Package 'bmotif'

October 12, 2022

Title Motif Analyses of Bipartite Networks

Version 2.0.2

Date 2020-09-04

Description Counts occurrences of motifs in bipartite networks, as well as the number of times each node or link appears in each unique position within motifs. Has support for both binary and weighted motifs: can calculate the mean weight of motifs and the standard deviation of their mean weights. Intended for use in ecology, but its methods are general and can be applied to any bipartite network. Full details are given in Simmons et al. (2019) <doi:10.1111/2041-210X.13149>.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports tensor (>= 1.5), Rcpp (>= 0.12.18), reshape2 (>= 1.4.3), gtools (>= 3.8.1), stats (>= 3.5.0)

Suggests testthat, knitr, rmarkdown

LinkingTo Rcpp

VignetteBuilder knitr

Language en-GB

NeedsCompilation yes

Author Benno Simmons [aut, cre], Michelle Sweering [aut], Maybritt Schillinger [aut], Riccardo Di Clemente [aut]

Maintainer Benno Simmons <benno.simmons@gmail.com>

Repository CRAN

Date/Publication 2020-09-11 09:40:06 UTC

R topics documented:

`link_positions` *Calculate link position vectors*

Description

Counts the number of times each link in a network occurs in each unique link position within the motifs

Usage

```
link_positions(M, six_node = FALSE, weights, normalisation = "none")
```

Arguments

`M`	A numeric matrix representing interactions between two groups of nodes. Each row corresponds to a node in one level and each column corresponds to a node in the other level. Elements of M are positive numbers if nodes interact, and 0 otherwise. Formally, M is a biadjacency matrix. When nodes i and j interact, m_ij > 0; if they do not interact, m_ij = 0.
`six_node`	Logical; should positions in six node motifs be counted? Defaults to FALSE.
`weights`	Logical; Should weights of the links be taken into account?
`normalisation`	Which normalisation should be used: 'none','sum', 'position', 'sizeclass', 'sizeclass_plus1', 'sizeclass_NAzero', 'levelsize', 'levelsize_plus1', 'levelsize_NAzero','motif', 'motif_plus1' or 'motif_NAzero'? Defaults to "none". (see details)

Details

Counts the number of times each link in a network occurs in each of the 29 (if `six_node` = FALSE) or 106 (if `six_node` = TRUE) unique link positions within motifs (to quantify a link's structural role). If `six_node` = FALSE, link positions in all motifs containing between 2 and 5 nodes are counted. If `six_node` = TRUE, link positions in all motifs containing between 2 and 6 nodes are counted. Analyses where `six_node` = FALSE are substantially faster than when `six_node` = TRUE, especially for large networks. For large networks, counting six node motifs is also memory intensive. In some cases, R can crash if there is not enough memory.

If interactions are weighted (non-zero matrix elements take values other than 1), these can be incorporated by setting `weights` = `TRUE`. If `weights` = `TRUE`, the function will return the number of times each link occurs in each position, multiplied by the weight of the link, following Mora et al. (2018).

Links between nodes with more interactions will tend to appear in more positions. Normalisation helps control for this effect. bmotif include four types of normalisation:

- **"none"**: performs no normalisation and will return the raw position counts.
- **"sum"**: divides the position measure for each link by the total number of times that link appears in any position (divides each element in a row by the row sum).
- **"position"**: divides the position measure for each link by the total number of times any link occurs in that link position (divides each element in a column by the column sum). This gives a measure of how often a link occurs in a position relative to the other links in the network.
- **Size class normalisation**
  - **"sizeclass"**: divides the position measure for each link by the total number of times that link appears in any position within the same motif size class (the number of nodes a motif contains).
  - **"sizeclass_plus1"**: same as 'sizeclass' but adds one to all position measure values. If a link does not occur in any motifs in a given size class, 'sizeclass' normalisation will return NAs. 'sizeclass_plus1' avoids this by adding one to all counts.
  - **"sizeclass_NAzero"**: same as 'sizeclass' but replaces all NA values with 0. If a link does not occur in any motifs in a given size class, 'sizeclass' normalisation will return NAs. 'sizeclass_NAzero' avoids this by replacing NAs with zero.
- **Levelsize normalisation**
  - **"levelsize"**: divides the position measure for each link by the total number of times that link appears in any position within a motif with a given number of nodes in the top level and the bottom level. For example, the relative frequencies of all position measures in motifs with three nodes in the top level and two nodes in the bottom level will sum to one, as will the relative frequency of all position measures in motifs with 2 nodes in the top level and two nodes in the bottom level, and so on.
  - **"levelsize_plus1"**: same as 'levelsize' but adds one to all position measure values. If a link does not occur in any motifs with a given number of nodes in the top level and the bottom level, 'levelsize' normalisation will return NAs. 'levelsize_plus1' avoids this by adding one to all counts.
  - **"levelsize_NAzero"**: same as 'levelsize' but replaces all NA values with 0. If a link does not occur in any motifs with a given number of nodes in the top level and the bottom level, 'levelsize' normalisation will return NAs. 'levelsize_NAzero' avoids this by replacing NAs with zero.
- **Motif normalisation**
  - **"motif"**: divides the position measure for each link by the total number of times that link appears in any position within the same motif. For example, the relative frequencies of all position measures in motif 5 will sum to one, as will the relative frequency of all position measures in motif 10, and so on.
  - **"motif_plus1"**: same as 'motif' but adds one to all position measure values. If a link does not occur in a particular motif, 'motif' normalisation will return NAs. 'motif_plus1' avoids this by adding one to all counts.
  - **"motif_NAzero"**: same as 'motif' but replaces all NA values with 0. If a link does not occur in a particular motif, 'levelsize' normalisation will return NAs. 'motif_NAzero' avoids this by replacing NAs with zero.

If a matrix is provided without row or column names, default names will be assigned: the first row will be called called 'r1', the second row will be called 'r2' and so on. Similarly, the first column will be called 'c1', the second column will be called 'c2' and so on.

Value

Returns a data frame with one column for each link position: 29 columns if `six_node` is FALSE, and 106 columns if `six_node` is TRUE. Columns names are given as "lpx" where x is the ID of the position as described in the motif dictionary. **To view the 'motif dictionary' showing which link position a given ID corresponds to, enter** `vignette("bmotif-dictionary")`**.**

Each row corresponds to one link in the network. Row names are gives as "x – y", where x is the species in the first level (rows) and y is the species in the second level (columns).

By default, the elements of the data frame will be the raw link position counts. If `weight = TRUE`, link position counts will be multiplied by the link weight. If `normalisation` is set to "sum", "sizeclass" or "position", the elements will be normalised position counts as described above.

References

Mora, B.B., Cirtwill, A.R. and Stouffer, D.B., 2018. pymfinder: a tool for the motif analysis of binary and quantitative complex networks. bioRxiv, 364703.

Simmons, B. I., Sweering, M. J. M., Dicks, L. V., Sutherland, W. J. and Di Clemente, R. bmotif: a package for counting motifs in bipartite networks. bioRxiv. doi: 10.1101/302356

Examples

```
set.seed(123)
row <- 10
col <- 10

# link positions in a binary network
m <- matrix(sample(0:1, row*col, replace=TRUE), row, col)
link_positions(M = m, six_node = TRUE, weights = FALSE, normalisation = "none")

# link positions in a weighted network
m[m>0] <- stats::runif(sum(m), 0, 100)
link_positions(M = m, six_node = TRUE, weights = TRUE, normalisation = "none")
```

`mcount` *Count bipartite motifs*

Description

Counts occurrences of motifs in a bipartite network

Usage

```
mcount(M, six_node = FALSE, normalisation, mean_weight, standard_dev)
```

Arguments

M	A numeric matrix representing interactions between two groups of nodes. Each row corresponds to a node in one level and each column corresponds to a node in the other level. Elements of M are positive numbers if nodes do interact, and 0 otherwise. Formally, M is a biadjacency matrix. When nodes i and j interact, m_ij > 0; if they do not interact, m_ij = 0.
six_node	Logical; should six node motifs be counted? Defaults to FALSE.
normalisation	Logical; should motif frequencies be normalised to control for network size?
mean_weight	Logical; used for weighted networks. Should the mean weight of each motif be computed?
standard_dev	Logical; should the standard deviation of the mean weight for each motif be computed? Warning: can be slow for larger networks.

Details

Counts the number of times each of the 17 motifs up to five nodes (if six_node = FALSE), or 44 motifs up to six nodes (if six_node = TRUE), occurs in a network (note: if the network has weights it will be converted to binary; see below for how to use the weights argument to account for network weights).

Six-node motifs

If six_node = FALSE, all motifs containing between 2 and 5 nodes are counted. If six_node = TRUE, all motifs containing between 2 and 6 nodes are counted. Analyses where six_node = FALSE are substantially faster than when six_node = TRUE, especially for large networks. For large networks, counting six node motifs is also memory intensive. In some cases, R can crash if there is not enough memory.

Normalisation

Larger networks tend to contain more motifs. Controlling for this effect by normalising motif counts is important if different sized networks are being compared. If normalisation = TRUE, motif frequencies are normalised in four ways:

- **"normalise_sum"**: converts each frequency to a relative frequency by expressing counts as a proportion of the total number of motifs in the network
- **"normalise_sizeclass"**: expresses counts as a proportion of the total number of motifs within each motif size class (the number of nodes a motif contains). For example, the relative frequency of all two-node motifs will sum to one, as will the relative frequency of all three-, four-, five- and six-node motifs.
- **"normalise_levelsize"**: expresses counts as a proportion of the total number of motifs with a given number of nodes in the top level and the bottom level. For example, the relative frequencies of all motifs with three nodes in the top level and two nodes in the bottom level will sum to one, as will the relative frequency of all motifs with 2 nodes in the top level and two nodes in the bottom level, and so on. This normalisation is helpful because each set of species with a given number of nodes in the top and bottom level is assigned to one motif that describes the interactions among those species (Cirtwill and Eklöf, 2018). For example, all sets of interacting species with two species in the top level and two in the bottom level will be assigned to either motif 5 or motif 6. 'normalise_levelsize' allows you to see the relative

proportion of species which were assigned to each of these motifs. Note that some motifs will always return a value of 1 as they are the only motif with that particular combination of nodes in the top and bottom level. For example, motif 2 will always sum to 1 because it is the only motif with one node in the top level and two nodes in the bottom level.

- **"normalise_nodesets"**: expresses frequencies as the number of node sets that are involved in a motif as a proportion of the number of node sets that could be involved in that motif (Poisot and Stouffer, 2017). For example, in a motif with three nodes in one level (A) and two nodes in the other level (P), the maximum number of node sets which could be involved in the motif is given by the product of binomial coefficients, choosing three nodes from A and two from P.

Weighted networks

`mcount` also supports weighted networks. We let the weight of a given subgraph be the arithmetic mean of the weights of its links (note: we only consider links which are actually present), following Mora et al. (2018).

For each motif we do the following:
We calculate the weights of all subgraphs of the same type as (formally: isomorphic to) the motif.
If `mean_weight = TRUE`, we compute the arithmetic mean of the subgraph weights.
If `standard_dev = TRUE`, we compute the standard deviation of the subgraph weights.

For example, let there be two subgraphs, A and B, which are isomorphic to motif 5. Subgraph A has three links with weights 1, 2 and 3; subgraph B has three links with weights 4, 5 and 6. The weight of subgraph A is the mean of 1, 2 and 3, which is 2. The weight of subgraph B is the mean of 4, 5 and 6 which is 5. The mean weight of motif 5 which would be returned by `mcount` is therefore the mean of 2 and 5 which is 3.5.

Value

Returns a data frame with one row for each motif: either 17 rows (if `six_node = FALSE`) or 44 rows (if `six_node = TRUE`). The data frame has three columns. The first column ("motif") indicates the motif ID as described in Simmons et al. (2017). **To view the 'motif dictionary' showing which motif a given ID corresponds to, enter** `vignette("bmotif-dictionary")`**.** The second column ("nodes") indicates how many nodes the motif contains. The third column ("frequency") is the number of times each motif appears in the network.

If `normalisation = TRUE`, three additional columns are added to the output data frame, each corresponding to a different method of normalising motif frequencies as described above. If `mean_weight = TRUE`, an additional column with the mean weight values is added. If `standard_dev = TRUE`, an additional column with the standard deviation values is added.

References

Baker, N., Kaartinen, R., Roslin, T., and Stouffer, D. B. (2015). Species' roles in food webs show fidelity across a highly variable oak forest. Ecography, 38(2):130–139.

Cirtwill, A. R. and Eklöf, A (2018), Feeding environment and other traits shape species' roles in marine food webs. Ecol Lett, 21: 875-884. doi:10.1111/ele.12955

Mora, B.B., Cirtwill, A.R. and Stouffer, D.B., 2018. pymfinder: a tool for the motif analysis of binary and quantitative complex networks. bioRxiv, 364703.

Poisot, T. & Stouffer, D. (2016). How ecological networks evolve. bioRxiv.

Simmons, B. I., Sweering, M. J. M., Dicks, L. V., Sutherland, W. J. and Di Clemente, R. bmotif: a package for counting motifs in bipartite networks. bioRxiv. doi: 10.1101/302356

Examples

```
set.seed(123)
row <- 10
col <- 10

# motif counts for a binary network
m <- matrix(sample(0:1, row*col, replace=TRUE), row, col)
mcount(M = m, six_node = TRUE, normalisation = TRUE, mean_weight = FALSE, standard_dev = FALSE)

# motif counts in a weighted network
m[m>0] <- stats::runif(sum(m), 0, 100)
mcount(M = m, six_node = TRUE, normalisation = TRUE, mean_weight = TRUE, standard_dev = TRUE)
```

---

node_positions *Calculate node position vectors*

Description

For binary networks, counts the number of times each node appears in each unique node position within motifs; for weighted networks calculates a range of weighted node position measures.

Usage

```
node_positions(
  M,
  six_node = FALSE,
  level = "all",
  weights_method,
  weights_combine = "none",
  normalisation = "none"
)
```

Arguments

M	A numeric matrix representing interactions between two groups of nodes. Each row corresponds to a node in one level and each column corresponds to a node in the other level. Elements of M are positive numbers if nodes do interact, and 0 otherwise. Formally, M is a biadjacency matrix. When nodes i and j interact, m_ij > 0; if they do not interact, m_ij = 0. If interactions are weighted (non-zero matrix elements take values other than 1), the function will automatically convert the matrix to a binary matrix.
six_node	Logical; should six node motifs be counted? Defaults to FALSE.
level	Which node level should positions be calculated for: "rows", "columns" or "all"? Defaults to "all".

`weights_method` The method for calculating weighted positions; must be one of 'none', 'mean_motifweights', 'total_motifweights', 'mean_nodeweights', 'total_nodeweights', 'contribution', 'mora' or 'all' (see details).

`weights_combine` Method for combining weighted position measures; must be one of 'none', 'mean' or 'sum' (see details). Defaults to 'none'.

`normalisation` Which normalisation should be used: "none","sum","sizeclass", "sizeclass_plus1", "sizeclass_NAzero", "position","levelsize","levelsize_plus1","levelsize_NAzero","motif","motif_plus1" or "motif_NAzero" (see details)? Defaults to "none".

Details

For binary networks, counts the number of times each node occurs in each unique position within motifs (to quantify a node's structural role). If networks are weighted, `node_positions` can also calculate various weighted node position measures.

If a matrix is provided without row or column names, default names will be assigned: the first row will be called called 'r1', the second row will be called 'r2' and so on. Similarly, the first column will be called 'c1', the second column will be called 'c2' and so on.

Six node

If `six_node` = FALSE, node positions in all motifs containing between 2 and 5 nodes are counted. If `six_node` = TRUE, node positions in all motifs containing between 2 and 6 nodes are counted. Analyses where `six_node` = FALSE are substantially faster than when `six_node` = TRUE, especially for large networks. For large networks, counting six node motifs is also memory intensive. In some cases, R can crash if there is not enough memory.

Level

The `level` argument controls which level of nodes positions are calculated for: "rows" returns position counts for all nodes in rows, "columns" returns position counts for all nodes in columns, and "all" return counts for all nodes in the network.

Weighted networks

`node_positions` also supports weighted networks for motifs up to five nodes. Weighted analyses are controlled using two arguments: `weights_method` and `weights_combine`. These are described in detail below:

- **'weights_method'**: determines how the weighted position of a node is calculated in each motif occurrence.
  - **'none'**: weights are ignored and `node_positions` returns the frequency with which each node occurs in each unique position within motifs. `'weights_combine'` must also be 'none'.
  - **'mean_motifweights'**: for a given node in a given position in a motif occurrence (formally a subgraph isomorphic to a particular motif), returns the mean weight of that motif occurrence i.e. the mean of all link strengths in that motif occurrence.
  - **'total_motifweights'**: for a given node in a given position in a motif occurrence (formally a subgraph isomorphic to a particular motif), returns the total weight of that motif occurrence i.e. the sum of all link strengths in that motif occurrence.

  - **'mean_nodeweights'**: for a given node in a given position in a motif occurrence (formally a subgraph isomorphic to a particular motif), returns the mean weight of that focal node's links.
  - **'total_nodeweights'**: for a given node in a given position in a motif occurrence (formally a subgraph isomorphic to a particular motif), returns the total weight of that focal node's links.
  - **'contribution'**: for a given node in a given position in a motif occurrence (formally a subgraph isomorphic to a particular motif), returns the total weight of that focal node's links as a proportion of the total weight of that motif occurrence. i.e. the sum of the focal node's links divided by the sum of all link strengths in that motif occurrence.
  - **'mora'**: calculates a contribution measure following Mora et al. (2018).
  - **'all'**: calculates all the above measures (except 'none') and returns them as a list of length five.

- **'weights_combine'**: determines how weighted position measures are combined across motif occurrences to give an overall measure for a each node in a each position.
  - **'none'**: weights are ignored and `node_positions` returns the frequency with which each node occurs in each unique positions within motifs. `'weights_method'` must also be 'none'.
  - **'sum'**: weighted measures are summed across occurrences.
  - **'mean'**: the mean of the weighted measure across occurrences is calculated.

Normalisation

Nodes with more interactions will tend to appear in more positions. Normalisation helps control for this effect. bmotif include six main types of normalisation:

- **"none"**: performs no normalisation and will return the raw position measure
- **"sum"**: divides the position measure for each node by the total number of times that node appears in any position (divides each element in a row by the row sum).
- **"position"**: divides the position measure for each node by the total number of times any node occurs in that node position (divides each element in a column by the column sum). This gives a measure of how often a node occurs in a position relative to the other nodes in the network.
- **Size class normalisation**
  - **"sizeclass"**: divides the position measure for each node by the total number of times that node appears in any position within the same motif size class (the number of nodes a motif contains).
  - **"sizeclass_plus1"**: same as 'sizeclass' but adds one to all position measures If a species does not occur in any motifs in a given size class, 'sizeclass' normalisation will return NAs. 'sizeclass_plus1' avoids this by adding one to all counts.
  - **"sizeclass_NAzero"**: same as 'sizeclass' but replaces all NA values with 0. If a species does not occur in any motifs in a given size class, 'sizeclass' normalisation will return NAs. 'sizeclass_NAzero' avoids this by replacing NAs with zero.
- **Levelsize normalisation**
  - **"levelsize"**: divides the position measure for each node by the total number of times that node appears in any position within a motif with a given number of nodes in the top level and the bottom level. For example, the relative frequencies of all position measures in

motifs with three nodes in the top level and two nodes in the bottom level will sum to one, as will the relative frequency of all position counts in motifs with 2 nodes in the top level and two nodes in the bottom level, and so on.

  - **"levelsize_plus1"**: same as 'levelsize' but adds one to all position measures If a species does not occur in any motifs with a given number of nodes in the top level and the bottom level, 'levelsize' normalisation will return NAs. 'levelsize_plus1' avoids this by adding one to all counts.
  - **"levelsize_NAzero"**: same as 'levelsize' but replaces all NA values with 0. If a species does not occur in any motifs with a given number of nodes in the top level and the bottom level, 'levelsize' normalisation will return NAs. 'levelsize_NAzero' avoids this by replacing NAs with zero.

- **Motif normalisation**
  - **"motif"**: divides the position measure for each node by the total number of times that node appears in any position within the same motif. For example, the relative frequencies of all position measures in motif 5 will sum to one, as will the relative frequency of all position counts in motif 10, and so on.
  - **"motif_plus1"**: same as 'motif' but adds one to all position measures. If a species does not occur in a particular motif, 'motif' normalisation will return NAs. 'motif_plus1' avoids this by adding one to all counts.
  - **"motif_NAzero"**: same as 'motif' but replaces all NA values with 0. If a species does not occur in a particular motif, 'levelsize' normalisation will return NAs. 'motif_NAzero' avoids this by replacing NAs with zero.

Value

Returns a data frame with one column for each node position: 46 columns if `six_node` is FALSE, and 148 columns if `six_node` is TRUE. Columns names are given as "npx" where x is the ID of the position as described in Simmons et al. (2017) (and originally in Appendix 1 of Baker et al. (2015)). **To view the 'motif dictionary' showing which node position a given ID corresponds to, enter** `vignette("bmotif-dictionary")`**.**

For a network with A rows and P columns, by default (where `level` = "all") the data frame has A + P rows, one for each node. If `level` = "rows", the data frame will have A rows, one for each row node; if `level` = "columns", it will have P rows, one for each column node.

By default, the elements of this data frame will be the raw binary or weighted position measures (depending on which was requested). If `normalisation` is set to something other than "none", the elements will be normalised position counts as described above.

If `weights_method` is set to 'all', `node_positions` instead returns a list of length five, each containing a data.frame corresponding to one of the five weighting methods described above.

References

Baker, N., Kaartinen, R., Roslin, T., and Stouffer, D. B. (2015). Species' roles in food webs show fidelity across a highly variable oak forest. Ecography, 38(2):130–139.

Mora, B.B., Cirtwill, A.R. and Stouffer, D.B., 2018. pymfinder: a tool for the motif analysis of binary and quantitative complex networks. bioRxiv, 364703.

Simmons, B. I., Sweering, M. J. M., Dicks, L. V., Sutherland, W. J. and Di Clemente, R. bmotif: a package for counting motifs in bipartite networks. bioRxiv. doi: 10.1101/302356

Examples

```
set.seed(123)
row <- 10
col <- 10

# node positions in a binary network
m <- matrix(sample(0:1, row*col, replace=TRUE), row, col)
node_positions(M = m, six_node = TRUE, weights_method = "none", weights_combine = "none")

# node positions in a weighted network
m[m>0] <- stats::runif(sum(m), 0, 100)
node_positions(M = m, six_node = FALSE, weights_method = "all", weights_combine = "sum")
```

Index