Package ‘CooccurrenceAffinity’

May 7, 2026

Type Package

Title Affinity in Co-Occurrence Data

Version 2.0.0

Date 2026-01-27

Description Computes a novel metric of affinity between two entities based on their co-occurrence (using binary presence/absence data). The metric and its maximum likelihood estimator (alpha hat) were advanced in Mainali, Slud, et al, 2021 <doi:10.1126/sciadv.abj9204>. Four types of confidence intervals and median interval were developed in Mainali and Slud, 2022 <doi:10.1101/2022.11.01.514801>. The \`finches\` dataset is bundled with the package.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1), BiasedUrn (>= 2.0.9)

Imports cowplot, ggplot2, plyr, reshape

URL `https://github.com/kpmainali/CooccurrenceAffinity`

BugReports `https://github.com/kpmainali/CooccurrenceAffinity/issues`

RoxygenNote 7.3.1

NeedsCompilation no

Author Kumar Mainali [aut, cre],
Eric Slud [aut]

Maintainer Kumar Mainali <`kpmainali@gmail.com`>

Repository CRAN

Date/Publication 2026-02-12 08:30:02 UTC

Contents

`AcceptAffCI` *Acceptability Interval*

Description

This function calculates the "Acceptability Interval" of Blaker for the log-odds parameter alpha in the Extended Hypergeometric distribution.

Usage

```
AcceptAffCI(x, marg, lev, CPint)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)
`CPint`	the exact conservative ("Clopper-Pearson-type") interval CI.CP calculated in the function AlphInts()

Details

This function calculates the "Acceptability Interval" based on "Acceptability Function" computed by AcceptAffin(). This interval, developed by Blaker (2000), was proved in that paper's Theorem 1 in a more general class of estimation problems to have three essential properties: it falls within the CI.CP confidence interval; it maintains the property of being conservative, i.e., of having coverage probability under the Extended Hypergeometric (mA,mB,N, alpha) distribution at least as large as the nominal level; and it is larger when the confidence level is larger.

Value

This function returns the "Acceptability Interval" of Blaker (2000). The code is adapted from Blaker's Splus code for the case of an unknown binomial proportion.

Author(s)

Eric Slud

References

Blaker, H. (2000), “Confidence curves and improved exact confidence intervals for discrete distributions", Canadian Journal of Statistics 28, 783-798.

Examples

```
auxCP = AlphInts(30,c(50,80,120), lev=0.9)$CI.CP
AcceptAffCI(30,c(50,80,120), 0.9, auxCP)

AlphInts(30,c(50,80,120), lev=0.9)$CI.Blaker
```

`AcceptAffin` *Calculates the "Acceptability Function" used in defining Blaker's (2000) Acceptability Interval and computing the latter in the function AcceptAffCI().*

Description

This function calculates the "Acceptability Function" of Blaker (2000, Thm.1, p.785) for the log-odds parameter alpha in the Extended Hypergeometric distribution.

Usage

```
AcceptAffin(x, marg, alph)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`alph`	a vector of (one or more) real-valued "alpha" values, where alpha is the log-odds parameter in the Extended Hypergeometric distribution

Details

This function calculates the "Acceptability Function" of Blaker (2000, Thm.1, p.785) for the log-odds parameter alpha in the Extended Hypergeometric distribution, a function from which the "Acceptability Interval" is calculated by another CooccurrenceAffinity package function AcceptAffCI().

Value

This function returns the "Acceptability Function" that is later used by another function AcceptAffCI() to compute "Acceptability Interval".

Author(s)

Eric Slud

References

Blaker, H. (2000), “Confidence curves and improved exact confidence intervals for discrete distributions", Canadian Journal of Statistics 28, 783-798.

`affinity` *Computes alpha, probability, expected co-occurence, median interval, various confidence intervals, other indices of affinity, etc.*

Description

This is the principal function of "CooccurrenceAffinity" package that analyzes occurrence or abundance data (e.g., species by site) using other functions of this package and returns several quantities in one main dataframe and (optionally) up to 11 square matrices. This function processes data using dataprep() function and then feeds the data to analytical pipeline which includes ML.Alpha() and AlphInts(). The outputs of the function in the $all dataframe include the following:

- alpha_mle: maximum likelihood estimate of the log-odds parameter alpha in the Extended Hypergeometric distribution with fixed margins (mA,mB) and table-total N, which is the "log-affinity" index of co-occurrence championed in a paper by Mainali et al. (2022) as an index of co-occurrence-based similarity; computed in ML.Alpha()

- exp_cooccur: expected co-occurrence count under the null (hypergeometric, corresponding to alpha=0) distribution; computed as ML.Alpha()$Null.Exp
- p_value: the commonly reported P-value of the observed co-occurrences; computed by AlphInts()$pval
- alpha_medianInt: the interval of alpha values compatible with x as median for the Extended Hypergeometric distribution (Harkness 1965)
  with fixed margins and alpha; computed in AlphInts() as $MedianIntrvl
- conf_level: confidence level for estimating the various types of confidence intervals
- ci_: fout types of confidence intervals (see details below)
- jaccard: Jaccard index
- sorensen: Sørensen-Dice index
- simpson: Simpson index

Usage

```
affinity(
  data,
  row.or.col,
  which.row.or.col = NULL,
  datatype = NULL,
  threshold = NULL,
  class0.rule = NULL,
  sigPval = NULL,
  sigdigit = NULL,
  squarematrix = NULL,
  ...
)
```

Arguments

`data`	occurrence matrix (binary or abundance) in matrix or dataframe format
`row.or.col`	specify if the pairs of rows or columns are analyzed for affinity. 'row' or 'column'.
`which.row.or.col`	a vector of name or the number of row/column if a subset of the data is intended to be analyzed; optional argument with default of all rows/columns.
`datatype`	specify if the datatype is 'abundance' or 'binary'; optional argument with default 'binary'.

`threshold`	cutoff for converting an abundance data to binary; needed if datatype is 'abundance'
`class0.rule`	'less.or.equal' or 'less'. 'less.or.equal' converts a threshold or lower values to zero and all the others to 1. 'less' converts a threshold and higher values to 1.
`sigPval`	acceptable rate of false positives or probability of rejecting the null hypothesis when it is true, commonly known as alpha in hypothesis testing
`sigdigit`	the number of decimals for rouding alpha mle, its intervals, expected cooccurence under the null, jaccard, sorensen and simpson indices
`squarematrix`	a vector of quantities so that a square matrix for each of them is generated on the top of the main long matrix of all outputs. "alpha_mle", "alpha_mle_sig", "p_value", "cooccur.null", "cooccur.obs", "jaccard", "jaccard_sig", "sorensen", "sorensen_sig", "simpson", "simpson_sig", "all".
`...`	Additional arguments to control behavior of the function.

Details

This function calculates "alpha_mle", which is the maximum likelihood estimate of the log-odds paramater alpha within the Extended Hypergeometric distribition (Harkness 1965) based on the count x and fixed table margins (mA,mB) and total N, which is the "affinity" index of co-occurrence championed in the paper of Mainali et al. (2022) as an index of cooccurrence-based similarity.

This function calculates five intervals, three of them using EHypQuInt, one using EHypMidP, and one using AcceptAffCI. First ("alpha_medianInt") is the interval of alpha values compatible with x as median for the Extended Hypergeometric distribution (Harkness 1965) with fixed margins and alpha. Computed as AlphInts()$MedianIntrvl. This interval quantifies the underlying discreteness of the Extended Hypergeometric and its impact on the estimation of alpha. MedianIntrvl is an interval that will contain the MLE alpha-hat, and the mid-point of that interval is another reasonable estimator of alpha from the data.

There are four confidence intervals computed in AlphInts(), called ci_cp, ci_blaker, ci_midQ, ci_midP, matching name of the outputs in standalone outputs of AlphInts(), except the differences in capital/small letters. The boolean "bound" parameter is an option to prevent the intervals containing alpha-estimates to extend to plus or minus infinity, based on a Bayesian argument. The bound substituted for the Infinite endpoints is provably larger than the largest value the MLE can take whenever x avoids the enpoints max(mA+mB-N,0) and min(mA,mB) of its logical range. The recommended confidence interval for alpha is CI.Blaker if a reliably conservative (over-large) coverage probability is desired, and CI.midP otherwise.

"ci_cp", computed as AlphInts()$CI.CP is an "exact" conservative test-based 2-sided confidence interval (analogous to the Clopper-Pearson (1934) confidence interval for unknown binomial proportion) for alpha based on data (x,mA,mB,N)

"ci_blaker", computed as AlphInts()$CI.Blaker is the Acceptability Confidence Interval of Blaker (2000, Theorem 1) which is a better confidence interval than the CP-type interval "CI.CP" in the sense of being contained within "CI.CP" but still probably conservative, i.e., with coverage probability always at least as large as the nominal level.

"ci_midQ", computed as AlphInts()$CI.midQ has the endpoints obtained as the midpoints of quantile intervals respectively to the (1+lev)/2 and (1-lev)/2 quantiles of the Extended Hypergeometric distribution.

"ci_midP", computed as AlphInts()$CI.midQ, behaves very similarly to "CI.midQ" and is defined by the midP approach analogous to the midP confidence interval for binomial proportions (Agresti 2013, p.605), and is calculated from EHypMidP().

The recommended (slightly conservative) confidence interval is CI.Blaker, while the very similar intervals CI.midQ and CI.midP have coverage generally closer than CI.CP or CI.Blaker to the nominal level of coverage, at the cost of occasionally under-covering by as much as 0.04 or 0.05 for confidence levels 0.90 or 0.95. The comparison among intervals, and different possible goals that CIs of conservative or close-to-nominal coverage can serve, are similar to those compared by Brown et al. (2001) for interval estimation of an unknown binomial proportion.

"p_value" is the two-sided p-value for the equal-tailed test of the null hypothesis alpha=0. This p-value is calculated when pval="Blaker" according to Blaker's (2000) "Acceptability" function; if the input parameter pvalType of AlphInts() is anything else, the p-value is calculated using the same idea as the midP confidence interval.

ADDITIONAL ARGUMENTS can be supplied from ML.Alpha() and AlphInts().

Value

This function returns one main long dataframe ($all) with various outputs in columns (a list given under "details") for each of the pairs of the entities in row. This function also outputs optionally upto 11 square matrices of NxN entities.

Author(s)

Kumar Mainali

References

Agresti, A. (2013) Categorical Data Analysis, 3rd edition, Wiley.

Blaker, H. (2000), “Confidence curves and improved exact confidence intervals for discrete distributions", Canadian Journal of Statistics 28, 783-798.

Brown, L., T. Cai, and A. DasGupta (2001), “Interval Estimation for a Binomial Proportion,” Statistical Science, 16, 101–117.

Clopper, C., and E. Pearson (1934), “The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial,” Biometrika, 26, 404–413.

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Mainali, K., Slud, E., Singer, M. and Fagan, W. (2022), "A better index for analysis of co-occurrence and similarity", Science Advances, to appear.

Examples

```
# when you have a binary presence absence occurrence data
# -----------------------------------------------------

data(finches)
head(finches)
```

```
# this dataset carries the occurrence records of 13 species in row in 17 islands in columns
dim(finches)


# the remainder of the script has been enclosed under \donttest{}
# to bypass the CRAN's 5 second limit on example files
# --------------------------------------------------------------




  # compute alpha and other quantities for island-pair affinity (beta diversity)
  myout <- affinity(data = finches, row.or.col = "col")
  myout

  # you can simply flip the analysis to rows to compute species-pair affinity
  myout <- affinity(data = finches, row.or.col = "row")
  myout


  # the rows of the outputs above include every single pair of the entities,
  # producing many columns for various quantities.
  # # can output an NxN square matrix for selected columns.
  # an example is here
 myout <- affinity(data = finches, row.or.col = "col", squarematrix = c("alpha_mle", "jaccard"))
  # it is a list of three elements: one main dataframe and two square matrices
  length(myout)
  myout
  head(myout)

  # you can also compute all the square matrices with an "all"
  myout <- affinity(data = finches, row.or.col = "col", squarematrix = c("all"))
  # this one has 12 elements
  length(myout)
  myout

  # when you want to compute for only certain pairs
  myout <- affinity(data = finches, row.or.col = "col", which.row.or.col = 4:6,
                    squarematrix = c("alpha_mle"))
  myout

  myout <- affinity(data = finches, row.or.col = "col",
              which.row.or.col = c("Isabella", "Espanola"), squarematrix = c("alpha_mle"))
  print(myout)

 #end of \donttest{}



# if you select only one column, the computation stops
## Not run:
  myout <- affinity(data = finches, row.or.col = "col",
                    which.row.or.col = c("Darwin"), squarematrix = c("alpha_mle"))
```

```
## End(Not run)
```

```
  # you can also add additional arguments bringing them from ML.Alpha() or AlphInts()
  myout1 <- affinity(data = finches, row.or.col = "col",
               which.row.or.col = c("Isabella", "Espanola"), lev=0.95, pvalType="Blaker")
  myout1
  myout2 <- affinity(data = finches, row.or.col = "col",
               which.row.or.col = c("Isabella", "Espanola"), lev=0.90, pvalType="Blaker")
  myout2
  identical(myout1, myout2)
  # myout1 and myout2 were generated with identical arguments except a difference in "lev",
  # which gave different confidence intervals

  myout3 <- affinity(data = finches, row.or.col = "col",
                     which.row.or.col = 4:6, lev=0.95, pvalType="Blaker")
  myout3
  myout4 <- affinity(data = finches, row.or.col = "col",
                     which.row.or.col = 4:6, lev=0.95, pvalType="midP")
  myout4
  myout3$all$p_value
  myout4$all$p_value
  # the p values are (or, can be) different

  # when you have abundance data requiring conversion to binary
  # -------------------------------------------------------------
  # abundance data is converted to binary based on a threshold supplied.
  # it might be a good idea to explore dataprep() function and its examples
  # first before workign on affinity() for abundance data.
  matrix.data <- matrix(runif(400, 0, 10), nrow = 100, ncol = 4)
  row.names(matrix.data) <- paste0("id_", 1:nrow(matrix.data))
  colnames(matrix.data) <- paste0("variable_", 1:ncol(matrix.data))

  # add some missing data and zero abundance
  matrix.data[1,1] <- matrix.data[2,3] <- matrix.data[1,4] <- matrix.data[1,2] <- NA
  matrix.data[10,4] <- 0
  head(matrix.data)
  # now this is an abundance data with some missing and some zero occurrences

  # inspecting how the abundance is converted to binary first
  dataprep(data = matrix.data, row.or.col = "col", datatype = "abundance",
           threshold = 5, class0.rule = "less")
  myout10 <- affinity(data = matrix.data, row.or.col = "col",
                      datatype = "abundance", threshold = 5, class0.rule = "less")
  myout10

  # you can also feed the output of dataprep() to affinity()
  myinput <- dataprep(data = matrix.data, row.or.col = "col",
                      datatype = "abundance", threshold = 5, class0.rule = "less")
  myout11 <- affinity(data = myinput, row.or.col = "col", datatype = "binary")
```

```
  myout11
  # myout 10 and myout11 are identical
  identical(myout10, myout11)

 # end of \donttest{}
```

`affinity2by2` *Maximum likelihood estimate and intervals of alpha, null expectation, p-value and traditional indices from a 2x2 table*

Description

This function uses ML.Alpha() and supplements to the outcome with traditional indices of Jaccard, Sorenson, and Simpson. ML.Alpha() calculates the maximum likelihood estimate and other quantities computed in AlphInts(), for the log-odds parameter alpha in the Extended Hypergeometric distribution with fixed margins (mA,mB) and table-total N, which is the "log-affinity" index of co-occurrence championed in a paper by Mainali et al. (2022) as an index of co-occurrence-based similarity.

Usage

```
affinity2by2(
  x,
  marg,
  bound = TRUE,
  scal = log(2 * marg[3]^2),
  lev = 0.95,
  pvalType = "Blaker"
)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`bound`	a boolean parameter which when TRUE replaces the MLE of "+/-Infinity", applicable when x is respectively at the upper extreme min(mA,mB) or the lower extreme max(mA+mB-N,0) of its possible range, by a finite value with absolute value upper-bounding the value of MLEs attainable for values of x not equal to its extremes
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)

`pvalType`	a character string telling what kind of p-value to calculate. ‘Blaker’ or “midP’. If ‘pvalType=Blaker” (the default value), the p-value is calculated according to "Acceptability" function of Blaker (2000). If ‘pvalType=midP’, the p-value is calculated using the same idea as the midP confidence interval.

Details

See the details of ML.Alpha(). In addition to the output of ML.Alpha, this function also computes Jaccard, Sorenson and Simpson indices.

Value

This function returns maximum likelihood estimate of alpha, the interval-endpoints of alpha values for which x is a median, four confidence intervals for alpha, described in detail under documentation for AlphInts(), and traditional indices of Jaccard, Sorenson and Simpson. In addition there are two output list-components for the null-distribution expected co-occurrence count and the p-value for the test of the null hypothesis alpha=0, calculated as in AlphInts.

Author(s)

Kumar Mainali and Eric Slud

References

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Mainali, K., Slud, E., Singer, M. and Fagan, W. (2022), "A better index for analysis of co-occurrence and similarity", Science Advances, to appear.

Examples

```
ML.Alpha(x=35, c(mA=50, mB=70, N=150), lev=0.95)
affinity2by2(x=35, c(mA=50, mB=70, N=150), lev=0.95)
# ML.Alpha() is a subset of affinity2by2()
a <- ML.Alpha(x=35, c(mA=50, mB=70, N=150), lev=0.95)
b <- affinity2by2(x=35, c(mA=50, mB=70, N=150), lev=0.95)
identical(a, b[1:11])
```

`AlphInts` *Median interval, four confidence intervals, null expectation of cooccurrence count, and p-value*

Description

This function calculates (i) MedianIntrvl, the interval of alpha values for which the co-occurrence count is a median, (ii) four Confidence Intervals, two using EHypQuInt(), one using EHypMidP(), and one using AcceptAffCI(), (iii) the Expected Co-occurrence count under the Null distribution, and (iv) the p-value for the observed co-occurrence count.

Usage

```
AlphInts(x, marg, scal = log(2 * marg[3]^2), lev = 0.95, pvalType = "Blaker")
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)
`pvalType`	a character string telling what kind of p-value to calculate. 'Blaker' or "midP'. If 'pvalType=Blaker" (the default value), the p-value is calculated according to "Acceptability" function of Blaker (2000). If 'pvalType=midP', the p-value is calculated using the same idea as the midP confidence interval.

Details

This function calculates five intervals, three of them using EHypQuInt, one using EHypMidP, and one using AcceptAffCI. First ("MedianIntrvl") is the interval of alpha values compatible with x as median for the Extended Hypergeometric distribution (Harkness 1965) with fixed margins and alpha; second ("CI.CP") an "exact" conservative test-based 2-sided confidence interval (analogous to the Clopper-Pearson (1934) confidence interval for unknown binomial proportion) for alpha based on data (x,mA,mB,N); third the Acceptability Confidence Interval ("CI.Blaker") of Blaker (2000, Theorem 1) which is a better confidence interval than the CP-type interval "CI.CP" in the sense of being contained within "CI.CP" but still provably conservative, i.e., with coverage probability always at least as large as the nominal level. The fourth confidence interval ("CI.midQ") is the one given in formula (2) above of the Introduction to this documentation, with endpoints obtained as the midpoints of quantile intervals respectively to the (1+lev)/2 and (1-lev)/2 quantiles of the Extended Hypergeometric distribution; and the fifth ("CI.midP") which behaves very similarly to "CI.midQ" is defined by the midP approach analogous to the midP confidence interval for binomial proportions (Agresti 2013, p.605), and is calculated from EHypMidP.

The first of these intervals quantifies the underlying discreteness of the Extended Hypergeometric and its impact on the estimation of alpha. MedianIntrvl is an interval that will contain the MLE alpha-hat, and the mid-point of that interval is another reasonable estimator of alpha from the data. The recommended (slightly conservative) confidence interval is CI.Blaker, while the very similar intervals CI.midQ and CI.midP have coverage generally closer than CI.CP or CI.Blaker to the nominal level of coverage, at the cost of occasionally under-covering by as much as 0.04 or 0.05 for confidence levels 0.90 or 0.95. The comparison among intervals, and different possible goals that CIs of conservative or close-to-nominal coverage can serve, are similar to those compared by Brown et al. (2001) for interval estimation of an unknown binomial proportion.

Two other output list components are computed. First is Null.Exp, the expected co-occurrence count under the null (hypergeometric, corresponding to alpha=0) distribution, and second is the two-sided p-value for the equal-tailed test of the null hypothesis alpha=0. This p-value is calculated when pval="Blaker" according to Blaker's (2000) "Acceptability" function; if the input parameter pval is anything else, the p-value is calculated using the same idea as the midP confidence interval.

Value

A list of seven components: the median interval MedianIntrvl; the four two-sided Confidence Intervals described above, two (CI.CP and CI.Blaker) conservative and two (CI.midQ and CI.midP) with coverage probabilities generally closer to the nominal level; the null expectation Null.Exp of the co-occurrence count associated with alpha=0; and pval, the two-sided p-value for the hypothesis test of alpha=0, calculated by the method selectied, which is the Blaker acceptability-function method if pvalType="Blaker" and otherwise the "midP" p-value associated with the midP confidence-interval type.

Of the four Confidence intervals produced, CI.Blaker is the recommended conservative interval and CI.midP the interval to use if coverage close to the nominal is desired.

Author(s)

Eric Slud

References

Agresti, A. (2013) Categorical Data Analysis, 3rd edition, Wiley.

Blaker, H. (2000), “Confidence curves and improved exact confidence intervals for discrete distributions", Canadian Journal of Statistics 28, 783-798.

Brown, L., T. Cai, and A. DasGupta (2001), “Interval Estimation for a Binomial Proportion,” Statistical Science, 16, 101–117.

Clopper, C., and E. Pearson (1934), “The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial,” Biometrika, 26, 404–413.

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Examples

```
unlist(AlphInts(30,c(50,80,120), lev=0.9))

AlphInts(30,c(50,80,120), lev=0.9)$CI.CP
AlphInts(30,c(50,80,120), lev=0.9)$MedianIntrvl

EHypMidP(30,c(50,80,120), 0.9)
AlphInts(30,c(50,80,120), lev=0.9)$CI.midP
# NB the third argument of AlphInts is "scal" if not named,
# so must use "lev=0.9" to define the confidence level.

EHypQuInt(30,c(50,80,120), 0.5)
AlphInts(30,c(50,80,120), lev=0.9)$MedianIntrvl

# Alpha capped warning examples
AlphInts(60,c(80,80,100), lev=0.9)
ML.Alpha(60,c(80,80,100), lev=0.9)

AlphInts(80,c(80,80,100), lev=0.9)
```

```
ML.Alpha(80,c(80,80,100), lev=0.9)

# impossible x warning examples
AlphInts(81,c(80,80,100), lev=0.9)
ML.Alpha(81,c(80,80,100), lev=0.9)

# Degenerate distribution warning example
AlphInts(80,c(80,100,100), lev=0.9)
ML.Alpha(80,c(80,100,100), lev=0.9)
```

`Bisect` *Bisections for finding a root of a function*

Description

Find a root of a function by the method of Bisections

Usage

```
Bisect(ffnc, intrv, tol = 1e-08)
```

Arguments

`ffnc`	an increasing function of a single scalar argument
`intrv`	an interval over which the root of ffnc is sought
`tol`	a tolerance determining when the successive bisections of the interval within which the root will lie have become small enough to stop

Details

This function finds the root of the increasing function ffnc over the scalar interval intrv by the Method of Bisections. The function must be increasing but need not be smooth, and it must have a negative sign (value less than -tol) at the left endpoint of intrv and positive sign (value greater than tol) at the right endpoint. The method of Bisection is used in successive iterations to successively halve the width of the interval in which the root lies.

Value

This function returns a vector consisting of two numbers. The first named root is an estimate of the root x solving ffnc(x) = 0, valid within an error of tol. The second output vector element named fval is the value of the function ffnc at root. It should be very close to 0 unless the function happens to jump from a value less than 0 to a value greater than 0 at root.

Author(s)

Eric Slud

References

to be added

Examples

```
Bisect(function(x) x^2-1, c(0,2),1e-8)
```

`Covrg` *Coverage Probabilities for Confidence Intervals about alpha, for fixed true alpha*

Description

This function calculates the coverage probability at the true value alpha of the four types of Cpnfidence Intervals (CI.CP, CI.Blaker, CI.midQ, CI.midP) computed in AlphInts().

Usage

```
Covrg(marg, alph, scal = log(2 * marg[3]^2), lev = 0.95)
```

Arguments

`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`alph`	True log-odds-ratio value alpha at which coverage probabilities (under Extended Hypergeometric with parameters mA,mB,N, exp(alp)) are to be calculated
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)

Details

See AlphInts() documentation for details of computation of the four confidence intervals CI.CP, CI.Blaker, CI.midQ, CI.midP. The confidence intervals are calculated for each x in the allowed range from max(mA+mB-N,0) to min(mA,mB), and the probability that X=x times the indicator of alph falling in each of them is summed.

Value

A vector covPrb containing the coverage propbabilities for the four Confidence Intervals

Author(s)

Eric Slud

References

to be added

Examples

```
Covrg(c(50,70,150), 1.2, lev=0.95)
Covrg(c(50,70,150), 0, lev=0.95)
Covrg(c(50,80,120), 1.5, lev=0.9)
```

`dataprep` *Occurrence matrix (e.g., species by site) data preparation for affinity() function*

Description

This function checks the format of the data for its appropriateness, converts abundance to binary and subsets the data for the selected columns or rows. Note that the affinity can be computed between columns or between rows. In the latter case, the dataset is transposed to bring rows into the columns.

Usage

```
dataprep(
  data,
  row.or.col,
  which.row.or.col = NULL,
  datatype = NULL,
  threshold = NULL,
  class0.rule = NULL
)
```

Arguments

`data`	occurrence matrix (binary or abundance) in matrix or dataframe format
`row.or.col`	specify if the pairs of rows or columns are analyzed for affinity. ’row’ or ’column’.
`which.row.or.col`	a vector of name or the number of row/column if a subset of the data is intended to be analyzed; optional argument with default of all rows/columns.
`datatype`	specify if the datatype is ’abundance’ or ’binary’; optional argument with default ’binary’.
`threshold`	cutoff for converting an abundance data to binary; needed if datatype is ’abundance’
`class0.rule`	’less.or.equal’ or ’less’. ’less.or.equal’ converts a threshold or lower values to zero and all the others to 1. ’less’ converts a threshold and higher values to 1.

Details

This function does the following:

1. checks if the supplied data is in matrix or dataframe formats which are the acceptable formats
2. if rows are selected for affinity analysis, it transposes the dataframe
3. subsets the data if specific columns or rows are selected for analysis; the selection can be made with number or name of the rows/columns
4. checks if the selected cols/rows are in numeric or integer format or not
5. checks if the selected cols/rows have data in binary 1/0 format or not; if datatype is specified as abundance, it converts it to binary format following the supplied rule

Value

A dataframe in binary 1/0 format ready to be analyzed by affinity(). Abundance data is converted to binary. A subset of the input data is returned if certain rows or columns selected. If rows are being analyzed for affinity between pairs, they are brought to columns by transposing the data.

Author(s)

Kumar Mainali

Examples

```
matrix.data <- matrix(1:40, nrow = 10, ncol = 4)

row.names(matrix.data) <- paste0("id_", 1:nrow(matrix.data))
colnames(matrix.data) <- paste0("variable_", 1:ncol(matrix.data))

# add some missing data and zero abundance
matrix.data[1,1] <- matrix.data[2,3] <- matrix.data[1,4] <- matrix.data[1,2] <- NA
matrix.data[10,4] <- 0
matrix.data
# abundance data with some missing and some zero occurrences

# some good examples
dataprep(data = matrix.data, row.or.col = "col", datatype = "abundance",
         threshold = 9, class0.rule = "less")
dataprep(data = matrix.data, row.or.col = "row", which.row.or.col = c("id_2", "id_4"),
         datatype = "abundance", threshold = 10, class0.rule = "less")
dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = c("variable_1", "variable_4"),
         datatype = "abundance", threshold = 8, class0.rule = "less")
dataprep(data = matrix.data, row.or.col = "col",
         which.row.or.col = c("variable_1", "variable_3", "variable_4"),
         datatype = "abundance", threshold = 8, class0.rule = "less.or.equal")
dataprep(data = matrix.data, row.or.col = "row", datatype = "abundance",
         threshold = 10, class0.rule = "less")
dataprep(data = matrix.data, row.or.col = "col", datatype = "abundance",
         threshold = 10, class0.rule = "less")

# bad examples of specifying the rows or cols that are not in the data
```

```
## Not run:
  dataprep(data = matrix.data, row.or.col = "row",
          which.row.or.col = c("id_1", "id_4", "id_11", "id_39"), datatype = "abundance",
           threshold = 10, class0.rule = "less")
  dataprep(data = matrix.data, row.or.col = "row", which.row.or.col = c(4,7,17),
           datatype = "abundance", threshold = 10, class0.rule = "less")
 dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = 2:12, datatype = "abundance",
           threshold = 10, class0.rule = "less")
  dataprep(data = matrix.data, row.or.col = "col",
        which.row.or.col = c("variable_1", "variable_9", "variable_6"), datatype = "abundance",
           threshold = 10, class0.rule = "less")

## End(Not run)


# what if you pick just one column or row
## Not run:
  dataprep(data = matrix.data, row.or.col = "row", which.row.or.col = c("id_4"),
           datatype = "abundance", threshold = 10, class0.rule = "less")

## End(Not run)

# the function fails when a required argument is missing
## Not run:
 dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = c("variable_1", "variable_4"),
           datatype = "abundance", threshold = 10)
 dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = c("variable_1", "variable_4"),
           datatype = "abundance", class0.rule = "less.or.equal")
 dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = c("variable_1", "variable_4"),
           datatype = "abundance")

## End(Not run)

# what if you have abundance data but do not specify the datatype
## Not run:
 dataprep(data = matrix.data, row.or.col = "col", which.row.or.col = c("variable_1", "variable_4"))

## End(Not run)

# however, if it is a binary data, it's okay to not specify the datatype
# although specifying is a good practice
matrix.bindata <- dataprep(data = matrix.data, row.or.col = "col", datatype = "abundance",
                           threshold = 9, class0.rule = "less")
matrix.bindata
dataprep(data = matrix.bindata, row.or.col = "col")
dataprep(data = matrix.bindata, row.or.col = "row")
```

`EHypMidP` *Quantile of the Extended Hypergeometric distribution approximated by the midP distribution function*

Description

This function does the analogous calculation to that of EHypQuInt, but with the Extended Hypergeometric distribution function F(x) = F(x,mA,mB,N, exp(alpha)) replaced by (F(x) + F(x-1))/2.

Usage

```
EHypMidP(x, marg, lev)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)

Details

This function does the analogous calculation to that of CI.CP, but with the Extended Hypergeometric distribution function F(z, alpha) = F(z,mA,mB,N, exp(alpha)) replaced by (F(z,alpha) + F(z-1,alpha))/2.

Value

This function returns the interval of alpha values with endpoints (F(x,alpha)+F(x-1,alpha))/2 = (1+lev)/2 and (F(x,alpha)+F(x+1,alpha))/2 = (1-lev)/2.

The idea of calculating a Confidence Interval this way is analogous to the midP CI used for unknown binomial proportions (Agresti 2013, p.605).

Author(s)

Eric Slud

References

Agresti, A. (2013) Categorical Data Analysis, 3rd edition, Wiley.

Examples

```
EHypMidP(30,c(50,80,120), 0.9)
AlphInts(30,c(50,80,120), lev=0.9)$CI.midP

EHypMidP(20, c(204,269,2016), 0.9)
```

`EHypQuInt` *Interval of alpha values for which X is a specified q'th quantile*

Description

This function outputs the largest interval of log-odds parameter values alpha for which the Extended Hypergeometric distribution function at x is >= q and the complementary distribution function 1 - F(x-) is >= 1-q.

Usage

```
EHypQuInt(x, marg, q, scal = log(2 * marg[3]^2))
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`q`	a quantile falling strictly between 0 and 1
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater

Details

This function outputs the endpoints a1, a2 defined by

F(x, a1) = q and F(x-1, a2) = q

where F(z, a) = F(z, mA,mB,N, exp(a)) is the extended Hypergeometric distribution function.

The interval of alpha values with these endpoints a1, a2 is viewed as the set of alpha values "compatible" with x being a q'th quantile for the Extended Hypergeometric.

Value

This function returns the vector (a1, a2) defined above, the endpoints of the set of alpha values for which x is a q'th quantile of the Extended Hypergeometric distribution.

Author(s)

Eric Slud

Examples

```
EHypQuInt(30,c(50,80,120), 0.95)
EHypQuInt(30,c(50,80,120), 0.05)

EHypQuInt(30,c(50,80,120), 0.5)
AlphInts(30,c(50,80,120), lev=0.9)$MedianIntrvl
```

`finches` *Darwin's finches presence–absence data*

Description

A presence (1)/absence (0) matrix of seven Darwin's finch species across 13 Galápagos Islands.

Usage

`finches`

Format

A data frame with 7 rows (Darwin's finch species, in the row names) and 17 columns (Galápagos Islands):

Baltra presence (1) or absence (0)
Darwin presence (1) or absence (0)
Espanola presence (1) or absence (0)
Fernandina presence (1) or absence (0)
Floreana presence (1) or absence (0)
Genovesa presence (1) or absence (0)
Isabella presence (1) or absence (0)
Marchena presence (1) or absence (0)
Pinta presence (1) or absence (0)
Pinzon presence (1) or absence (0)
Rabida presence (1) or absence (0)
San.Cristobal presence (1) or absence (0)
Santa.Cruz presence (1) or absence (0)
Santa.Fe presence (1) or absence (0)
Santiago presence (1) or absence (0)
Seymour presence (1) or absence (0)
Wolf presence (1) or absence (0)

Details

Row names are the seven species: Geospiza magnirostris, G. fortis, G. fuliginosa, G. difficilis, G. scandens, G. conirostris, and G. fuliginosa.

Source

Sanderson J.G. (2000) Testing ecological patterns: a well-known algorithm from computer science aids the evaluation of species distributions. *American Scientist* 88, 332–339.

Griffith D.M., Veech J.A., Marsh C.J. (2016) *cooccur*: probabilistic species co-occurrence analysis in R. *Methods in Ecology and Evolution* 7, 539–543.

`logLikExtHyp` *log of Extended Hypergeometric Likelihood at (X, mA,mB,N, alpha)*

Description

This function calculates the logarithm of the Extended Hypergeometric likelihood at specified x and alpha, with marginal totals mA, mB, N fixed.

Usage

```
logLikExtHyp(x, marg, alpha)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`alpha`	a real number, the log odds ratio or affinity parameter for the 2x2 contingency table

Details

This is simply the logarithm of the Extended Hypergeometric (Harkness 1965) or Fisher noncentral Hypergeometric, as calculated by the R package BiasedUrn. The formula is log(pFNCHypergeo(x,mA,N-mA,mB,exp(alpha))

Value

scalar loglikelihood value

Author(s)

Eric Slud

References

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Examples

```
require(BiasedUrn)
c(logLikExtHyp(30,c(50,80,120),1),  log(dFNCHypergeo(30,50,70,80,exp(1))))
```

`MaxX.Int` *MaxX.Int computation*

Description

Helper function

Usage

```
MaxX.Int(marg, scal = log(2 * marg[3]^2), lev = 0.95)
```

Arguments

`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)

Details

This is a helper function.

Value

helper function

Author(s)

Eric Slud

`midP.EHyp` *midP.EHyp computation*

Description

Helper function

Usage

```
midP.EHyp(alp)
```

Arguments

`alp`	"alpha" parameter, the log-odds parameter in the Extended Hypergeometric distribution

Details

This is a helper function.

param x integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]

Value

helper function for midP CI computation with EHypMidP

Author(s)

Eric Slud

`minmaxAlpha.pFNCH` *integer-endpoint of range for which BiasedUrn::pFNCHHypergeo() works without error*

Description

This function calculates an integer-endpoint of range for which BiasedUrn::pFNCHHypergeo() works without error.

Usage

```
minmaxAlpha.pFNCH(x, marg)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table

Details

Without this function, BiasedUrn::pFNCHHypergeo() returns inconsistency message for extreme examples like: AlphInts(20,c(204,269,2016), lev=0.9, scal=10). This problem is solved within our package by restricting the range of allowed alpha to the computed (alphmin, alphmax) range.

Value

minimum and maximum of Alpha

Author(s)

Eric Slud

References

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Examples

```
minmaxAlpha.pFNCH(10,c(100,200,300))
minmaxAlpha.pFNCH(20,c(204,269,2016))
minmaxAlpha.pFNCH(20,c(204,269,20160))
```

`MinX.Int` *MinX.Int computation*

Description

Helper function

Usage

```
MinX.Int(marg, scal = log(2 * marg[3]^2), lev = 0.95)
```

Arguments

`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)

Details

This is a helper function.

Value

helper function

Author(s)

Eric Slud

`ML.Alpha` *Maximum likelihood estimate and intervals of alpha, null expectation and p-value of a 2x2 table*

Description

This function calculates the maximum likelihood estimate and other quantities computed in AlphInts(), for the log-odds parameter alpha in the Extended Hypergeometric distribution with fixed margins (mA,mB) and table-total N, which is the "log-affinity" index of co-occurrence championed in a paper by Mainali et al. (2022) as an index of co-occurrence-based similarity.

Usage

```
ML.Alpha(
  x,
  marg,
  bound = TRUE,
  scal = log(2 * marg[3]^2),
  lev = 0.95,
  pvalType = "Blaker"
)
```

Arguments

`x`	integer co-occurrence count that should properly fall within the closed interval [max(0,mA+mB-N), min(mA,mB)]
`marg`	a 3-entry integer vector (mA,mB,N) consisting of the first row and column totals and the table total for a 2x2 contingency table
`bound`	a boolean parameter which when TRUE replaces the MLE of "+/-Infinity", applicable when x is respectively at the upper extreme min(mA,mB) or the lower extreme max(mA+mB-N,0) of its possible range, by a finite value with absolute value upper-bounding the value of MLEs attainable for values of x not equal to its extremes
`scal`	an integer parameter (default 2*N^2, capped at 10 within the function) that should be 2 or greater
`lev`	a confidence level, generally somewhere from 0.8 to 0.95 (default 0.95)
`pvalType`	a character string telling what kind of p-value to calculate. ‘Blaker’ or “midP’. If ‘pvalType=Blaker” (the default value), the p-value is calculated according to "Acceptability" function of Blaker (2000). If ‘pvalType=midP’, the p-value is calculated using the same idea as the midP confidence interval.

Details

This function calculates the maximum likelihood estimate of the log-odds paramater alpha within the Extended Hypergeometric distribition (Harkness 1965) based on the count x and fixed table margins (mA,mB) and total N, which is the "affinity" index of co-occurrence championed in the paper of Mainali et al. (2022) as an index of cooccurrence-based similarity, along with the intervals computed in AlphInts, called CI.CP, CI.Balker, CI.midQ and CI.midP. The boolean "bound" parameter is an option to prevent the intervals containing alpha-estimates to extend to plus or minus infinity, based on a Bayesian argument. T he bound substituted for the Infinite endpoints is provably larger than the largest value the MLE can take whenever x avoids the enpoints max(mA+mB-N,0) and min(mA,mB) of its logical range. The recommended confidence interval for alpha is CI.Blaker if a reliably conservative (over-large) coverage probability is desired, and CI.midP otherwise.

Value

This function returns maximum likelihood estimate of alpha, the interval-endpoints of alpha values for which x is a median, and four confidence intervals for alpha, described in detail under documentation for AlphInts(). In addition there are two output list-components for the null-distribution expected co-occurrence count and the p-value for the test of the null hypothesis alpha=0, calculated as in AlphInts.

Author(s)

Eric Slud

References

Fog, A. (2015), BiasedUrn: Biased Urn Model Distributions. R package version 1.07.

Harkness, W. (1965), “Properties of the extended hypergeometric distribution“, Annals of Mathematical Statistics, 36, 938-945.

Mainali, K., Slud, E., Singer, M. and Fagan, W. (2022), "A better index for analysis of co-occurrence and similarity", Science Advances, to appear.

Examples

```
unlist(ML.Alpha(30,c(50,80,120), lev=0.9))
AlphInts(30,c(50,80,120), lev=0.9)

AlphInts(61,c(80,80,100), lev=0.9)
ML.Alpha(61,c(80,80,100), lev=0.9)

# Alpha capped warning examples
AlphInts(60,c(80,80,100), lev=0.9)
ML.Alpha(60,c(80,80,100), lev=0.9)

AlphInts(80,c(80,80,100), lev=0.9)
ML.Alpha(80,c(80,80,100), lev=0.9)

# impossible x warning examples
AlphInts(81,c(80,80,100), lev=0.9)
ML.Alpha(81,c(80,80,100), lev=0.9)

# Degenerate distribution warning example
AlphInts(80,c(80,100,100), lev=0.9)
ML.Alpha(80,c(80,100,100), lev=0.9)
```

`plotgg` *Heatmap plot of affinity() output*

Description

This function works on the output of `affinity` and uses `ggplot2::ggplot()` to generate a heatmap for numeric columns of the `$all` dataframe, excluding interval columns (median interval and confidence intervals) and the confidence level (which is constant across pairs in a single run).

Usage

```
plotgg(
  data,
  variable,
  legendlimit,
  col = NULL,
  show.value = NULL,
  value.digit = NULL,
  text.size = NULL,
  text.col = NULL,
  plot.margin = NULL,
  drop.empty = TRUE,
  sig.only = FALSE,
  ...
)
```

Arguments

`data`	Output list returned by `affinity`.
`variable`	Name of a numeric column in `data$all` to plot.
`legendlimit`	Either `"datarange"` or `"balanced"`.
`col`	Color specification for the fill scale. For `alpha_mle` (diverging), supply `c(low, high)` or `c(low, mid, high)`. For all other variables (sequential), supply `c(low, high)`. If `NULL`, defaults are used (including an auto-generated low color for sequential scales).
`show.value`	Logical; if `TRUE`, values are printed on tiles. If `NULL`, values are printed automatically when the number of plotted entities is ≤ 20.
`value.digit`	Number of digits used when printing values; default is 2.
`text.size`	Size of printed values; default is 2.5.
`text.col`	Color of printed values on tiles (used when values are shown).
`plot.margin`	Plot margin passed to `ggplot2::theme(plot.margin = ...)`. Typically a `ggplot2::margin(t, r, b, l, unit)` object.
`drop.empty`	Logical; if `TRUE` (default), entities whose values are all `NA` for the selected `variable` are removed from the plot. Set to `FALSE` to keep all entities.

`sig.only`	Logical or numeric. If `FALSE` (default), all values are plotted. If `TRUE`, tiles are masked to `NA` wherever `p_value > 0.05`. If numeric, the value is used as the p-value cutoff (e.g., `sig.only = 0.01`). Requires a `p_value` column in `data$all`. When `variable = "p_value"`, p-values above the cutoff are masked to `NA`.
`...`	Additional arguments (currently unused).

Details

This function is a wrapper around `ggplot2` with carefully chosen defaults to generate an interpretable heatmap of pairwise associations. The plot shows the lower triangle of an $N \times N$ matrix (diagonal excluded), where both rows and columns represent the same set of entities. The upper triangle is omitted because it is a mirror image of the lower triangle.

By default (`drop.empty = TRUE`), entities whose values are entirely `NA` for the selected `variable` are removed from both axes. This avoids plotting empty rows and columns when an entity has no usable values (e.g., due to degenerate distributions or missing data). Set `drop.empty = FALSE` to retain all entities and reproduce the full grid, including empty rows or columns.

If `sig.only` is enabled, values of the selected `variable` are masked to `NA` wherever `p_value` exceeds the specified cutoff, so only statistically significant tiles are shown. Use `sig.only = TRUE` to apply the default cutoff (0.05), or supply a numeric cutoff (e.g., `sig.only = 0.01`). Requires a `p_value` column in `data$all`. When `variable = "p_value"`, p-values above the cutoff are masked to `NA`.

Legend titles are mapped to human-readable labels (some shown on two lines), rather than using raw column names from `data$all`.

The plot can be requested using column names from the `$all` dataframe returned by `affinity`. Additional `ggplot2` layers or theme modifications can be added by appending them with `+`, as in standard `ggplot2` usage.

The `legendlimit` argument controls how the color scale is defined. For `alpha_mle`, the default midpoint is 0 (null expectation), and the color scale can be either data-driven (`"datarange"`) or symmetrically balanced around zero (`"balanced"`), using the maximum absolute value observed. For indices bounded in $[0, 1]$ (`p_value`, `jaccard`, `sorensen`, `simpson`), the balanced scale uses fixed limits $[0, 1]$. For `p_value`, the color mapping is reversed so smaller p-values appear more intense. For count-based variables, no natural midpoint exists; the color scale spans the observed range. For `obs_cooccur_X` and `exp_cooccur`, a shared color scale is applied so the two plots are visually comparable.

When `show.value = TRUE`, numeric values are printed on each tile using `ggplot2::geom_text()`. If `show.value = NULL` (default), values are printed automatically when the number of plotted entities is ≤ 20. Rounding and text appearance are controlled by `value.digit`, `text.size`, and `text.col`.

Value

A heatmap plot generated with `ggplot2`.

Author(s)

Kumar Mainali

See Also

affinity

Examples

```
data(finches)
head(finches)

library(ggplot2)


# the remainder of the script has been enclosed under \donttest{}
# to bypass the CRAN's 5 second limit on example files
# --------------------------------------------------------------




  # plotting various variables
  # ---------------------------------------------
  # compute alpha and other quantities for island-pair affinity (beta diversity)
  # the square matrices are not used for plotting
  myout <- affinity(data = finches, row.or.col = "col")
  # myout

  plotgg(data = myout, variable = "alpha_mle", legendlimit = "datarange")
  # in the example above, null expectation of the alpha_mle (=0) has white color,
 # and negative and positive values stretch between "#87beff" and "#fd6a6c", respectively
  # so that the color spectrum is applied NOT to the range of data
  # but to the same extent of values
  # on both sides of zero, which is max(abs(valrange)) and -(max(abs(valrange))).
  # however, the legend can be printed to show the extent of data with "datarange"
  # or the entire spectrum where the color is applied with "balanced".
  plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced")
  # notice that the two plots above are identical but the legend has
  # different range with the same color scale.


  plotgg(data = myout, variable = "sorensen", legendlimit = "balanced")
  plotgg(data = myout, variable = "jaccard", legendlimit = "balanced")

 # in the case of observed and expected cooccurrences, one color scale is applied for both plots
  # so that the shades of color across plots can be visually compared
  plotgg(data = myout, variable = "exp_cooccur", legendlimit = "datarange")
  plotgg(data = myout, variable = "exp_cooccur", legendlimit = "balanced")
  plotgg(data = myout, variable = "obs_cooccur_X", legendlimit = "balanced")

  plotgg(data = myout, variable = "entity_1_count_mA", legendlimit = "datarange")
  plotgg(data = myout, variable = "entity_2_count_mB", legendlimit = "datarange")
  plotgg(data = myout, variable = "total_N", legendlimit = "datarange")
  # for "entity_1_count_mA", "entity_2_count_mB", "sites_total_N",
  # if legendlimit is set to "balanced", it will be changed to "datarange"
  plotgg(data = myout, variable = "entity_2_count_mB", legendlimit = "balanced")
```

```
# plot only statistically significant tiles (based on p_value)
# -----------------------------------------------------------
# sig.only = TRUE masks non-significant tiles (p_value > 0.05) to NA
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced", sig.only = TRUE)

# you can also supply a stricter p-value cutoff (e.g., 0.01)
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced", sig.only = 0.01)


# change color of the plot and text
# ---------------------------------------------
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced")
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced",
       col = c('#99cc33', 'black', '#ff9933'), text.col = "white")
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced",
       col = c('#99cc33', '#ff9933'), text.col = "white")

plotgg(data = myout, variable = "obs_cooccur_X", legendlimit = "balanced")
plotgg(data = myout, variable = "obs_cooccur_X", legendlimit = "balanced",
       col = c('black', 'red'), text.col = "white")


# change the characteristics of text printed in the plot
# ------------------------------------------------------
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced")

# change the number of digits; the default is 2
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced", value.digit = 3)

# make the fonts bigger; the default is 2.5
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced", text.size = 3.5)

# hide values from the plot
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced", show.value = FALSE)


# increase or decrease margin
# ---------------------------------------------
myout <- affinity(data = finches, row.or.col = "row")
# myout

plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced")
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced",
       plot.margin = ggplot2::margin(1,1,5,2, "cm"))


# change angle of x-axis tick label; the default is 35 degrees
# -----------------------------------------------------------
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced")
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced") +
  ggplot2::theme(axis.text.x = element_text(angle = 45))
```

```
# to change to 90 degrees, adjust vjust
# bad ->
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced") +
  ggplot2::theme(axis.text.x = element_text(angle = 90))
# good ->
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced") +
  ggplot2::theme(axis.text.x = element_text(angle = 90, vjust = 0.5))


# additional elements in the plot
# ----------------------------------
# because it is ggplot output, you can use the arguments of ggplot() to make changes

# add plot title and change legend title
plotgg(data = myout, variable = "alpha_mle", legendlimit = "balanced") +
  ggplot2::theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
  ggplot2::ggtitle("Affinity of island pairs measured with Alpha MLE") +
  ggplot2::labs(fill = 'My Personal\nTitle')



# show/hide entities that are entirely empty (all-NA tiles)
# -------------------------------------------------------
# Here we create an artificial "empty" entity by setting one column to NA.
# This guarantees that all pairwise comparisons involving that entity have no usable data,
# so the corresponding tiles become NA for variables such as alpha_mle.

finches2 <- as.matrix(finches)
storage.mode(finches2) <- "numeric"
finches2[, 3] <- NA_real_  # make the first entity entirely missing (choose any column)
myout2 <- affinity(data = finches2, row.or.col = "col")

# Default behavior: drop.empty = TRUE (empty entity removed from the axes)
plotgg(data = myout2, variable = "alpha_mle", legendlimit = "balanced")

# Keep empty entities (legacy/full grid): shows the empty row/column
plotgg(data = myout2, variable = "alpha_mle", legendlimit = "balanced", drop.empty = FALSE)

# keep empty entities even after masking (shows rows/columns with all-NA tiles)
plotgg(data = myout2, variable = "alpha_mle", legendlimit = "balanced",
       sig.only = TRUE, drop.empty = FALSE)


# automatic suppression of numeric values on tiles
# ------------------------------------------------
# By default, numeric values are printed on tiles only when the number of
# plotted entities is reasonably small (<= 20). This avoids severe visual
# clutter when the heatmap becomes large.

finches_big <- finches

# duplicate columns to artificially inflate the number of entities
```

```
 finches_big <- cbind(finches_big, finches_big[, 1:5])
 colnames(finches_big)[(ncol(finches) + 1):ncol(finches_big)] <-
   paste0(colnames(finches)[1:5], "_dup")

 myout_big <- affinity(data = finches_big, row.or.col = "col")

 # Numeric values are NOT printed because the number of entities exceeds 20
 plotgg(data = myout_big, variable = "alpha_mle", legendlimit = "balanced")

 # To force printing numeric values despite the large number of entities:
plotgg(data = myout_big, variable = "alpha_mle", legendlimit = "balanced", show.value = TRUE)


#end of \donttest{}
```

Index