Package: cfa (via r-universe)

September 12, 2024

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 0.10-1

Date 2024-03-14

Depends R (>= 3.0.1)

Suggests parallel

License GPL (>= 2)

NeedsCompilation yes

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

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

Date/Publication 2024-03-15 08:36:26 UTC

Repository https://pmair78.r-universe.dev

RemoteUrl https://github.com/cran/cfa

RemoteRef HEAD

RemoteSha 5440f75129e866c5600637cc4efe2d482f3ad3bf

Contents

`bcfa` *Bootstrap-CFA*

Description

The bootstrap-CFA tries to replicate the pattern of significant configurations by re-sampling.

Usage

```
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 implicitely 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` must be a vector.
`runs`	Number of samples to be drawn.
`sig.item`	Indicator of significance in the result table (sig.z,sig.chisq,sig.perli,sig.zl, sig.zl.corr). Do not forget to set the proper parameters for the CFA if sig.perli,sig.zl or 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 antiypes
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

bcfa() 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

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse 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")[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)
bcfa(configs,counts,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 can contain numbers, characters, factors, or booleans. The matrix can consist of numbers, characters, or booleans (factors are implicitely 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 to 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 configuration.
`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 contigency 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 contigency 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`	The cfa output table
`table["label"]`	Label for the given configuration
`table["n"]`	Observed n for this configuration
`table["expected"]`	Expected n for this configuration
`table["Q"]`	Coefficient of pronouncedness (varies between 0 and 1)
`table["chisq"]`	Chi squared for the given configuration
`table["p.chisq"]`	p for the chi squared test
`table["sig.chisq"]`	Is it significant (will Bonferroni-adjust if argument `bonferroni` is set)
`table["z"]`	z-approximation for chi squared
`table["p.z"]`	p of z-test
`table["sig.z"]`	Is it significant (will Bonferroni-adjust if argument `bonferroni` is set)?
`table["x.perli"]`	Statistic for Perli's test
`table["sig.perli"]`	Is it significant (this is designed as a multiple test)?
`table["zl"]`	z for Lehmacher's test
`table["sig.zl"]`	Is it significant (this is designed as a multiple test)?
`table["zl.corr"]`	z for Lehmacher's test (with continuity correction)
`table["sig.zl.corr"]`	Is it significant (this is designed as a multiple test)?
`table["p.exact.bin"]`	p for exact binomial test
`summary.stats`	Summary stats for entire table
`summary.stats["totalchisq"]`	Total chi squared
`summary.stats["df"]`	Degrees of freedom
`summary.stats["p"]`	p for the chi squared test
`summary.stats["sum of counts"]`	Sum of all counts
`levels`	Levels for each configuration. Should all be 2 for the bivariate case

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 ist done.

Author(s)

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

References

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin. Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin. Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

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.i,  X, tabdim, alpha = 0.05)
kvCFA(m.i, X, tabdim, alpha = 0.05)
```

Arguments

m.i	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
struc.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

von Eye, A., and Mair, P. (2008). A functional approach to configural frequency analysis. Austrian Journal of Statistics, 37, 161-173.

Kieser, M., and Victor, N. (1999). Configural frequency analysis (CFA) revisited: A new look at an old approach. Biometrical Journal, 41, 967-983.

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 implicitely 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

`chisq`	Global chi squared
`df`	Degrees of freedom for this subtable
`order`	Order (number of configuration variables)

Note

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

Author(s)

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

References

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

See Also

cfa, 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)
hcfa(configs,counts)
```

---

mcfa *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 an 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 implicitely 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

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

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

`plot.hcfa` *Plotting method for a hcfa object*

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

```
#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))
```

`plot.mcfa` *Plotting method for a mcfa object*

Description

Plots an object of the class `mcfa`

Usage

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

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:
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)

plot(cfa(configs,cbind(counts1,counts2)))
```

---

plot.scfa *Plotting method for a scfa object*

Description

Plots an object of the class scfa

Usage

```
## 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:
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(cfa(configs,counts))
```

---

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 implicitely.

Value

Returns NULL

Author(s)

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

References

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

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<-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

```
## 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 implicitely.

Value

Returns NULL.

Author(s)

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

References

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

See Also

hcfa

Examples

```
#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<-hcfa(configs,counts)
#print(result)
```

---

print.mcfa *Print an object of the class mcfa*

Description

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

Usage

```
## 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 implicitely.

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

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

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])
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

```
## 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 implicitely.

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

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and anti-types in cross-classification. Cambride 1990

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)
```

PXisM *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 Linder

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

Lindner, K.: Eine exakte Auswertungsmethode zur Konfigurationsfrequenzanalyse [An exact procedure for the configural frequency analysis]. Psycholog Beitraege 26, 393?415 (1984)

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(lm0: 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 Linder

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

Lindner, K.: Eine exakte Auswertungsmethode zur Konfigurationsfrequenzanalyse [An exact procedure for the configural frequency analysis]. Psycholog Beitraege 26, 393?415 (1984)

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


# Old algorithm
starttime=proc.time()
pHXsmallerequalM0<-sum(unlist(mclapply(0:lm0,PXisMclassic,ln,lNt,lk)))
pHXequalM0<-PXisMclassic(lm0,ln,lNt,lk)
pHlargerequalM0<-sum(unlist(mclapply(lm0: min(ln),PXisMclassic,ln,lNt,lk)))
stoptime<-proc.time()
list(pHXsmallerequalM0=pHXsmallerequalM0,pHXequalM0=pHXequalM0,pHlargerequalM0=pHlargerequalM0,
timed.required=stoptime-starttime)
# End of the old algorithm
}
```

---

scfa *One sample CFA*

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 by* `cfa()`. Both the simple an the multiple cfa are handled by `cfa()`

Usage

```
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 implicitely 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

Krauth J., Lienert G. A. (1973, Reprint 1995) Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin, Beltz Psychologie Verlagsunion

Lautsch, E., von Weber S. (1995) Methoden und Anwendungen der Konfigurationsfrequenzanalyse Psychologie und Medizin, Beltz Psychologie Verlagsunion

Eye, A. von (1990) Introduction to configural frequency analysis. The search for types and antitypes in cross-classification. Cambride 1990

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