Package ‘paradigmEvo2023’

June 18, 2024

Type Package

Title Models of paradigm evolution via the PCFP

Version 0.1.1

Maintainer `<e.round@surrey.ac.uk>`

Description Modelling the evolution of morphological paradigms with associative and dissociative evidence.

Imports dplyr,
ggplot2,
gganimate,
matrixStats,
scales,
stats,
stringr

License CC BY 4.0

Encoding UTF-8

LazyData false

Collate 'evolve.R'
'initialisation.R'
'matrix_tools.R'
'mplat_tools.R'
'plotting.R'
'stats.R'

RoxygenNote 7.2.3

Contents

`add_stats` *Add statistics to an evolution*

Description

Add statistics to an evolution

Usage

```
add_stats(
  evolution,
  step_method = c("generation", "change"),
  skip_steps = 20,
  entropy_classwise = FALSE
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`step_method`	A string, specifying the kind of steps to take through the evolution: by generation or by change.
`skip_steps`	An integer, specifying how many steps to skip ahead each time.

Value

A list, the evolution including the stats

`cluster_sort_mat` *Cluster-sort the rows of a matrix*

Description

Cluster-sort the rows of a matrix

Usage

```
cluster_sort_mat(m)
```

Arguments

`m`	A matrix

Value

A matrix

`cluster_sort_order` *Cluster order*

Description

Cluster order

Usage

```
cluster_sort_order(m)
```

Arguments

`m`	A matrix

Value

A vector of integers, the sort order of rows.

`compile_mean_hi_lo` *Get mean and ribbon edges for plotting, for the column "value"*

Description

Get mean and ribbon edges for plotting, for the column "value"

Usage

```
compile_mean_hi_lo(df)
```

Arguments

`df`	A dataframe with columns generation and value

Value

A dataframe with columns generation, upper, lower & mean_value

`conflate_and_sum_wt` *Merge matrix rows that are identical outside of the first column; sum the first column when rows are merged*

Description

Merge matrix rows that are identical outside of the first column; sum the first column when rows are merged

Usage

```
conflate_and_sum_wt(m, w)
```

Arguments

`m`	A matrix
`w`	A vector of weights

Value

A list, containing $matrix and $weight

`count_arrangements` *Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; and cell (1,1) is filled.*

Description

Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; and cell (1,1) is filled.

Usage

```
count_arrangements(m, n, p)
```

Arguments

`m`	An integer. The number of grid rows.
`n`	An integer. The number of grid columns.
`p`	An integer. The number of stones

Value

An integer. The number of solutions.

`count_arrangements_1j` *Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (1,j) must also be filled, for some j > 1. This indicates the how many times a cell (1,j) is filled, among all the solutions to the grid-filling constraints described for ‘count_arrangements()‘.*

Description

Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (1,j) must also be filled, for some j > 1. This indicates the how many times a cell (1,j) is filled, among all the solutions to the grid-filling constraints described for ‘count_arrangements()‘.

Usage

```
count_arrangements_1j(m, n, p)
```

Arguments

`m`	An integer. The number of grid rows.
`n`	An integer. The number of grid columns.
`p`	An integer. The number of stones

Value

An integer. The number of solutions.

`count_arrangements_i1` *Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (i,1) must also be filled, for some i > 1. This indicates the how many times a cell (i,1) is filled, among all the solutions to the grid-filling constraints described for ‘count_arrangements()‘.*

Description

Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (i,1) must also be filled, for some i > 1. This indicates the how many times a cell (i,1) is filled, among all the solutions to the grid-filling constraints described for ‘count_arrangements()‘.

Usage

```
count_arrangements_i1(m, n, p)
```

Arguments

`m`	An integer. The number of grid rows.
`n`	An integer. The number of grid columns.
`p`	An integer. The number of stones

Value

An integer. The number of solutions.

`count_arrangements_ij` *Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (i,j) must also be filled, for some i > 1, j > 1. This indicates the how many times a cell (i,j) is filled, among all the solutions to the grid-filling constraints described for 'count_arrangements()'.*

Description

Count the number of ways to place p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; cell (1,1) is filled; and cell (i,j) must also be filled, for some i > 1, j > 1. This indicates the how many times a cell (i,j) is filled, among all the solutions to the grid-filling constraints described for 'count_arrangements()'.

Usage

```
count_arrangements_ij(m, n, p)
```

Arguments

`m`	An integer. The number of grid rows.
`n`	An integer. The number of grid columns.
`p`	An integer. The number of stones

Value

An integer. The number of solutions.

count_exponents *Count the number of unique values in rows or columns*

Description

Count the number of unique values in rows or columns

Usage

```
count_exponents(m)
```

Arguments

m	A matrix
p_type	A string

Value

A vector of counts

evolve_mplat *Simulate mplat evolution*

Description

Simulate mplat evolution

Usage

```
evolve_mplat(
  x = NULL,
  param_dict = mplat_params(),
  pivot_type = c("morphosite", "lexeme"),
  foci_lex_selector = c("uniform", "zipf"),
  foci_morphosite_selector = c("uniform", "zipf"),
  n_foci = 1,
  pivot_weighting = c("uniform", "zipf", "classwise"),
  pivot_use_proportion = 0,
  evidence_weighting = c("uniform", "zipf", "classwise"),
  evidence_use_proportion = 1,
  evidence_balance = 1/(neg_evidence_coeff + 1),
  neg_evidence_coeff = 0,
  change_classwise = FALSE,
  halt_method = c("n_generations", "quasi_stability"),
  halt_n = 5000,
  n_rep = 1
)
```

Arguments

`x`	Either a list or a matrix. If a matrix, x is interpreted as an mplat representing the initial state of the inflectional system; if a list, then a set of parameters to generate that state using initialise_mplat().
`pivot_type`	A string, giving the model type: morphosite pivot (as in A&M) or lexeme pivot (as in Esher 2015)
`foci_lex_selector`	A string, giving the method for selecting the foci lexemes, which change.
`foci_morphosite_selector`	A string, giving the method for selecting the foci morphosites, which change.
`n_foci`	An integer, giving the number of foci to change in a single generation.
`pivot_weighting`	A string, giving the method for weighting pivots; used either for sampling them, or weighting them when all are taken into account.
`pivot_use_proportion`	An numeric, giving the proportion of the available pivots to use. If set to 0, then just one pivot is used.
`evidence_weighting`	A string, giving the method for weighting evidence morphosites; used either for sampling them, or weighting them when all are taken into account.
`evidence_use_proportion`	An numeric, giving the proportion of the available evidence morphosites to use. If set to 0, just one is used.
`evidence_balance`	A numeric, between 0 and 1, giving the balance of negative evidence (max when 0) and positive evidence (max when 1).
`neg_evidence_coeff`	A non-negative numeric, an alternative parameterisation of evidence balance, giving negative evidence strength as a multiple of positive evidence.
`change_classwise`	A logical, whether to change whole classes at once.
`halt_method`	A string, giving the method for choosing when to halt the simulation.
`halt_n`	An integer, indicating either the total number of generations to evolve through or the number of consecutive generations of quasi-stability before halting.
`n_rep`	An integer, the number of repetitions of the simulation to perform

Value

A list, containing: containing: mplat_0; total_generations; a list of the model parameters; a list of final mplats; a dataframe of changes (with columns generation, foci_lex, foci_morphosite, a_new); a dataframe of statistics.

`generate_arrangements`	*Generate the solutions to placing p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; and cell (1,1) is filled.*

Description

Generate the solutions to placing p stones in an m x n grid, under the constraints that no row is empty; no column is empty; only one stone per cell; and cell (1,1) is filled.

Usage

```
generate_arrangements(m, n, p)
```

Arguments

`m`	An integer. The number of grid rows.
`n`	An integer. The number of grid columns.
`p`	An integer. The number of stones

Value

A list of matrices. The solutions, where 1 = filled cell.

`get_colours`	*Choose n brewer.pal colours, but allow n < 4, in which case choose more saturated end of the n=4 set*

Description

Choose n brewer.pal colours, but allow n < 4, in which case choose more saturated end of the n=4 set

Usage

```
get_colours(n, col)
```

Arguments

`n`	An integer, how many colours
`col`	A string, the palette to choose from

Value

A vector of hex colour specifications

`get_row_joint_H` *Get pairwise joint entropies of rows*

Description

Get pairwise joint entropies of rows

Usage

```
get_row_joint_H(m)
```

Arguments

`m`	A matrix

Value

A matrix of numerics, the joint entropies.

`get_steps` *Get step rows and generations*

Description

Get step rows and generations

Usage

```
get_steps(
  evolution,
  repetition = 1,
  step_method = c("generation", "change"),
  skip_steps
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`repetition`	An integer. Which repetition to unpack.
`step_method`	A string, specifying the kind of steps to take through the evolution: by generation or by change.
`skip_steps`	An integer, specifying how many steps to skip ahead each time.

Value

A list.

`get_weights` *Uniform and zipf distribution*

Description

Uniform and zipf distribution

Usage

```
get_weights(dist_type, n)
```

Arguments

`dist_type`	A string, giving the distribution type
`n`	An integer, the number of data points

Value

A vector of numerics of length n

`initialise_mplat` *Initialise a parametrically random mplat*

Description

Initialise a parametrically random mplat

Usage

```
initialise_mplat(
  x = 50,
  polymorphies = rep(6, times = 8),
  allele_distribution = c("uniform", "zipfian"),
  class_distribution = c("uniform", "zipfian"),
  col_class_distribution = c("uniform", "zipfian"),
  max_classes = NULL,
  max_col_classes = NULL
)
```

Arguments

`x`	Either an integer or a list. If a list, x is treated a list of parameters, else as the number of lexemes.
`polymorphies`	A vector of integers: the number of alleles for each morphosite.
`allele_distribution`	A numeric, giving the exponent of a powerlaw distribution for skewing the sampling from a morphosite's alleles. A value of 0 is uniform; a value of 1 is Zipfian.

`class_distribution`	A numeric, giving the exponent of a powerlaw distribution for skewing the repetitive sampling of lexical classes.
`col_class_distribution`	A numeric, giving the exponent of a powerlaw distribution for skewing the repetitive sampling of morphosites, i.e., having multiple morphosites repeating the same pattern.
`max_classes`	An integer, placing an upper bound on the number of distinct classes sampled when class_distribution is not zero.
`max_col_classes`	An integer, placing an upper bound on the number of distinct morphosites sampled when col_class_distribution is not zero.

Value

A matrix, the mplat.

`loglike2_trans` *A log-like transformation for data that goes to 0*

Description

A log-like transformation for data that goes to 0

Usage

```
loglike2_trans()
```

`loglike_trans` *A log-like transformation for data that goes to 0*

Description

A log-like transformation for data that goes to 0

Usage

```
loglike_trans()
```

`match_rows` *Which rows in a matrix are identical to a given vector*

Description

Which rows in a matrix are identical to a given vector

Usage

```
match_rows(m, v)
```

Arguments

`m`	A matrix
`v`	A vector, of length ncol(m)

Value

A vector of row indices

`match_sorted_rows` *Which rows in matrix 1 are in matrix 2 Only works for sorted matrices!*

Description

Which rows in matrix 1 are in matrix 2 Only works for sorted matrices!

Usage

```
match_sorted_rows(m1, m2)
```

Arguments

`m1`	A matrix
`m2`	A martix

Value

A vector of row indices

`mean_ignoring_diag` *Mean without diagonal*

Description

Mean without diagonal

Usage

```
mean_ignoring_diag(m)
```

Arguments

`m`	A matrix

`mplat_params` *Make a list of mplat parameters*

Description

Make a list of mplat parameters

Usage

```
mplat_params(
  n_lexemes = 50,
  polymorphies = rep(6, times = 8),
  allele_distribution = "uniform",
  class_distribution = "uniform",
  col_class_distribution = "uniform",
  max_classes = NULL,
  max_col_classes = NULL
)
```

Value

A list, of parameters for ‘initialise_mplat()‘

`multicolourise` *Add multiple colours*

Description

Add multiple colours

Usage

```
multicolourise(data, axis = c("columns", "rows"))
```

Arguments

`data`	A matrix, giving an mplat or a dataframe giving a long form evo_df.
`axis`	A string, indicating which axis to apply multicolour to.

Value

A list, containing the modified mplat and a vector of colours.

`pivot_mplat_longer` *Change mplat to a long format dataframe*

Description

Change mplat to a long format dataframe

Usage

```
pivot_mplat_longer(mplat)
```

Arguments

`mplat`	A matrix

Value

A dataframe with columns lexeme, morphosite, allele

plot_class_contrasts *Plot class contrast stats*

Description

Plot class contrast stats

Usage

```
plot_class_contrasts(
  evolution,
  y_limits = c(1, max(evolution$stats$n_classes)),
  y_breaks = c(1, 5, 20, 50, 100),
  step = c("generation", "change")
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`y_limits`	A vector of two numerics. The y axis limits.
`y_breaks`	A vector of numerics. The y axis tick mark locations.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

plot_evolution *Animate the evolution of a mplat*

Description

Animate the evolution of a mplat

Usage

```
plot_evolution(
  evolution,
  which_rep = 1,
  step_method = c("generation", "change"),
  skip_steps = if (substr(step_method[1], 1, 1) == "g") {

    evolution$total_generations/100
 } else {
     nrow(evolution$change)/100
 },
  telic_order = FALSE,
  multicolour = FALSE,
  multicolour_axis = c("columns", "rows"),
  fps = 10
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`which_rep`	An integer, specifying which repetition to plot.
`step_method`	A string, specifying the kind of steps to take through the evolution: by generation or by change.
`skip_steps`	An integer, specifying how many steps to skip ahead each time.
`telic_order`	A logical, Whether to order columns and rows so they cluster at the final generation.
`multicolour`	A logical. Whether to put column/rows in different colours.
`multicolour_axis`	A string, which axis to apply multicolour to.
`fps`	An integer

Value

A dataframe of the evolution in long form.

`plot_exp_contrasts` *Plot exponent contrast stats*

Description

Plot exponent contrast stats

Usage

```
plot_exp_contrasts(
  evolution,
  y_limits = c(1, 5),
  y_breaks = c(1, 1.5, 2, 3, 4, 5),
  step = c("generation", "change")
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`y_limits`	A vector of two numerics. The y axis limits.
`y_breaks`	A vector of numerics. The y axis tick mark locations.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

`plot_H` *Plot entropy stats*

Description

Plot entropy stats

Usage

```
plot_H(
  evolution,
  y_limits = c(0, 3),
  y_breaks = c(0, 0.05, 0.2, 0.5, 1, 3),
  step = c("generation", "change")
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`y_limits`	A vector of two numerics. The y axis limits.
`y_breaks`	A vector of numerics. The y axis tick mark locations.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

`plot_mplat` *Plot an mplat*

Description

Plot an mplat

Usage

```
plot_mplat(
  x,
  cluster_lexemes = FALSE,
  cluster_morphosites = FALSE,
  multicolour = FALSE,
  multicolour_axis = c("columns", "rows")
)
```

Arguments

`x`	A matrix or a list. If a list, it is used as parameters for initialise_mplat(); if a matrix, then it is an mplot.
`cluster_lexemes`	A logical. Whether to sort the lexemes by similarity.

`cluster_morphosites`	A logical. Whether to sort the morphosites by similarity.
`multicolour`	A logical. Whether to put column/rows in different colours.
`multicolour_axis`	A string, which axis to apply multicolour to.

Value

A ggplot raster plot.

`plot_stats` *Plot stats*

Description

Plot stats

Usage

```
plot_stats(
  evolution,
  suppress_top2 = FALSE,
  turnover_limits = c(0, 200),
  turnover_breaks = c(0, 2, 8, 20, 50, 100, 200),
  classes_limits = c(1, 100),
  classes_breaks = c(1, 2, 4, 8, 20, 50, 100)
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`suppress_top2`	A logical, whether to suppress the plotting of the top 2 classes
`turnover_limits`	A vector of two numerics. The y axis limits for the plot of turnover
`turnover_breaks`	A vector of numerics. The y axis tick mark locations for the plot of turnover
`classes_limits`	A vector of two numerics. The y axis limits for the plot of classes
`classes_breaks`	A vector of numerics. The y axis tick mark locations for the plot of classes

`plot_top2classes` *Plot top two classes stats*

Description

Plot top two classes stats

Usage

```
plot_top2classes(evolution, step = c("generation", "change"))
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

`plot_turnover` *Plot class turnover stats*

Description

Plot class turnover stats

Usage

```
plot_turnover(
  evolution,
  y_limits = c(0, 200),
  y_breaks = c(0, 2, 5, 20, 50, 100, 200),
  step = c("generation", "change")
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`y_limits`	A vector of two numerics. The y axis limits.
`y_breaks`	A vector of numerics. The y axis tick mark locations.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

plot_U *Plot entropy stats*

Description

Plot entropy stats

Usage

```
plot_U(
  evolution,
  y_limits = c(0, 1),
  y_breaks = c(0, 0.05, 0.2, 0.5, 1),
  step = c("generation", "change")
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`y_limits`	A vector of two numerics. The y axis limits.
`y_breaks`	A vector of numerics. The y axis tick mark locations.
`step`	A string, specifying the kind of steps to take through the evolution: by generation or by change.

power_dist *Powerlaw distribution*

Description

Powerlaw distribution

Usage

```
power_dist(n, exponent)
```

Arguments

`n`	An integer, the number of items
`exponent`	A numeric

`repeat_row` *Make a matrix of a repeated row*

Description

Make a matrix of a repeated row

Usage

```
repeat_row(row, n)
```

Arguments

`row`	A vector
`n`	An integer

Value

A matrix

`replicate_evo` *Evaluate an expression n times, returning results in a list.*

Description

Evaluate an expression n times, returning results in a list.

Usage

```
replicate_evo(n, expr)
```

Arguments

`n`	An integer
`expr`	An expression

Value

A list

`sample_power_dist` *Powerlaw sample*

Description

Powerlaw sample

Usage

```
sample_power_dist(n, size, exponent)
```

Arguments

`n`	An interger, the number of levels to sample from, with replacement
`size`	The sample size
`exponent`	A numeric

`scalar_mult_cols` *Multiply each column by an amount*

Description

Multiply each column by an amount

Usage

```
scalar_mult_cols(m, x)
```

Arguments

`m`	A matrix
`x`	A vector length ncol(m)

Value

A matrix

`scalar_mult_rows` *Multiply each row by an amount*

Description

Multiply each row by an amount

Usage

```
scalar_mult_rows(m, x)
```

Arguments

`m`	A matrix
`x`	A vector length nrow(m)

Value

A matrix

`select_replacement_alleles` *Select replacement alleles according to the generalised substance or identity model*

Description

Select replacement alleles according to the generalised substance or identity model

Usage

```
select_replacement_alleles(
  mplat,
  foci_loc_b,
  focus_loc_e,
  pivot_weighting,
  pivot_use_proportion,
  evidence_weighting,
  evidence_use_proportion,
  evidence_balance
)
```

Arguments

`mplat`	A matrix of integers representing the current state of the inflectional system, rotated so that pivots are in columns.
`foci_loc_b`	A vector of integers, column indices for mplat giving the position of the foci.
`focus_loc_e`	An integers, a row index for mplat giving the positions of the foci.
`pivot_weighting`	A string, giving the method for weighting pivots; used either for sampling them, or weighting them when all are taken into account.
`pivot_use_proportion`	A numeric, giving the proportion of the available pivots to use. If set to 0, then just one is used.
`evidence_weighting`	A string, giving the method for weighting evidence morposites; used either for sampling them, or weighting them when all are taken into account.
`evidence_use_proportion`	A numeric, giving the proportion of the available evidence morposites to use. If set to 0, then just one is used.
`evidence_balance`	A numeric, between 0 and 1, giving the balance of negative evidence (max when 0) and positive evidence (max when 1).

Value

A vector of integers, the replacements for foci

`shuffle_within_cols` *Shuffle within columns*

Description

Shuffle within columns

Usage

```
shuffle_within_cols(m, cols = 1:ncol(m))
```

Arguments

`m`	A matrix.
`cols`	A vector of integers, the columns to shuffle within.

Value

A matrix, with each. column’s contents shuffled

`shuffle_within_rows` *Shuffle within rows*

Description

Shuffle within rows

Usage

```
shuffle_within_rows(m, rows = 1:nrow(m))
```

Arguments

`m`	A matrix.
`rows`	A vector of integers, the rows to shuffle within.

Value

A matrix, with each row's contents shuffled.

`sort_mat` *Sort a matrix by all columns, left to right*

Description

Sort a matrix by all columns, left to right

Usage

```
sort_mat(m)
```

Arguments

`m`	A matrix

Value

A matrix, same as m but with rows sorted.

square_mat *Turn a vector into square matrix by repeating it either in every row or every column*

Description

Turn a vector into square matrix by repeating it either in every row or every column

Usage

```
square_mat(rows, cols)
```

Arguments

rows	A vector
cols	A vector

str2ints *Comma delimited string to vector of numerics*

Description

Comma delimited string to vector of numerics

Usage

```
str2ints(s)
```

Arguments

s	A string

Value

A vector of numerics

sum_row_mins *Sum rows' minimums*

Description

Sum rows' minimums

Usage

```
sum_row_mins(m)
```

Arguments

m	A matrix

`sum_row_mins_in_lower_triangle` *Sum rows' minimums in lower triangle*

Description

Sum rows' minimums in lower triangle

Usage

```
sum_row_mins_in_lower_triangle(m)
```

Arguments

`m`	A matrix

`unpack_evolution` *Unpack evolution*

Description

Unpack evolution

Usage

```
unpack_evolution(
  evolution,
  repetition = 1,
  step_method = c("generation", "change"),
  skip_steps = 10
)
```

Arguments

`evolution`	A list, the output from evolve_mplat.
`repetition`	An integer. Which repetition to unpack.
`step_method`	A string, specifying the kind of steps to take through the evolution: by generation or by change.
`skip_steps`	An integer, specifying how many steps to skip ahead each time.

Value

A dataframe of the evolution in long form.

`update_long_mplat` *Update longform mplat by one or more changes*

Description

Update longform mplat by one or more changes

Usage

```
update_long_mplat(mplat, changes)
```

Arguments

`mplat`	A dataframe
`changes`	A dataframe

Value

A dataframe, the mplat updated by the changes

`update_mat` *Replace m1 cells with non-na cells of m2*

Description

Replace m1 cells with non-na cells of m2

Usage

```
update_mat(m1, m2)
```

Arguments

`m1`	A matrix
`m2`	A matrix

Value

A matrix

`update_mplat` *Update mplat by one or more changes*

Description

Update mplat by one or more changes

Usage

```
update_mplat(mplat, changes)
```

Arguments

`mplat`	A matrix
`changes`	A dataframe

Value

A list, the mplat updated by the changes; a vector of changed rows and a vector of changed columns

`update_row_joint_H` *Update pairwise joint entropies of rows*

Description

Update pairwise joint entropies of rows

Usage

```
update_row_joint_H(h_mat, m, rows)
```

Arguments

`h_mat`	A matrix of H
`m`	A matrix, whose contents to get the entropy of.
`rows`	A vector of integers, the rows to update.

Value

A matrix of numerics, the joint entropies.

`which_is_sampled_max` *Which is the maximum value, sampling among equal top*

Description

Which is the maximum value, sampling among equal top

Usage

```
which_is_sampled_max(x)
```

Arguments

`x`	A vector of numerics

Value

A numeric

Index