Package ‘cfa’

May 1, 2017

Description  Analysis of configuration frequencies for simple and repeated measures, multiple-samples CFA, hierarchical CFA, bootstrap CFA, functional CFA, Kieser-Victor CFA, and Lindner’s test using a conventional and an accelerated algorithm.

Title  Configural Frequency Analysis (CFA)

Version  0.10-0

Date  2017-04-29

Depends  R (>= 3.0.1)

Suggests  parallel

License  GPL (>= 2)

NeedsCompilation  yes

LazyData  true

Author  Patrick Mair [aut, cre],
        Stefan Funke [aut],
        Joachim Harloff [ctb],
        Alexander von Eye [ctb]

Maintainer  Patrick Mair <mair@fas.harvard.edu>

Repository  CRAN

Date/Publication  2017-05-01 16:18:32 UTC

R topics documented:

bcfa ................................................. 2
  cfa .................................................. 3
  fCFA ................................................ 7
  hcfa ............................................... 8
  mcfaf ............................................. 10
  plot.bcfa .................................... 11
  plot.hcfa .................................... 12
  plot.mcfaf .................................. 13
  plot.scfa .................................... 15
  print.bcfa .................................... 16
  print.hcfa .................................... 17
The bootstrap-CFA tries to replicate the pattern of significant configurations by re-sampling.

Usage

\[ \text{bcfa(configs, cnts, runs=100, sig.item="sig.z",...)} \]

Arguments

- **configs**: Contains the configurations. This can be a dataframe or a matrix. The dataframe can contain numbers, characters, factors, or booleans. The matrix can consist of numbers, characters or booleans (factors are implicitly re-converted to numerical levels). There must be >=3 columns.

- **cnts**: Contains the counts for the configuration. If it is set to NA, a count of one is assumed for every row. This allows untabulated data to be processed. \texttt{cnts} must be a vector.

- **runs**: Number of samples to be drawn.

- **sig.item**: Indicator of significance in the result table (\texttt{sig.z},\texttt{sig.chisq},\texttt{sig.perli},\texttt{sig.zl}, \texttt{sig.zl.corr}). Do not forget to set the proper parameters for the CFA if \texttt{sig.perli},\texttt{sig.zl} or \texttt{sig.zl.corr} are to be used!

- ... Parameters to be to relayed to the CFA

Details

Takes 'runs' samples and does as many CFAs while counting how many times this configuration was considered to be significant.

Repeated-measures CFAs (mcfa) are not provided.

This is a heuristic method rather than a strict test of significance since there is no adjustment for multiple testing whatsoever. The advantage is a more reliable picture compared to splitting the original data, doing a CFA, and checking if the configurations re-appear in a CFA with the other half of the data.
Value

cnt.antitype    Number of antitypes
cnt.type        Number of types
pct.types       Number of types in percent
cnt.sig         Number of significant results
pct.cnt.sig     Number of significant results in percent

Note

cfa() performs many CFAs which are by themselves slow, so the execution can be very time-
consuming, especially if a sufficiently high value for runs was selected

Author(s)

Stefan Funke <s.funke@t-online.de>

References

Psychologie und Medizin, Beltz Psychologie Verlagsunion

See Also

cfa, scfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B")c(rbinom(250,1,0.3)+1),c("C","D")c(rbinom(250,1,0.1)+1),
c("E","F")c(rbinom(250,1,0.3)+1),c("G","H")c(rbinom(250,1,0.1)+1))

counts<-trunc(runif(250)*10)
bcfa(configs,count,runs=25)

cfa                        Analysis of configuration frequencies

Description

This is the main function which will call scfa() und mcfa() as required to handle the simple and the
multiple cfa.
Usage

cfa(cfg, cnts=NA, sorton="chisq", sort.descending=TRUE, format.labels=TRUE,
casewise.delete.empty=TRUE,
binom.test=FALSE, exact.binom.test=FALSE, exact.binom.limit=10,
perli.correct=FALSE, lehmacher=FALSE, lehmacher.corr=TRUE,
alpha=0.05, bonferroni=TRUE)

Arguments

cfg Contains the configurations. This can be a dataframe or a matrix. The dataframe
contains numbers, characters, factors, or booleans. The matrix can consist of
numbers, characters, or booleans (factors are implicitly re-converted to numerical
levels). There must be >=3 columns.

cnts Contains the counts for the configuration. If it is set to NA, a count of one is
assumed for every row. This allows untabulated data to be processed. cnts can
be a vector or a matrix/dataframe with >=2 columns.

sorton Determines the sorting order of the output table. Can be set to chisq, n, or
label.
sort.descending Sort in descending order
format.labels Format the labels of the configuration. This makes output wider but it will
increase the readability.
casewise.delete.empty If set to TRUE all configurations containing a NA in any column will be deleted.
Otherwise NA is handled as the string "NA" and will appear as a valid configu-
ration.

binom.test Use z approximation for binomial test.
exact.binom.test Do an exact binomial test.
exact.binom.limit Maximum n for which an exact binomial test is performed (n >10 causes p to
become inexact).
perli.correct Use Perli's correction for multiple test.
lehmacher Use Lehmacher's correction for multiple test.
lehmacher.corr Use a continuity correction for Lehmacher's correction.
alpha Alpha level
bonferroni Do Bonferroni adjustment for multiple test (irrelevant for Perli's and Lehmacher's
test).

Details

The cfa is used to sift large tables of nominal data. Usually it is used for dichotomous variables
but can be extended to three or more possible values. There should be at least three configuration
variables in cfg - otherwise a simple contingency table would do. All tests of significance are two-
sided: They test for both types or antitypes, i.e. if n is significantly larger or smaller than the
expected value. The usual caveats for testing contingency tables apply. If a configuration has a $n < 5$ an exact test should be used. As an alternative the least interesting configuration variable can be left out (if it is not essential) which will automatically increase the $n$ for the remaining configurations.

Value

Some of these elements will only be returned when the corresponding argument in the function call has been set. The relation is obvious due to corresponding names.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The cfa output table</td>
</tr>
<tr>
<td>table&quot;label&quot;</td>
<td>Label for the given configuration</td>
</tr>
<tr>
<td>table&quot;n&quot;</td>
<td>Observed $n$ for this configuration</td>
</tr>
<tr>
<td>table&quot;expected&quot;</td>
<td>Expected $n$ for this configuration</td>
</tr>
<tr>
<td>table&quot;Q&quot;</td>
<td>Coefficient of pronouncedness (varies between 0 and 1)</td>
</tr>
<tr>
<td>table&quot;chisq&quot;</td>
<td>Chi squared for the given configuration</td>
</tr>
<tr>
<td>table&quot;p.chisq&quot;</td>
<td>$p$ for the chi squared test</td>
</tr>
<tr>
<td>table&quot;sig.chisq&quot;</td>
<td>Is it significant (will Bonferroni-adjust if argument bonferroni is set)</td>
</tr>
<tr>
<td>table&quot;z&quot;</td>
<td>$z$-approximation for chi squared</td>
</tr>
<tr>
<td>table&quot;p.z&quot;</td>
<td>$p$ of $z$-test</td>
</tr>
<tr>
<td>table&quot;sig.z&quot;</td>
<td>Is it significant (will Bonferroni-adjust if argument bonferroni is set)?</td>
</tr>
<tr>
<td>table&quot;x.perli&quot;</td>
<td>Statistic for Perli’s test</td>
</tr>
<tr>
<td>table&quot;sig.perli&quot;</td>
<td>Is it significant (this is designed as a multiple test)?</td>
</tr>
<tr>
<td>table&quot;zl&quot;</td>
<td>$z$ for Lehmacher’s test</td>
</tr>
<tr>
<td>table&quot;sig.zl&quot;</td>
<td>Is it significant (this is designed as a multiple test)?</td>
</tr>
<tr>
<td>table&quot;zl.corr&quot;</td>
<td>$z$ for Lehmacher’s test (with continuity correction)</td>
</tr>
<tr>
<td>table&quot;sig.zl.corr&quot;</td>
<td>Is it significant (this is designed as a multiple test)?</td>
</tr>
<tr>
<td>table&quot;p.exact.bin&quot;</td>
<td>$p$ for exact binomial test</td>
</tr>
<tr>
<td>summary.stats</td>
<td>Summary stats for entire table</td>
</tr>
<tr>
<td>summary.stats&quot;totalchisq&quot;</td>
<td>Total chi squared</td>
</tr>
<tr>
<td>summary.stats&quot;df&quot;</td>
<td>Degrees of freedom</td>
</tr>
<tr>
<td>summary.stats&quot;p&quot;</td>
<td>$p$ for the chi squared test</td>
</tr>
<tr>
<td>summary.stats&quot;sum of counts&quot;</td>
<td>Sum of all counts</td>
</tr>
<tr>
<td>levels</td>
<td>Levels for each configuration. Should all be 2 for the bivariate case</td>
</tr>
</tbody>
</table>
WARNING

Note than spurious "significant" configurations are likely to appear in very large tables. The results should therefore be replicated before they are accepted as real. boot.cfa can be helpful to check the results.

Note

There are no hard-coded limits in the program so even large tables can be processed. The output table can be very wide if the levels of factors variables are long strings so 'options(width=..)' may need to be adjusted.

The object returned has the class scfa if a one-sample CFA was performed or the class mcfa if a repeated-measures CFA was performed. cfa() decides which one is appropriate by looking at cnts: If it is a vector, it will do a simple CFA. If it is a dataframe or matrix with 2 or more columns, a repeated-measures CFA is done.

Author(s)

Stefan Funke <s.funke@t-online.de>

References


See Also

scfa, mcfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B")[,rbinom(250,1,0.3)+1],c("C","D")[,rbinom(250,1,0.1)+1],
c("E","F")[,rbinom(250,1,0.3)+1],c("G","H")[,rbinom(250,1,0.1)+1])
counts<-trunc(runif(250)*10)
cfa(configs, counts)
**fcfa**

**Stepwise CFA approaches**

**Description**

These CFA methods detect and eliminate stepwise types/antitypes cells by specifying an appropriate contrast in the design matrix. The procedures stop when model fit is achieved. Functional CFA (fcfa) uses a residual criterion, Kieser-Victor CFA (kvCFA) a LR-criterion.

**Usage**

fcfa(m, X, tabdim, alpha = 0.05)
kvCFA(m, X, tabdim, alpha = 0.05)

**Arguments**

- m Vector of observed frequencies.
- X Design Matrix of the base model.
- tabdim Vector of table dimensions.
- alpha Significance level.

**Value**

- restable Fit results for each step
- design.mat Final design matrix
- struct.mat Structural part of the design matrix for each step
- typevec Type or antitype for each step
- resstep Design matrix, expected frequency vector, and fit results for each step

**Author(s)**

Patrick Mair, Alexander von Eye

**References**


Examples

# Functional CFA for a internet terminal usage data set by Wurzer
# (An application of configural frequency analysis: Evaluation of the
# usage of internet terminals, 2005, p.82)
dd <- data.frame(a1=gl(3,4),b1=gl(2,2,12),c1=gl(2,1,12))
X <- model.matrix(~a1+b1+c1,dd,contrasts=list(a1="contr.sum",b1="contr.sum",
c1="contr.sum"))
ofreq <- c(121,13,44,37,158,69,100,79,24,0,26,3)
tabdim <- c(3,2,2)
res1 <- fCFA(ofreq, X, tabdim=tabdim)
res1
summary(res1)

# Kieser-Vector CFA for Children's temperament data from
# von Eye (Configural Frequency Analysis, 2002, p. 192)
dd <- data.frame(a1=gl(3,9),b1=gl(3,3,27),c1=gl(3,1,27))
X <- model.matrix(~a1+b1+c1,dd,contrasts=list(a1="contr.sum",b1="contr.sum",c1="contr.sum"))
ofreq <- c(3,2,4,23,23,6,39,33,9,11,29,13,19,36,19,21,26,18,13,30,41,12,14,23,8,6,7)
tabdim <- c(3,3,3)
res2 <- kvCFA(ofreq, X, tabdim=tabdim)
res2
summary(res2)

hcfa

Hierachical analysis of configuration frequencies

Description

Recursively eliminates one variable in the configuration to generate all possible sub-tables and
performs a global chi-squared-test on them

Usage

hcfa(configs, cnts)

Arguments

configs Contains the configurations. This can be a dataframe or a matrix. The dataframe
can contain numbers, characters, factors or booleans. The matrix can consist of
numbers, characters or booleans (factors are implicitly re-converted to numerical
levels). There must be >=3 columns.

cnts Contains the counts for the configuration. If it is set to NA, a count of one is
assumed for every row. This allows untabulated data to be processed. cnts can
be a vector or a matrix/dataframe with >=2 columns.
Details

The hierarchical CFA assists in the selection of configuration variables by showing which variables contribute the most to the variability. If eliminating a variable does not markedly decrease the global chi squared the variable is likely to be redundant, provided there are no extraneous reasons for retaining it.

The output is in decreasing order of chi squared so the most useful combinations of variables come first.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chisq</td>
<td>Global chi squared</td>
</tr>
<tr>
<td>df</td>
<td>Degrees of freedom for this subtable</td>
</tr>
<tr>
<td>order</td>
<td>Order (number of configuration variables)</td>
</tr>
</tbody>
</table>

Note

The p for the test of significance ist provided by the print method

Author(s)

Stefan Funke <s.funke@t-online.de>

References


See Also

cfa, scfa, mcfa

Examples

```R
# library(cfa) if not yet loaded
# Some random configurations:
configs <- cbind(c("A","B")%*%rbinom(250,1,0.3)+1,  
c("C","D")%*%rbinom(250,1,0.1)+1,  
c("E","F")%*%rbinom(250,1,0.3)+1,c("G","H")%*%rbinom(250,1,0.1)+1)
counts <- trunc(runif(250)*10)
hcfa(configs, counts)
```
Two or more-sample CFA

Description

Performs an analysis of configuration frequencies for two or more sets of counts. This function is not designed to be called directly by the user but will only be used internally by cfa(). Both the simple and the multiple cfa are handled by cfa().

Usage

mcfa(cfg, cnts, sorton="chisq", sort.descending=TRUE, format.labels=TRUE)

Arguments

cfg
Contains the configurations. This can be a dataframe or a matrix. The dataframe can contain numbers, characters, factors or booleans. The matrix can consist of numbers, characters or booleans (factors are implicitly re-converted to numerical levels). There must be >=3 columns.

cnts
Contains the counts for the configuration. cnts is a matrix or dataframe with 2 or more columns.

sorton
Determines the sorting order of the output. Can be set to chisq, n, or label.

sort.descending
Sort in descending order

format.labels
Format the labels of the configuration. This makes to output wider but it will increase the readability.

Details

This function is the "engine" cfa() will use. It does the aggregation, summing up, and will calculate chi squared. All tests of significance are left to cfa().

Value

The function returns the following list:

labels   Configuration label
sums     Sums for each configuration and each variable in the configuration
counts   Matrix of observed n of the given configuration
expected Matrix of expected n for the given configuration
chisq    Chi squared for each configuration

Note

There are no hard-coded limits in the program so even large tables can be processed.
Author(s)

Stefan Funke <s.funke@t-online.de>

References


See Also

cfa, scfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B"),rbinom(250,1,0.3)+1),c("C","D"),rbinom(250,1,0.1)+1),
c("E","F"),rbinom(250,1,0.3)+1),c("G","H"),rbinom(250,1,0.1)+1))

counts1<-trunc(runif(250)*10)
counts2<-trunc(runif(250)*10)
cfa(configs,cbind(counts1,counts2))
# cfa rather than mcfa!

plot.bcfa  

Plotting method for a bcfa object

Description

Plots an object of the class bcfa

Usage

## S3 method for class 'bcfa'

plot(x,...)

Arguments

x  
An object of the class bcfa which is returned by the function boot.cfa()

...  
Any arguments to be given to plot
Details

Plots the number of cases considered significant vs. the number of cases considered to be a type (n > expected).

This is in some way like other plots of quality versus quantity.
Configurations can be identified by left-clicking on them until the right mouse button is pressed. The labels of the configurations selected will be displayed in the text window.

Value

Returns a vector of the configurations selected with their name set to the labels

Note

This function is usually invoked plotting an object returned by bcfa

Author(s)

Stefan Funke <s.funke@t-online.de>

References

None - plots have been rarely used with the CFA

See Also

bcfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B") [rbinom(250,1,0.3)+1], c("C","D") [rbinom(250,1,0.1)+1],
               c("E","F") [rbinom(250,1,0.3)+1], c("G","H") [rbinom(250,1,0.1)+1])
counts<-runc(runiform(250)*10)
plot(bcfa(configs, counts, runs=25))

Description

Plots an object of the class hcfa

Usage

## S3 method for class 'hcfa'
plot(x,...)
Arguments

- **x**: An object of the class `hcfa`
- **...**: Any arguments to be used by `plot`

Details

A dotchart is generated which plots chi squared vs. the order of the configuration (i.e. the number of configuration variables it contains).

Value

Returns NULL.

Note

This function is usually invoked plotting an object returned by `hcfa`

Author(s)

Stefan Funke <s.funke@t-online.de>

References

None - plots have been rarely used with the CFA

See Also

cfa, hcfa

Examples

```r
# configs <- cbind(c("A", "B"), rbinom(250, 1, 0.3) + 1),
#                  c("C", "D")%*%(rbinom(250, 1, 0.1) + 1),
#                  c("E", "F")%*%(rbinom(250, 1, 0.3) + 1),
#                  c("G", "H")%*%(rbinom(250, 1, 0.1) + 1))
# counts <- trunc(runif(250) * 10)
# plot(hcfa(configs, counts))
```
Arguments

x  An object of the class mcfa which is returned by the function cfa() (rather than mcfa()) which performs a repeated measures CFA (two or more columns of counts)

... Any arguments to be used by plot

Details

Plots chi squared vs. the sum of all counts for this configuration which indicates pronouncedness of the configuration vs. practical importance. Configurations can be identified by left-clicking on them until the right mouse button is pressed. The labels of the configurations selected will be displayed in the text window.

Value

Returns a list of the labels of the configurations selected.

Note

This function is usually invoked plotting an object returned by cfa

Author(s)

Stefan Funke <s.funke@t-online.de>

References

None - plots have been rarely used with the CFA

See Also

cfa, mcfa

Examples

# Some random configurations:
cfgs<-cbind(c("A","B")rbinom(250,1,0.3)+1],c("C","D")rbinom(250,1,0.1)+1],
c("E","F")rbinom(250,1,0.3)+1],c("G","H")rbinom(250,1,0.1)+1])
counts1<-trunc(runif(250)*10)
counts2<-trunc(runif(250)*10)
plot(cfa(cfgs,cbind(counts1,counds2)))
plot.scfa

Plotting method for a scfa object

Description

Plots an object of the class scfa

Usage

```r
## S3 method for class 'scfa'
plot(x, ...)
```

Arguments

- `x`: An object of the class scfa which is returned by the function cfa() (rather than scfa()) which performs a simple CFA (one column of counts)
- `...`: Any arguments to be used by plot

Details

Plots chi squared vs. n which indicates pronouncedness of the configuration vs. practical importance. Configurations can be identified by left-clicking on them until the right mouse button is pressed. The labels of the configurations selected will be displayed in the text window.

Value

Returns a list of the labels of the configurations selected.

Note

This function is usually invoked plotting an object returned by cfa

Author(s)

Stefan Funke <s.funke@t-online.de>

References

None - plots have been rarely used with the CFA

See Also

- cfa, scfa
Examples

# library(cfa) if not yet loaded
# Some random configurations:
cfg <- cbind(c("A", "B") [rbinom(250, 1, 0.3) + 1], c("C", "D") [rbinom(250, 1, 0.1) + 1],
              c("E", "F") [rbinom(250, 1, 0.3) + 1], c("G", "H") [rbinom(250, 1, 0.1) + 1])

cnts <- trunc(runif(250) * 10)
plot(cfa(cfg, cnts))

print.bcfa

Print an object of the class hcfa

Description

Printing method for an object returned by boot.cfa()

Usage

## S3 method for class 'bcfa'
print(x, ...)

Arguments

x 
An object of the class bcfa

... 
Additional arguments given to print

Details

This function is usually called implicitly.

Value

Returns NULL

Author(s)

Stefan Funke <s.funke@t-online.de>

References


**See Also**

`bcfa`

**Examples**

```r
# library(cfa) if not yet loaded
# Some random configurations:
configs <- cbind(c("A","B")[rbinom(250, 1, 0.3)+1], c("C","D")[rbinom(250, 1, 0.1)+1],
                 c("E","F")[rbinom(250, 1, 0.3)+1], c("G","H")[rbinom(250, 1, 0.1)+1])

counts <- trunc(runif(250))*10

result <- bcfa(configs, counts, runs=25)

print(result)
```

**print.hcfa**

*Print an object of the class hcfa*

**Description**

Printing method for an object returned by `hier.cfa()`

**Usage**

```r
## S3 method for class 'hcfa'

print(x, ...)
```

**Arguments**

- `x`: An object of the class `hcfa`
- `...`: Additional arguments given to `print`

**Details**

This function is usually called implicitly.

**Value**

Returns NULL.

**Author(s)**

Stefan Funke <s.funke@t-online.de>
References


See Also

hcfa

Examples

```r
cfg <- cbind(c("A","B"), rbinom(250, 1, 0.3)+1, c("C","D"), rbinom(250, 1, 0.1)+1,
             c("E","F"), rbinom(250, 1, 0.3)+1, c("G","H"), rbinom(250, 1, 0.1)+1)
cnts <- trunc(runif(250)*10)
result <- hcfa(cfg, cnts)
print(result)
```

Description

Printing method for one of two possible objects returned by cfa()

Usage

```r
## S3 method for class 'mcfa'
print(x,...)
```

Arguments

- **x**: An object of the class mcfa
- **...**: Additional arguments given to print

Details

This function is usually called implicitly.

Value

Returns NULL
Note

Note that `cfa()` will return an object with the class `scfa` if there is only one row of counts. If there are two or more of them, an object with the class `mcfa` is returned. In contrast `scfa()` and `mcfa()` return a list which has no class of its own.

Author(s)

Stefan Funke <s.funke@t-online.de>

References


See Also

cfa, mcfa

Examples

```r
# library(cfa) if not yet loaded
# Some random configurations:
configs <- cbind(c("A","B") [rbinom(250,1,0.3)+1], c("C","D") [rbinom(250,1,0.1)+1],
                   c("E","F") [rbinom(250,1,0.3)+1], c("G","H") [rbinom(250,1,0.1)+1])

counts1 <- trunc(runif(250)*10)
counts2 <- trunc(runif(250)*10)
result <- cfa(configs, cbind(counts1, counts2))
print(result)
```

---

**print.scfa**

*Print an object of the class scfa*

Description

Printing method for one of two possible objects returned by `cfa()`

Usage

```r
# S3 method for class 'scfa'
print(x,...)
```
Arguments

x An object of the class scfa

... Additional arguments given to print

Details

This function is usually called implicitly.

Value

Returns NULL

Note

Note that cfa() will return an object with the class scfa if there is only one row of counts. If there are two or more of them, an object with the class mcfa is returned. In contrast scfa() and mcfa() return a list which has no class of it’s own.

Author(s)

Stefan Funke <s.funke@t-online.de>

References


See Also

cfa, scfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B")|rbinom(250,1,0.3)+1],c("C","D")|rbinom(250,1,0.1)+1],
c("E","F")|rbinom(250,1,0.3)+1],c("G","H")|rbinom(250,1,0.1)+1])
counts<-trunc(runif(250)*10)
result<-cfa(configs,counts)
print(result)
Test according to Lindner

Description

Performs a test of significance according to Lindner

Usage

PXisM(m, n, Nt, k)

Arguments

m  
Observed frequency of the observation tested

n  
Marginal sums of the parameters realized in the configuration to be tested (vector)

Nt  
Sample size of configurations

k  
Number of parameters

Value

returns p for the test according to Lindner

Note

The test according to Lindner requires the packages parallel. All other parts of cfa do not.

Author(s)

J. Harloff <oachimharloff@joachimharloff.de>

References


Harloff, Joachim, An efficient algorithm for Lindners test (configural frequency analysis), Qual Quant DOI 10.1007/s11135-011-9499-9

See Also

cfa
Examples

# Does not work with windows since there is no parallel for it
if (require(parallel)) {
  lk<-4 # number of parameters
  ln<-c(59,57,59,58) # marginal sums of the parameters realized in the configuration to be tested
  lnt<-116 # sample size of configurations
  lm0<-16 # observed frequency of the configuration tested

  # New algorithm
  starttime=proc.time()
  pHXsmallerequalM0<-sum(unlist(mclapply(0:lm0, PXisM, ln, lnt, lk)))
  pHXequalM0<-PXisM(lm0, ln, lnt, lk)
  pHlargerequalM0<-sum(unlist(mclapply(1m0: min(ln), PXisM, ln, lnt, lk)))
  stoptime<proc.time()
  list(PHXsmallerequalM0=PHXsmallerequalM0, PHXequalM0=PHXequalM0, PHlargerequalM0=PHlargerequalM0, timed.required=stoptime-starttime)

  # End of the new algorithm
}

PXisMclassic  Test according to Lindner

Description

Performs a test of significance according to Lindner - old algorithm

Usage

PXisMclassic(m,n,Nt,k)

Arguments

- **m**: Observed frequency of the observation tested
- **n**: Marginal sums of the parameters realized in the configuration to be tested (vector)
- **Nt**: Sample size of configurations
- **k**: Number of parameters

Value

returns p for the test according to Lindner

Note

The test according to Lindner requires the packages parallel. All other parts of cfa do not.
**Author(s)**

J. Harloff <oachimharloff@joachimharloff.de>

**References**


Harloff, Joachim, An efficient algorithm for Lindners test (configural frequency analysis), Qual Quant DOI 10.1007/s11135-011-9499-9

**See Also**

cfa

**Examples**

```r
# Does not work with windows since there is no parallel for it
if (require (parallel)) {

lk<-4 # number of parameters
ln<-c(59,57,59,58) # marginal sums of the parameters realized in the configuration to be tested
lNt<-116 # sample size of configurations
lm0<-16 # observed frequency of the configuration tested

# Old algorithm
starttime=proc.time()
phxsmallerequalm0<-sum(unlist(mclapply(0:lm0,PXIsMcclassic,ln,lNt,lk)))
phxequalm0<-PXIsMcclassic(lm0,ln,lNt,lk)
phlargerequalm0<-sum(unlist(mclapply(lm0: min(ln),PXIsMcclassic,ln,lNt,lk)))
stoptime<-proc.time()
list(phxsmallerequalm0=phxsmallerequalm0,phxequalm0=phxequalm0,phlargerequalm0=phlargerequalm0, timed.required=stop-time-starttime)
# End of the old algorithm
}
```

---

**Description**

Performs a configuration frequency analysis if only one set of counts exists. *This function is not designed to be called directly by the user but will only be used internally by cfa().* Both the simple and the multiple cfa are handled by cfa()

**Usage**

```r
scfa(cfg, cnt=NA, sorton="chisq", sort.descending=TRUE, format.labels=TRUE)
```
Arguments

*cfg*  
Contains the configurations. This can be a dataframe or a matrix. The dataframe can contain numbers, characters, factors or booleans. The matrix can consist of numbers, characters or booleans (factors are implicitly re-converted to numerical levels). There must be >=3 columns.

*cnt*  
Contains the counts for the configuration. If it is set to NA, a count of one is assumed for every row. This allows untabulated data to be processed. *cnts* is a vector.

*sorton*  
Determines the sorting order of the output. Can be set to *chisq*, *n*, or *label*.

*sort.descending*  
Sort in descending order

*format.labels*  
Format the labels of the configuration. This makes to output wider but it will increase the readability.

Details

This function is the "engine" *cfa()* will use. It does the aggregation, summing up, and will calculate chi squared. All tests of significance are left to *cfa()*

Value

The function returns the following list:

- **labels**: Configuration label
- **n.levels**: Number of levels for each configuration
- **sums**: Sums for each configuration and each variable in the configuration
- **counts**: Observed n of the given configuration
- **expected**: Expected n for the given configuration
- **chisq**: Chi squared for each configuration

Note

There are no hard-coded limits in the program so even large tables can be processed.

Author(s)

Stefan Funke <s.funke@t-online.de>

References


scfa

See Also
cfa, mcfa

Examples

# library(cfa) if not yet loaded
# Some random configurations:
configs<-cbind(c("A","B")%*%rbinom(250,1,0.3)+1,c("C","D")%*%rbinom(250,1,0.1)+1,
c("E","F")%*%rbinom(250,1,0.3)+1,c("G","H")%*%rbinom(250,1,0.1)+1)

counts<-trunc(runif(250)*10)
cfa(configs,counts)
# cfa rather than scfa!
Index

*Topic htest
  bcfa, 2
  cfa, 3
  hcfa, 8
  mcfa, 10
  plot.bcfa, 11
  plot.hcfa, 12
  plot.mcfa, 13
  plot.scfa, 15
  print.bcfa, 16
  print.hcfa, 17
  print.mcfa, 18
  print.scfa, 19
  PXisM, 21
  PXisMclassic, 22
  scfa, 23

*Topic models
  fcfa, 7

*Topic multivariate
  bcfa, 2
  cfa, 3
  hcfa, 8
  mcfa, 10
  plot.bcfa, 11
  plot.hcfa, 12
  plot.mcfa, 13
  plot.scfa, 15
  print.bcfa, 16
  print.hcfa, 17
  print.mcfa, 18
  print.scfa, 19
  PXisM, 21
  PXisMclassic, 22
  scfa, 23

bcfa, 2, 12, 17

cfa, 3, 9, 11, 13–15, 19–21, 23, 25

fcFA, 7

hcfa, 8, 13, 18

kvCFA (fcFA), 7

mcfa, 6, 9, 10, 14, 19, 25

plot.bcfa, 11
plot.hcfa, 12
plot.mcfa, 13
plot.scfa, 15
print.bcfa, 16
print.fcFA (fcFA), 7
print.hcfa, 17
print.kvCFA (fcFA), 7
print.mcfa, 18
print.scfa, 19
PXisM, 21
PXisMclassic, 22

scfa, 3, 6, 9, 11, 15, 20, 23
summary.fcFA (fcFA), 7
summary.kvCFA (fcFA), 7