IsingFit()`.

Value

A `3d_Isingland` object that contains the following components:

- `dist_raw,dist` Two tibbles containing the probability distribution and the potential values for different states.
- `thresholds,weiadj,beta` The parameters supplied to the function.
- `Nvar` The number of variables (nodes) in the Ising network.

See Also

`make_2d_Isingland()` for the algorithm.

`make_Ising_grid` *Make a Grid to Specify Multiple Ising Networks*

Description

Specify one or two varying parameters for Ising networks. The output of `make_Ising_grid()` can be used to make landscapes of multiple networks.

Usage

```
make_Ising_grid(par1, par2 = NULL, thresholds, weiadj, beta = 1)
```

Arguments

`par1, par2`	Generated from one of `single_threshold()`, `all_thresholds()`, `single_wei()`, `[whole_weiadj()]` = NULL‘ if you only want to vary one parameter.
`thresholds, weiadj`	The thresholds and the weighted adjacency matrix of the Ising network. If you have an `IsingFit` object estimated using `IsingFit::IsingFit()`, you can find those two parameters in its components (`<IsingFit>$thresholds` and `<IsingFit>$weiadj`).
`beta`	The β value for calculating the Hamiltonian.

Details

There are five possible ways to vary the parameters for Ising networks, corresponding to five control functions:

- `single_threshold()` Vary a threshold value for a single variable.
- `all_thresholds()` Vary all threshold values together.
- `single_wei()` Vary a single weight value for a path between two variables.
- `whole_weiadj()` Vary the whole weighted adjacency matrix.
- `beta_list()` Use a list of different beta values.

See make_Ising_grid-control-functions for details.

Value

An `Ising_grid` object that is based on a tibble and contains the information of all simulation conditions.

make_Ising_grid-control-functions

Control Functions to Specify the Varying Parameters for an Ising Grid.

Description

Control Functions to Specify the Varying Parameters for an Ising Grid.

Usage

```
single_threshold(pos, seq, .f = `*`)

single_wei(pos, seq, .f = `*`)

all_thresholds(seq, .f = `*`)

whole_weiadj(seq, .f = `*`)

beta_list(seq, .f = `*`)
```

Arguments

pos	The position of the single threshold or the weight value that should vary across Ising networks. Should be a single number for `single_threshold()` or a numeric vector of length 2 for `single_wei()`.
seq	A vector that specify the values. Can be generated with `base::seq()`.
.f	What calculation should be done for `seq` and the original threshold value(s) or the original weight(ed adjacency matrix)? `*` by default, which means the values supplied in `seq` will be multiplied to the original value, vector, or matrix.

Value

An `ctrl_*` object specifying the varying parameters.

MDDNetwork

Estimation data for the Ising network of major depressive disorder

Description

Estimation data for the Ising network of major depressive disorder

Usage

```
MDDConnectivity

MDDThresholds
```

Format

An object of class `matrix` (inherits from `array`) with 9 rows and 9 columns.

An object of class `numeric` of length 9.

Functions

- `MDDConnectivity`: The connectivity strength of the network, represented in a weighted adjacency matrix.
- `MDDThresholds`: The thresholds of the network nodes, represented in a vector.

Source

`https://figshare.com/projects/Major_depression_as_a_complex_dynamic_system_accepted_for_publication_in_PLoS_ONE_/17360`

`shiny_Isingland_MDD` *A shiny app that shows the landscape for the Ising network of major depressive disorder*

Description

A shiny app that shows the landscape for the Ising network of major depressive disorder

Usage

```
shiny_Isingland_MDD(...)
```

Arguments

`...`	Not in use.

Value

This function opens a Shiny app session without a return value.

simulate_Isingland *Simulate a 2D Ising landscape*

Description

Perform a numeric simulation using the landscape. The simulation is performed using a similar algorithm as Glauber dynamics, that the transition possibility is determined by the difference in the potential function, and the steady-state distribution is the same as the Boltzmann distribution (if not setting `beta2`). Note that, the conditional transition possibility of this simulation may be different from Glauber dynamics or other simulation methods.

Usage

```
simulate_Isingland(l, ...)

## S3 method for class '`2d_Isingland`'
simulate_Isingland(
  l,
  mode = "single",
  initial = 0,
  length = 100,
  beta2 = l$beta,
  ...
)

## S3 method for class '`2d_Isingland_matrix`'
simulate_Isingland(
  l,
  mode = "single",
  initial = 0,
  length = 100,
  beta2 = NULL,
  ...
)
```

Arguments

`l`	An `Isingland` object constructed with `make_2d_Isingland()` or `make_2d_Isingland_matrix()`.
`...`	Not in use.
`mode`	One of `"single"`, `"distribution"`. Do you want to simulate the state of a single system stochastically or simulate the distribution of the states? `"single"` is used by default.
`initial`	An integer indicating the initial number of active nodes for the simulation. Float numbers will be converted to an integer automatically.
`length`	An integer indicating the simulation length.

`beta2`	The $beta$ value used for simulation. By default use the same value as for landscape construction. Manually setting this value can make the system easier or more difficult to transition to another state, but will alter the steady-state distribution as well.

Details

In each simulation step, the system can have one more active node, one less active node, or the same number of active nodes (if possible; so if all nodes are already active then it is not possible to have one more active node). The possibility of the three cases is determined by their potential function:

$$P(n_t = b | n_{t-1} = a) = \frac{e^{-\beta U(b)}}{\sum_{i \in \{a-1, a, a+1\}, 0 \le i \le N} e^{-\beta U(i)}}, \text{ if } b \in \{a-1, a, a+1\} \;\&\; 0 \le i \le N; 0, \text{otherwise},$$

where n_t is the number of active nodes at the time t, and $U(n)$ is the potential function.

Value

A `sim_Isingland` object with the following components:

- `output` A tibble of the simulation output.
- `landscape` The landscape object supplied to this function.
- `mode` A character representing the mode of the simulation.

Examples

```
Nvar <- 10
m <- rep(0, Nvar)
w <- matrix(0.1, Nvar, Nvar)
diag(w) <- 0
result1 <- make_2d_Isingland(m, w)
plot(result1)

set.seed(1614)
sim1 <- simulate_Isingland(result1, initial = 5)
plot(sim1)
```

Index