Package 'cassandRa'

May 8, 2026

Type Package

Title Finds Missing Links and Metric Confidence Intervals in Ecological Bipartite Networks

Version 0.2.0

Description Provides methods to deal with under sampling in ecological bipartite networks from Terry and Lewis (2020) Ecology <doi:10.1002/ecy.3047> Includes tools to fit a variety of statistical network models and sample coverage estimators to highlight most likely missing links. Also includes simple functions to resample from observed networks to generate confidence intervals for common ecological network metrics.

License GPL-3

Encoding UTF-8

Imports bipartite (>= 2.11), reshape2 (>= 1.4.3), magrittr (>= 1.5), vegan (>= 2.5-3), purrr (>= 0.2.5), dplyr, tidyr (>= 0.8), ggplot2 (>= 3.1.0), boot, stats, rlang

Suggests knitr (>= 1.20), rmarkdown (>= 1.10), pROC (>= 1.13.0), lattice

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation no

Author Chris Terry [aut, cre, cph]

Maintainer Chris Terry <christerry3@btinternet.com>

Repository CRAN

Date/Publication 2024-06-10 17:10:09 UTC

Contents

`CalcHostLevelCoverage` *Estimated probabilities of missing links based on the host Level Coverage Deficit*

Description

Calls CoverageEstimator() to calculate host-level coverage deficit, then divides this by the number of unobserved interactions of that host.

Usage

```
CalcHostLevelCoverage(list)
```

Arguments

`list`	Network List

Value

A network list, with 'C_defmatrix', a matrix of probabilities based on coverage deficit, and 'OverallChaoEst' an estimate of the overall coverage deficit of the network.

`ComputeCI` *Compute Basic Confidence Intervals*

Description

Compute Basic Confidence Intervals

Usage

```
ComputeCI(df)
```

Arguments

`df`	A data frame produced by `RarefyNetwork()`

Value

a dataframe detailing confidence intervals at each tested sample size

Examples

```
data(Safariland, package = 'bipartite')
X<-RarefyNetwork(Safariland, n_per_level = 100)
PlotRarefaction(X)
```

`CoverageEstimator` *Coverage Estimator, using Chao1 Index, Turing-Good or Binomial depending on what is possible*

Description

An estimate of the sample coverage, which tries to use the most appropriate method

Usage

```
CoverageEstimator(x, cutoff = 5, BayesPrior = "Flat")
```

Arguments

`x`	A vector of integers, the observed sample counts
`cutoff`	When to switch from binomial model to Chao1 estimator
`BayesPrior`	Prior to use. Either 'Flat' or 'Jeffereys'.

Details

Sample coverage is defined as the probability that the next interaction drawn is of a type not yet seen

If the sample size is at or below a cutoff (5) or if all the samples are singletons, this is calculated as the posterior mean of a binomial model using a flat prior (this can be changed to a Jeffereys).

If there are singletons but no doubletons, the Turing-Good estimate is used: c_hat = 1 - (f1/n)

If there are both singletons and doubletons, the Chao1 index is used:

c_hat = 1 -((f1/n) * ((f1*(n-1))/((n-1)*(f1+(2*f2)))))

Value

c_hat, the estimated coverage. (i.e. 1- C_def)

`CreateListObject` *Generates a network list from a food web*

Description

Gets a network in the base bipartite package format into a list format. N.B. Throughout this package uses hosts to refer to the focal layer, and 'wasps' the response layer, although this could equally be 'plants' and 'pollinators'.

Usage

```
CreateListObject(web)
```

Arguments

`web`	in format specified by the bipartite package. Rows = focal layer, columns = response layer

Value

A network list for use with other functions in EcoLinkPredict package

Examples

```
data(Safariland, package = 'bipartite')
demolist<-CreateListObject(Safariland)
str(demolist)
```

FitAllModels *Fit all the models*

Description

Internal function called by `PredictLinks()` Fits the coverage deficit, Trait, Centrality, Matching-Centrality and SBM models by sequentially calling the individual functions.

Usage

```
FitAllModels(list, RepeatModels = 10)
```

Arguments

`list`	A network list
`RepeatModels`	How many times to fit each model from different starting points. Uses best half (rounding up)

Value

A network list including the model fit

FitBothMandC *Fit Matching-Centrality Model*

Description

Fit a model that contains both a trait-matching and a centrality term based on Rohr et al. (2016)

Usage

```
FitBothMandC(
  list,
  N_runs = 10,
  maxit = 10000,
  method = "Nelder-Mead",
  ExtraSettings = NULL
)
```

Arguments

`list`	Network List
`N_runs`	Number of different start points for k2 and lambda to try. The best (maximum likelihood) half will be used to construct the probability matrix
`maxit`	Default = 10'000
`method`	Passed to optim, default = 'Nelder-Mead'
`ExtraSettings`	Other control settings to pass to optim()

Value

Network list with added 'B_par',the best fitting parameters, 'M_ProbsMatrix', the probability matrix

References

Rohr, R.P., Naisbit, R.E., Mazza, C. & Bersier, L.-F. (2016). Matching-centrality decomposition and the forecasting of new links in networks. Proc. R. Soc. B Biol. Sci., 283, 20152702

`FitCentrality` *Fit Centrality Model*

Description

Repeatedly fits a centrality model to a binary interaction network to return a probability matrix

Usage

```
FitCentrality(
  list,
  N_runs = 10,
  maxit = 10000,
  method = "Nelder-Mead",
  ExtraSettings = NULL
)
```

Arguments

`list`	Network List
`N_runs`	Number of start points to try. The best (maximum likelihood) half will be used to construct the probability matrix
`maxit`	Default = 10'000
`method`	Passed to optim, default = 'Nelder-Mead'
`ExtraSettings`	Other control settings to pass to optim()

Value

Network list with added 'C_par', best fitting parameters, C_ProbsMatrix, the probability matrix

`FitMatching` *Fit Latent Trait (Matching Model)*

Description

Repeatedly fits a latent trait model to a binary interaction network to return a probability matrix

Usage

```
FitMatching(
  list,
  N_runs = 10,
  maxit = 10000,
  method = "Nelder-Mead",
  ExtraSettings = NULL
)
```

Arguments

`list`	Network List
`N_runs`	Number of start points for k2 and lambda to try. The best (maximum likelihood) half will be used to construct the probability matrix
`maxit`	Default = 10'000
`method`	Passed to optim, default = 'Nelder-Mead'
`ExtraSettings`	Other control settings to pass to optim()

Details

The optimiser is started at values derived from the row-sums and column-sums of a CCA analysis, which correspond closely to latent traits by matching closely related species together.

The k2 and lambda parameters are started from points drawn from a uniform distribution 0:1.

Value

Network list with added 'M_par',the best fitting parameters, 'M_ProbsMatrix', the probability matrix

FitSBM *Fit SBM Model*

Description

Fit SBM Model

Usage

```
FitSBM(list, n_SBM = 10, G = NULL)
```

Arguments

list	Network List
n_SBM	Number of SBM models to fit. Default is 10. The top half (rounding up) are retained and averaged to produce a probability matrix.
G	The number of groups to divide the top layer and the focal layer into.

Value

Network list with 'SBM_ProbsMat', a matrix of probabilities assigned to each possible interaction, 'SBM1', the best model fit derived from Optimise_SBM(), and 'SBM_G', the number of fitted groups.

make_true_and_sample_web

Make an artificial bipartite networks with some properties of ecological networks, then sample from it

Description

Core model adapted from: "Sampling bias is a challenge [...]: lessons from a quantitative nichemodel" by Jochen Frund, Kevin S. McCann and Neal M. Williams

Usage

```
make_true_and_sample_web(
  seed = NULL,
  specpar = 1,
  n_hosts = 10,
  n_wasps = 10,
  TargetTrueConn = 0.5,
  SampleObs = 1000,
  abun_mean = 5,
  abun_sdlog = 1,
```

```
  traitvsnested = 0.5,
  hosttrait_n = "two"
)
```

Arguments

seed	Random number generator seed, if specified.
specpar	Specialisation parameter, equal to 1/sd of the normal curve that defines the consumption range
n_hosts	Number of focal level species (e.g. hosts, flowers)
n_wasps	Number of non-focal level species (e.g. parasitic wasps, pollinators)
TargetTrueConn	Proportion of possible interactions to keep
SampleObs	Number of samples to draw
abun_mean	Mean abundance level (log scale).
abun_sdlog	Distributon of abundance level (SD log scale).
traitvsnested	The relative balance between the nestedness generator and the trait-based generator
hosttrait_n	Number of trait dimensions. Default 'two', uses two traits, with one dominant. 'single' and 'multi' retained from Frund et al.

Details

Abundances are assigned by generating abundances that match a log-normal distribution (but without introducing noise)

Value

A network list containing 'obs' a matrix of observations, 'TrueWeb' a matrix of the 'true'] drawn web, and number of other properties of these networks.

Examples

```
make_true_and_sample_web()
```

Optimiser *Optimiser wrapper for network models*

Description

Optimiser wrapper for network models

Usage

```
Optimiser(
  i = NULL,
  maxit = 10000,
  method = "Nelder-Mead",
  A,
  N_p,
  fixedSt_P = c(),
  N_unif_P = 0,
  func,
  ExtraSettings = NULL
)
```

Arguments

i	RNG Seed to set
maxit	Maximum number of iterations to be passed to optim (default is 10000)
method	Optimiser method to pass to optim. Default is
A	Interaction Presence-Absence matrix
N_p	Number of parameters to draw from a normal distribution
fixedSt_P	Vector of fixed parameters to pass
N_unif_P	Number of parameters to take from a uniform distribution
func	Function to optimiser
ExtraSettings	Additional setting to pass to control

Value

A 'fit' object form optim, with a few of the input parameters attached.

Optimise_SBM *Custom optimiser function for SBM models*

Description

Designed to be called by FitSBM()

Usage

```
Optimise_SBM(i = NULL, A, G, N_Rounds_max = 500, plot = FALSE)
```

Arguments

i	Seed
A	Binary Interaction Matrix
G	Number of Groups
N_Rounds_max	Maximum number round to keep drawing
plot	If set to TRUE, plots the progress of likelihood improvement, used to check if convergence is good.

Details

Based on optimising algorithm described in Larremore, D.B., Clauset, A. & Jacobs, A.Z. (2014). Efficiently inferring community structure in bipartite networks. Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., 90, 1-12

Initially all species are randomly assigned to groups. Then, one at a time, each species is swapped into a different group and the likelihood of the model assessed (with SBMLik()).

The best model of all these swaps is then selected (even if it is worse) and used in the next round of swapping.

This fits the 'degree-corrected' biSBM mdoel of Larremore et al., which is generally better when there are broad degree distributions

This is repeated until either n_rounds_max is reached, or the (most commonly), if the best model in the last 20 is within 0.1 log-likelihood of the best overall (implying it has stopped improving).

Value

A list containing 'LogLik' (the maximum likelihood found) 'SB_H', the group assignments of the host, 'SB_W', the group assignments of the other level, and 'Omega_rs', the interaction probabilities between groups.

PlotFit *Plot the fitted network models*

Description

Takes the output from other functions (including `PredictLinks()`) to visualise the fit to the data and predictions of missing links.

Usage

```
PlotFit(
  list,
  Matrix_to_plot,
  OrderBy = "Default",
  addDots = TRUE,
  title = NULL,
```

```
  Combine = "+",
  RemoveTP = FALSE,
  GuidesOff = TRUE
)
```

Arguments

list	A list-format network (output from xxx)
Matrix_to_plot	Which matrix / matrices to plot. One or more of 'C_def','C', 'M', 'B', 'SBM'
OrderBy	How to order the plot. One of 'Default','Degree','Manual', 'LatentTrait','SBM', 'AsPerMatrix'
addDots	Should dots be added to show observations. TRUE, FALSE or 'Size', to plot by interaction strength
title	A title. By default it will use the value of Matrix_to_plot
Combine	How should multiple matrices be combined. Either '+' which averages them (default), or '*' which multiples
RemoveTP	Should true positives be set to NA in order to highlight differences in predictions. Default is FALSE
GuidesOff	Should the legends be switched off. Defaults to TRUE

Details

See the vignette for a more through description and examples.

Value

A ggplot object, which by default will print to the device, but can be added to make further tweaks

Examples

```
## Not run:
data(Safariland, package = 'bipartite')
Predictions<- PredictLinks(Safariland)
PlotFit(Predictions, Matrix_to_plot = 'SBM')

## End(Not run)
```

PlotRarefaction *Plot Metric Response To Network Rarefaction*

Description

Used to plot the output from `RarefyNetwork()`. See vignette!

Usage

```
PlotRarefaction(df)
```

Arguments

`df`	A data frame produced by RarefyNetwork

Value

A ggplot

Examples

```
data(Safariland, package = 'bipartite')
X<-RarefyNetwork(Safariland, n_per_level = 100)
ComputeCI(X)
```

PredictLinks *Generates a network list from a food web and fits all network models*

Description

First calls `CreateListObject` to convert a matrix suitable for the bipartite package into a list structure.

Usage

```
PredictLinks(web, RepeatModels = 10)
```

Arguments

`web`	in format specified by the bipartite package. Rows = focal layer, columns = response layer
`RepeatModels`	How many times to fit each model from different starting points. Uses best half (rounding up)

Details

Then it calls `FitAllModels` to fit each of the missing link models in turn.

Value

A network list including a large number of outputs.

Examples

```
## Not run:
data(Safariland, package = 'bipartite')
PredictLinks(Safariland)

## End(Not run)
```

`RarefyNetwork` *Recalculate Network Metrics With Rarefied Webs*

Description

Resamples empirical network observations at a range of sampling levels and calls networklevel() function from bipartite package to calculate network metrics.

Usage

```
RarefyNetwork(
  web,
  n_per_level = 1000,
  frac_sample_levels = seq(0.2, 1, l = 5),
  abs_sample_levels = NULL,
  metrics = "info",
  PARALLEL = FALSE,
  cores = 2,
  output = "df",
  ...
)
```

Arguments

`web`	A matrix format web, as for `bipartite`
`n_per_level`	How many samples to take per sample level. Default is 1000.

`frac_sample_levels`
: Sequence of fractions of original sample size to resample at.

`abs_sample_levels`
: If supplied, vector of absolute sample sizes to use to override `frac_sample_levels`. Default = NULL

`metrics`
: vector of metrics to calculate. Will be passed to `index` of `networklevel()`. Default = 'info'

`PARALLEL`
: Logical. If TRUE, will use parallel package to speed up metric calculation. Default = FALSE

`cores`
: If using parallel, how man cores to use. Default = 2

`output`
: String specifying output. If 'plot' will return a ggplot facetted by metric using `PlotRarefaction()`. If 'CI' will return a data frame (using `ComputeCI()` containing 5 columns: Metric, LowerCI, UpperCI, Mean, SampleSize. Otherwise will return a data frame of the raw recalculated metrics, with a separate column for each metric, and the last column specifying the resample size.

`...`
: Additional arguments to pass to `networklevel`. e.g. empty.web=FALSE

Details

Can return either a data frame of raw metrics, a ggplot or a data frame of 'confidence intervals'.

These CI are calculated from the set of resamples by ordering the network values and taking the value of the metric ranked at the 5th and 95th percentile. (this method is very similar to that employed by Casas *et al.* 2018 *Assessing sampling sufficiency of network metrics using bootstrap* Ecological Complexity 36:268-275.)

Note that confidence intervals for many metrics, particularly qualitative ones, will be biased by the issue of false-negatives. Resampling of observations will not introduce missing links.

By default the size of resamples are taken to be proportional to the original sample size. Original sample size is defined as the sum of the supplied web. If a specific set of sample sizes is wanted, use `abs_sample_levels`

It is possible to extrapolate how increases sample size may lead to increased confidence in a metric too. Set the sequence to `frac_sample_levels` to go beyond 1.

Value

Either a dataframe or a ggplot object. See details.

See Also

`networklevel`

Examples

```
data(Safariland, package = 'bipartite')
RarefyNetwork(Safariland, n_per_level = 100)
```

SortResponseCategory *Adds a dataframe that defines each interaction as true positive, false negative or true negative*

Description

Adds a dataframe that defines each interaction as true positive, false negative or true negative

Usage

```
SortResponseCategory(list)
```

Arguments

list	Network list

Value

A Network list object with ObsSuccess, a dataframe detailing all the interactions and whether they are True Positives, False Negative or True Negatives

TestAllModels *Test the models by AUC*

Description

The function assumes FitAllModels() has already been run. It is a wrapper for 'SortResponseCategory()' and 'TestAUC()'

Usage

```
TestAllModels(list)
```

Arguments

list	A network list

Value

the network list with added AUC data. Key values are 'AUC', a dataframe with the AUC of each model and many combinations.

`TestAUC` *Test via AUC the predictive capacity of each model or combination of models*

Description

Test via AUC the predictive capacity of each model or combination of models

Usage

```
TestAUC(list)
```

Arguments

`list`	Network List

Value

a list with 'DataforAUC', a data frame with each interaction as a row and the predictions of each model, and 'AUC', a data frame with the predictive capacity of all the models and many combinations

Index