Package ‘ggmuller’

February 12, 2023

**Title**  Create Muller Plots of Evolutionary Dynamics

**Version**  0.5.6

**Description**  Create plots that combine a phylogeny and frequency dynamics.
Phylogenetic input can be a generic adjacency matrix or a tree of class `phylo`.
Inspired by similar plots in publications of the labs of RE Lenski and JE Barrick. Named for HJ Muller (who popularised such plots) and H Wickham (whose code this package exploits).

**Depends**  R (>= 3.2.0)

**Imports**  dplyr (>= 0.7.0), ggplot2, ape

**Suggests**  RColorBrewer, knitr, rmarkdown

**VignetteBuilder**  knitr

**License**  MIT + file LICENSE

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.1.1

**NeedsCompilation**  no

**Author**  Robert Noble [aut, cre] (<https://orcid.org/0000-0002-8057-4252>)

**Maintainer**  Robert Noble <robnjohnnoble@gmail.com>

**Repository**  CRAN

**Date/Publication**  2023-02-12 11:50:02 UTC

**R topics documented:**

add_empty_pop .................................................. 2
add_start_points ............................................. 3
adj_matrix_to_tree ........................................... 4
branch_singles ................................................ 4
example_df ...................................................... 5
example_edges ................................................ 5
example_pop_df ................................................ 6
find_start_node ............................................... 6
add_empty_pop

Modify a dataframe to enable plotting of populations instead of frequencies

Description

The function adds rows at each time point recording the difference between the total population and its maximum value. Generally there is no need to use this function as Muller_pop_plot calls it automatically.

Usage

add_empty_pop(Muller_df)

Arguments

Muller_df Dataframe created by get_Muller_df

Value

A dataframe that can be used as input in Muller_plot.

Author(s)

Rob Noble, <robjohnnoble@gmail.com>

See Also

get_Muller_df Muller_pop_plot

Examples

Muller_df <- get_Muller_df(example_edges, example_pop_df)
Muller_df2 <- add_empty_pop(Muller_df)
Description

The function 1) identifies when genotypes first have non-zero populations; 2) copies all the rows of data for these time points; 3) modifies the copied rows by decreasing Generation and setting Population of the emerging genotypes to be close to zero; and then 4) adds the modified rows to the dataframe. This ensures that ggplot plots genotypes arising at the correct time points.

Usage

add.start_points(pop_df, start_positions = 0.5)

Arguments

pop_df Dataframe with column names "Identity", "Population", and either "Generation" or "Time"

start_positions Numeric value between 0 and 1 that determines the times at which genotypes are assumed to have arisen (see examples)

Details

By default, the function assumes that each genotype arose half way between the latest time at which its population is zero and the earliest time at which its population is greater than zero. You can override this assumption using the start_positions parameter. If start_positions = 0 (respectively 1) then each genotype is assumed to have arisen at the earliest (respectively latest) time compatible with the data. Intermediate values are also permitted.

Value

The input Dataframe with additional rows.

Author(s)

Rob Noble, <robjohnnoble@gmail.com>

Examples

pop1 <- data.frame(Generation = rep(1:5, each = 4), Identity = rep(1:4, 5), Population = c(1,0,0,0,1,1,0,0,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1))
add.start_points(pop1)

# to see the effect of changing start_positions, compare the Generation columns:
add.start_points(pop1, 0)
add.start_points(pop1, 1)
adj_matrix_to_tree  
Create a tree object of class "phylo" from an adjacency matrix

Description
Create a tree object of class "phylo" from an adjacency matrix

Usage
adj_matrix_to_tree(edges)

Arguments
edges  
Dataframe comprising an adjacency matrix, in which the first column is the parent and the second is the daughter.

Value
A phylo object.

Author(s)
Rob Noble, <robjohnnoble@gmail.com>

Examples
edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6)
tree <- adj_matrix_to_tree(edges1)
class(tree)

branch_singles  
Add branches of length zero to get rid of single nodes in an adjacency matrix

Description
Single nodes are those with exactly one daughter. This function is required by adj_matrix_to_tree, since valid "phylo" objects cannot contain single nodes. If pre-existing branches lack lengths then these are set to 1.

Usage
branch_singles(edges)
example_df

**Arguments**

edges  
Dataframe comprising an adjacency matrix, with column names "Parent" and "Identity"

**Value**

A dataframe comprising the augmented adjacency matrix.

**Author(s)**

Rob Noble, <robjohnnoble@gmail.com>

**Examples**

```r
edges1 <- data.frame(Parent = c(1,1,1,3), Identity = 2:5)
branch_singles(edges1)
```

---

**example_df**  
**Example dataframe**

**Description**

Example dataframe containing both phylogenetic information and population dynamics.

**Usage**

```r
data(example_df)
```

**Format**

A dataframe with column names "Generation", "Identity", "Parent", "Population" and "RelativeFitness"

---

**example_edges**  
**Example adjacency matrix**

**Description**

Example dataframe comprising an adjacency matrix.

**Usage**

```r
data(example_edges)
```

**Format**

A dataframe with column names "Parent" and "Identity"
### example_pop_df

_Example population dataframe_

**Description**

Example dataframe containing population dynamics.

**Usage**

```r
data(example_pop_df)
```

**Format**

A dataframe with column names "Generation", "Identity" and "Population"

### find_start_node

_Move to top of adjacency matrix_

**Description**

Returns the Parent value of the common ancestor.

**Usage**

```r
find_start_node(edges)
```

**Arguments**

- `edges` Dataframe comprising an adjacency matrix, with column names "Parent" and "Identity"

**Value**

The Parent that is the common ancestor.

**Author(s)**

Rob Noble, <robjohnnoble@gmail.com>

**Examples**

```r
edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6)
find_start_node(edges1)
```
get_edges

Extract an adjacency matrix from a larger data frame

Description

Extract an adjacency matrix from a larger data frame

Usage

get_edges(df, generation = NA)

Arguments

df
Dataframe including column names "Identity", "Parent", and either "Generation" or "Time"
generation
Numeric value of Generation (or Time) at which to determine the adjacency matrix (defaults to final time point)

Value

A dataframe comprising the adjacency matrix.

Author(s)

Rob Noble, <robjohnnoble@gmail.com>

See Also

get_population_df

Examples

## Not run:
edges <- get_edges(example_df)

# extract the adjacency matrix from the data frame:
pop_df <- get_population_df(example_df)

# create data frame for plot:
Muller_df <- get_Muller_df(edges, pop_df)

require(RColorBrewer) # for the palette

# draw plot:
um_cols <- length(unique(Muller_df$RelativeFitness)) + 1
Muller_df$RelativeFitness <- as.factor(Muller_df$RelativeFitness)
Muller_plot(Muller_df, colour_by = "RelativeFitness",
            palette = rev(colorRampPalette(brewer.pal(9, "YlOrRd"))(num_cols)),
            add_legend = TRUE)
### get_Muller_df

Create a data frame from which to create a Muller plot

#### Description

Create a data frame from which to create a Muller plot

#### Usage

```r
get_Muller_df(
  edges,
  pop_df,
  cutoff = 0,
  start_positions = 0.5,
  threshold = NA,
  add_zeroes = NA,
  smooth_start_points = NA
)
```

#### Arguments

- `edges`: Dataframe comprising an adjacency matrix, or tree of class "phylo"
- `pop_df`: Dataframe with column names "Identity", "Population", and either "Generation" or "Time"
- `cutoff`: Numeric cutoff; genotypes that never become more abundant than this value are omitted
- `start_positions`: Numeric value between 0 and 1 that determines the times at which genotypes are assumed to have arisen (see examples)
- `threshold`: Deprecated (use cutoff instead, but note that "threshold" omitted genotypes that never become more abundant than *twice* its value)
- `add_zeroes`: Deprecated (now always TRUE)
- `smooth_start_points`: Deprecated (now always TRUE)

#### Value

A dataframe that can be used as input in Muller_plot and Muller_pop_plot.

#### Author(s)

Rob Noble, <robjohnnoble@gmail.com>
get_population_df

Extract population data from a larger data frame

Description

Extract population data from a larger data frame

Usage

get_population_df(df)

See Also

Muller_plot Muller_pop_plot

Examples

# by default, all genotypes are included,
# but one can choose to omit genotypes with max frequency < cutoff:
Muller_df <- get_Muller_df(example_edges, example_pop_df, cutoff = 0.01)

# the genotype names can be arbitrary character strings instead of numbers:
example_edges_char <- example_edges
example_edges_char$Identity <- paste0("foo", example_edges_char$Identity, "bar")
example_edges_char$Parent <- paste0("foo", example_edges_char$Parent, "bar")
example_pop_df_char <- example_pop_df
example_pop_df_char$Identity <- paste0("foo", example_pop_df_char$Identity, "bar")
Muller_df <- get_Muller_df(example_edges_char, example_pop_df_char, cutoff = 0.01)

# the genotype names can also be factors (which is the default for strings in imported data):
example_edges_char$Identity <- as.factor(example_edges_char$Identity)
example_edges_char$Parent <- as.factor(example_edges_char$Parent)
example_pop_df_char$Identity <- as.factor(example_pop_df_char$Identity)
Muller_df <- get_Muller_df(example_edges_char, example_pop_df_char, cutoff = 0.01)

# to see the effect of changing start_positions, compare these two plots:
edges1 <- data.frame(Parent = c(1,2,1), Identity = 2:4)
pop1 <- data.frame(Time = rep(1:4, each = 4),
  Identity = rep(1:4, times = 4),
  Population = c(1, 0, 0, 0,
  2, 2, 0, 0,
  4, 8, 4, 0,
  8, 32, 32, 16))
df0 <- get_Muller_df(edges1, pop1, start_positions = 0)
df1 <- get_Muller_df(edges1, pop1, start_positions = 1)
Muller_plot(df0)
Muller_plot(df1)
Arguments

df         Dataframe including column names "Identity", "Parent", and either "Generation" or "Time"

Value

A dataframe comprising the population dynamics.

Author(s)

Rob Noble, <robjohnnoble@gmail.com>

See Also

get_edges

Examples

## Not run:
# extract the adjacency matrix from the data frame:
edges <- get_edges(example_df)

# extract the populations (and any other attributes) from the data frame:
pop_df <- get_population_df(example_df)

# create data frame for plot:
Muller_df <- get_Muller_df(edges, pop_df)

require(RColorBrewer)  # for the palette

# draw plot:
num_cols <- length(unique(Muller_df$RelativeFitness)) + 1
Muller_df$RelativeFitness <- as.factor(Muller_df$RelativeFitness)
Muller_plot(Muller_df, colour_by = "RelativeFitness",
            palette = rev(colorRampPalette(brewer.pal(9, "YlOrRd"))(num_cols)),
            add_legend = TRUE)

## End(Not run)

---

move_down       Move to daughter in adjacency matrix

Description

Returns the first Identity value in the sorted set of daughters. When parent has no daughters, returns the input Identity.
move_right

Usage

move_down(edges, parent)

Arguments

edges Dataframe comprising an adjacency matrix, with column names "Parent" and "Identity"

parent number or character string specifying whose daughter is to be found

Value

The daughter's Identity.

Author(s)

Rob Noble, <robnasonic@gmail.com>

See Also

move_up move_right

Examples

edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6)
move_down(edges1, 3)

move_right

Move to sibling in adjacency matrix

Description

Returns the next Identity value among the sorted set of siblings. When there is no such sibling, returns the input Identity.

Usage

move_right(edges, identity)

Arguments

edges Dataframe comprising an adjacency matrix, with column names "Parent" and "Identity"

identity number or character string specifying whose sibling is to be found

Value

The sibling’s Identity.
move_up

Author(s)
Rob Noble, <robjohnnoble@gmail.com>

See Also
move_up move_down

Examples
edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6)
move_right(edges1, 3)

move_up

Move to parent in adjacency matrix

Description
Returns the corresponding Parent value. When there is no parent (i.e. at the top of the tree), returns
the input Identity.

Usage
move_up(edges, identity)

Arguments
edges Dataframe comprising an adjacency matrix, with column names "Parent" and
"Identity"
identity number or character string specifying daughter whose parent is to be found

Value
The Parent value.

Author(s)
Rob Noble, <robjohnnoble@gmail.com>

See Also
move_down move_right

Examples
edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6)
move_up(edges1, 3)
Muller_plot

**Description**

Draw a Muller plot of frequencies using ggplot2

**Usage**

```r
Muller_plot(
  Muller_df,
  colour_by = "Identity",
  palette = NA,
  add_legend = FALSE,
  xlab = NA,
  ylab = "Frequency",
  pop_plot = FALSE,
  conceal_edges = FALSE
)
```

**Arguments**

- **Muller_df**: Dataframe created by `get_Muller_df`
- **colour_by**: Character containing name of column by which to colour the plot
- **palette**: Either a brewer palette or a vector of colours (if `colour_by` is categorical)
- **add_legend**: Logical whether to show legend
- **xlab**: Label of x axis
- **ylab**: Label of y axis
- **pop_plot**: Logical for whether this function is being called from `Muller_pop_plot` (otherwise should be FALSE)
- **conceal_edges**: Whether try to conceal the edges between polygons (usually unnecessary or undesirable)

**Value**

None

**Author(s)**

Rob Noble, <robjohnnoble@gmail.com>

**See Also**

`get_Muller_df` `Muller_pop_plot`
Examples

# include all genotypes:
Muller_df1 <- get_Muller_df(example_edges, example_pop_df)
Muller_plot(Muller_df1)

# omit genotypes with max frequency < 0.1:
Muller_df2 <- get_Muller_df(example_edges, example_pop_df, cutoff = 0.2)
Muller_plot(Muller_df2)

# colour by a continuous variable:
Muller_df1 <- get_Muller_df(example_edges, example_pop_df)
Muller_df1$Val <- as.numeric(Muller_df1$Identity)
Muller_plot(Muller_df1, colour_by = "Val", add_legend = TRUE)

Muller_pop_plot

Draw a Muller plot of population sizes using ggplot2

Description

This variation on the Muller plot, which shows variation in population size as well as frequency, is also known as a fish plot.

Usage

Muller_pop_plot(
  Muller_df,
  colour_by = "Identity",
  palette = NA,
  add_legend = FALSE,
  xlab = NA,
  ylab = "Population",
  conceal_edges = FALSE
)

Arguments

Muller_df Dataframe created by get_Muller_df
colour_by Character containing name of column by which to colour the plot
palette Either a brewer palette or a vector of colours (if colour_by is categorical)
add_legend Logical whether to show legend
xlab Label of x axis
ylab Label of y axis
conceal_edges Whether try to conceal the edges between polygons (usually unnecessary or undesirable)

Value

None
**path_vector**

**Author(s)**
Rob Noble, <robjohnnoble@gmail.com>

**See Also**
`get_Muller_df` `Muller_plot`

**Examples**

```r
Muller_df <- get_Muller_df(example_edges, example_pop_df)
Muller_pop_plot(Muller_df)
```

---

**path_vector**

Record a path through all nodes of an adjacency matrix

**Description**
Nodes are traversed in the order that they should be stacked in a Muller plot. Each node appears exactly twice.

**Usage**

```r
path_vector(edges)
```

**Arguments**

- `edges` Dataframe comprising an adjacency matrix, with column names "Parent" and "Identity"

**Value**
A vector specifying the path.

**Author(s)**
Rob Noble, <robjohnnoble@gmail.com>

**Examples**

```r
edges1 <- data.frame(Parent = c(1,1,3,3), Identity = 2:6)
path_vector(edges1)
```
Description

Reorder a Muller plot dataframe by a vector

Usage

reorder_by_vector(df, vector)

Arguments

- df: Dataframe with column names "Identity", "Parent", and either "Generation" or "Time", in which each Identity appears exactly twice
- vector: Vector of Identity values

Value

The reordered dataframe.

Author(s)

Rob Noble, <robjohnnoble@gmail.com>

See Also

path_vector

Examples

df <- data.frame(Generation = c(rep(0, 6), rep(1, 6)),
                 Identity = rep(1:6, 2), Population = c(1, rep(0, 5), 10, rep(1, 5)))
df <- rbind(df, df) # duplicate rows
require(dplyr)
df <- arrange(df, Generation) # put in chronological order
edges1 <- data.frame(Parent = c(1,1,1,3,3), Identity = 2:6) # adjacency matrix
path <- path_vector(edges1) # path through the adjacency matrix
reorder_by_vector(df, path)
Index

* datasets
  example_df, 5
  example_edges, 5
  example_pop_df, 6

add_empty_pop, 2
add_start_points, 3
adj_matrix_to_tree, 4

branch_singles, 4

example_df, 5
example_edges, 5
example_pop_df, 6

find_start_node, 6

get_edges, 7, 10
get_Muller_df, 2, 8, 13, 15
get_population_df, 7, 9

move_down, 10, 12
move_right, 11, 11, 12
move_up, 11, 12, 12
Muller_plot, 9, 13, 15
Muller_pop_plot, 2, 9, 13, 14

path_vector, 15, 16

reorder_by_vector, 16