Package: confreq (via r-universe)

November 23, 2026

Type Package

Title Configural Frequencies Analysis Using Log-Linear Modeling

Version 1.7.0-7

License GPL-3

LazyData true

Encoding UTF-8

Depends R (>= 3.5.0), stats, gmp, methods, grid, vcd

Date 2026-11-21

Maintainer Joerg-Henrik Heine <jhheine@googlemail.com>

Description Offers several functions for Configural Frequencies Analysis (CFA), which is a useful statistical tool for the analysis of multiway contingency tables. CFA was introduced by G. A. Lienert as 'Konfigurations Frequenz Analyse - KFA'. Lienert, G. A. (1971). Die Konfigurationsfrequenzanalyse: I. Ein neuer Weg zu Typen und Syndromen. Zeitschrift für Klinische Psychologie und Psychotherapie, 19(2), 99–115.

NeedsCompilation no

Config/roxygen2/version 8.1.1

Author Joerg-Henrik Heine [aut, cre, com], Rainer W. Alexandrowicz [aut] (Developed functions for the Stirling test), Luciana Diaconescu [aut] (Developed functions for the determination coefficient Q), Mark Stemmler [rev, dtc] (Comprehensive package testing)

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

Date/Publication 2026-11-23 08:10:36 UTC

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

RemoteRef HEAD

RemoteSha 746ff95eaa741ddcfc47a9608c2576f97cde5449

Contents

[.Pfreq . . . 2
anova.CFA . . . 3
binomial_test_cfa . . . 4
CFA . . . 5
chi_local_test_cfa . . . 8
coef.CFA . . . 9
confreq_PACKAGE . . . 9
dat2cov . . . 11
dat2fre . . . 13
design_cfg_cfa . . . 15
df_des_cfa . . . 16
expected_cfa . . . 17
expected_margin_cfa . . . 18
fre2dat . . . 19
fre2tab . . . 20
ftab . . . 22
lazar . . . 22
Lienert1978 . . . 23
LienertLSD . . . 24
lr . . . 25
newborns . . . 26
plot.CFA . . . 27
plot.S2CFA . . . 28
pos_cfg_cfa . . . 30
print.Pfreq . . . 31
Q_determination_bound_cfa . . . 32
Q_determination_cfa . . . 33
S2CFA . . . 34
stirling_cfa . . . 36
suicide . . . 37
summary.CFA . . . 38
summary.S2CFA . . . 40
z_tests_cfa . . . 42

Index **44**

`[.Pfreq` *Subset Pattern-Frequency Tables without Misaligning Weights*

Description

Keep cell-level weight attributes aligned when selecting or reordering rows of a `Pfreq` object.

Usage

```
## S3 method for class 'Pfreq'
x[i, j, drop]
```

Arguments

x	A Pfreq object.
i, j	Row and column selections, as for [.data.frame.
drop	Passed to the data-frame method if supplied. Otherwise its ordinary default applies.

Details

The row attributes wgtfreq, WGT, wgtfreq2 and wgtpositive follow the selected rows, including named rows. A row permutation retains the original individual-weight provenance attribute wgtdat; selecting fewer configurations removes it because the remaining individual weights cannot be reconstructed from cell totals.

The Pfreq class is retained only for nonempty selections of unique, existing rows that keep all configuration columns exactly once and the original frequency column last. Other data-frame results lose the Pfreq class and weight attributes. No automatic marginalization is performed when configuration columns are removed. Vector results follow the ordinary data-frame dropping rules. This method does not protect arbitrary editing by [<-, $<-, [[<- or manual attribute replacement.

An incomplete selection of unique configurations remains a frequency table, but is not necessarily suitable for CFA, which needs a complete grid. The corresponding analysis input checks should be respected.

Value

The selected object, with aligned weight attributes if it remains a Pfreq object; otherwise an ordinary data frame or dropped vector.

See Also

dat2fre, fre2tab

Examples

```
d <- data.frame(A = factor(c("a", "b", "a")),
                B = factor(c("x", "x", "y")))
pf <- dat2fre(d, wgt = c(1, 2, 3))
sorted <- pf[rev(seq_len(nrow(pf))), ]
attr(sorted, "wgtfreq")
```

---

anova.CFA *S3 analysis of variance (or deviance) tables for CFA objects*

Description

S3 anova method for object of class"CFA".

Usage

```
## S3 method for class 'CFA'
anova(object, ..., test = "Chisq")
```

Arguments

object	object of class "CFA".
...	additional objects of the same type.
test	Character string specifying the statistical test used for model comparison. See anova.glm for available options. Defaults to "Chisq".

Details

Model descriptions use the CFA specification stored in used.formula, including functional patterns and covariates. The model fits and comparison statistics are not changed. With a single object, the complete design matrix is treated as one GLM term; the table does not decompose the individual CFA main effects or interactions.

Value

This (generic) function returns an object of class anova.

---

binomial_test_cfa *Binomial Test*

Description

Calculates the (exact) binomial test based on obseved, expected frequencies an the total number of observations.

Usage

```
binomial_test_cfa(observed, expected, ntotal = sum(observed))
```

Arguments

observed	a vector giving the observed frequencies.
expected	a vector giving the expected frequencies.
ntotal	optional a numeric giving the total number of observations. By default ntotal is calculated as ntotal=sum(observed).

Details

No details

Value

a numeric giving the p-value.

References

No references in the moment

Examples

```
#######################################
# first calculate expected counts for LienertLSD data example.
designmatrix<-design_cfg_cfa(kat=c(2,2,2)) # generate an designmatrix (only main effects)
data(LienertLSD) # load example data
observed<-LienertLSD[,4] # extract observed counts
expected<-expected_cfa(des=designmatrix, observed=observed) # calculation of expected counts
 binomial_test_cfa(observed,expected)
#######################################
```

CFA *Configural Frequencies Analysis Main Function*

Description

Calculates various coefficients for the Configural Frequencies Analysis (CFA) defining main- and (optional) interaction effects. The core principle is to use `glm` in package `stats` to calculate the expected counts considering a designmatrix, which is constructed based on an formula definition given in argument `form`.

Usage

```
CFA(
  patternfreq,
  alpha = 0.05,
  form = NULL,
  ccor = FALSE,
  family = poisson(),
  intercept = FALSE,
  method = "log",
  blank = NULL,
  cova = NULL,
  bintest = TRUE,
  ...
)
```

Arguments

patternfreq	an object of class "Pfreq", which is data in pattern frequencies representation - see function dat2fre.
alpha	a numeric giving the alpha level for testing (default set to alpha=.05)
form	either a character expression which can be coerced into a model formula with the function as.formula in the package stats. If this argument is left empty (at default form=NULL) the (internal) function design_cfg_cfa() will return a designmatrix coding only main effects and no interactions – for a designmatrix referring to three variables (V1, V2, V3) for example, leaving the argument form empty will be equivalent to assigning the character "~ V1 + V2 + V3" to the argument (form="~ V1 + V2 + V3"). A special case is to define a null-model or rather a cfa model of order zero. In such a model no (main) effects are considered. This can be achieved bei passing the character expression "null" to the argument form – so: form = "null" – not to be confound with the default setting of this argument form=NULL. Another option is to define your own designmatrix and assign it to this argument (form) in this case the object assigned to form must be of class "matrix" and must logical match to the argument patternfreq, which is currently not checked! - but simply assumed.
ccor	either a logical (TRUE / FALSE) determining whether to apply a continuity correction or not for the Binomial Approximation to the z-Test. When set to ccor=TRUE continuity correction is applied for expected values 5 =< expected =< 10. For ccor=FALSE no continuity correction is applied. Another option is to set ccor=c(x,y) where x is the lower and y the upper bound for expected values where continuity correction is applied. So ccor=c(5,10) is equivalent to ccor=TRUE.
family	argument passed to glm with default set to poisson()
intercept	deprecated argument currently ignored – former was passed to glm.fit with default set to FALSE
method	character defining the estimation method for expected frequencies with default set to method="log" to estimate the expected frequencies using glm. An other option is to set this argument to method="margins" which will result in expected frequencies calculated based on the margins of the multidimensional contingency table. Only main effects models are possible in this case and thus the arguments form, family cova and intercept are ignored.
blank	can be used to indicate which pattern (configurations) are declared as structural cells (configurations) for functional CFA. Should be either (1) a character vector defining the pattern (with spaces between variable categories), which will be ignored for calculation of expected frequencies; or (2) a numeric vector defining the (row) positions of the pattern in an object of class "Pfreq" (see. argument patternfreq), which will be ignored for calculation of expected frequencies. At default (blank=NULL) all possible pattern, as listed in object of class "Pfreq", are included for calculation of expected frequencies.
cova	a matrix (possibly with one or more columns) holding the covariate (mean) values for each pattern (configurations) see function dat2cov.

bintest	a logical with default set to bintest=TRUE; if set to bintest=FALSE no calculations for the exact binomial test are performed, which can reduce processing time in some cases dramatically.
...	additional parameters passed through to other functions.

Details

This is the main function of the package. It internal calls several functions of the package confreq-package which are also available as single functions. For classification of the observed patterns into 'Types' and 'Antitypes' according to Lienert (1971), a S3 summary method for the resulting object of class "CFA" can be applied - see summary.CFA. An S3 plot method is useful for visualization of the contingency table and the 'Types' and 'Antitypes' – see plot.CFA. Since version 1.6.0-1 of confreq survey weights are supported when tabluating a data set with function dat2fre. In case that for the resulting tabulated data in the object of class c("data.frame","Pfreq") survey weights were used the function CFA will take into account those weigts for estimation of the expected counts – currently only when method="log".

Input must contain a complete, nonduplicated configuration grid in canonical category order. Invalid input is rejected before fitting, rather than silently pairing frequencies with the wrong design-matrix rows. After selecting or sorting rows, recreate canonical input and align any covariates or custom design matrix explicitly. Weighted input is detected from a valid wgtfreq attribute, not from an arbitrary comment. Weighted method="margins" is rejected because the historical margin calculation uses raw frequencies. The existing weighted log-linear path is retained; it is not a new design-based survey-variance implementation.

Value

an object of class CFA with results.

References

Lienert, G. A. (1971). Die Konfigurationsfrequenzanalyse: I. Ein neuer Weg zu Typen und Syndromen. *Zeitschrift für Klinische Psychologie und Psychotherapie, 19*(2), 99-115.

Glück, J., & Von Eye, A. (2000). Including covariates in Configural Frequency Analysis. *Psychologische Beitrage, 42*, 405–417.

Victor, N. (1989). An Alternativ Approach to Configural Frequency Analysis. *Methodika, 3*, 61–73.

Stemmler, M. (2020). *Person-Centered Methods*. Cham: Springer International Publishing.

Examples

```
#######################################
data(LienertLSD)
LienertLSD
res1 <- CFA(LienertLSD)
summary(res1)
plot(res1)
coef(res1)
## testing independence conditional on structural pattern number 8 (functional approach)
res1_f8 <- CFA(LienertLSD,blank = 8)
summary(res1_f8)
```

```
plot(res1_f8)
coef(res1_f8)
## testing with (full) interactions
res2 <- CFA(LienertLSD,form="~ C + T + A + C:T + C:A + T:A + C:T:A")
summary(res2)
plot(res2)
coef(res2)
#' ## testing the null model
res3 <- CFA(LienertLSD,form="null")
summary(res3)
plot(res3)
coef(res3)
########################
data(suicide)
suicide
# suicide data is in non tabulated data representation - so it must be tabulated !
res4 <- CFA(dat2fre(suicide))
summary(res4)
plot(res4)
coef(res4)
```

`chi_local_test_cfa` *Local Chi-Square Test*

Description

Calculates the local chi-square test based on obseved and expected frequencies.

Usage

```
chi_local_test_cfa(observed, expected)
```

Arguments

`observed`	a vector giving the observed frequencies.
`expected`	a vector giving the expected frequencies.

Details

No details in the moment.

Value

a list with chi-square statistic and corresponding degrees of freedom an p-value.

References

No references in the moment

Examples

```
#####################################
# first calculate expected counts for LienertLSD data example.
designmatrix<-design_cfg_cfa(kat=c(2,2,2)) # generate an designmatrix (only main effects)
data(LienertLSD) # load example data
observed<-LienertLSD[,4] # extract observed counts
expected<-expected_cfa(des=designmatrix, observed=observed) # calculation of expected counts
 chi_local_test_cfa(observed,expected)
#####################################
```

`coef.CFA` *S3 coefficients for CFA*

Description

S3 coefficients method for object of class"CFA".

Usage

```
## S3 method for class 'CFA'
coef(object, ...)
```

Arguments

`object`	object of class "CFA".
`...`	other parameters passed trough.

Value

Coefficients extracted from the model object of class"CFA".

`confreq_PACKAGE` *Configural Frequencies Analysis Using Log-linear Modeling*

Description

The package `confreq` offers some functions for Configural Frequencies Analysis (CFA) proposed by G.A. Lienert as an analysis of types and antitypes of persons or objects grouped according to their characteristic (response) pattern. The core principle in the package `confreq` is to use the function `glm` to compute the expected counts based on a model (design) matrix. The main functions are `CFA` and `S2CFA` (see details).

Details

The simplest entry to the package `confreq` is to use the main function `CFA`, which will compute several coefficients of Configural Frequencies Analysis at once.

More sophisticated control can be achieved by using the several single functions like `expected_cfa`, `design_cfg_cfa`, `chi_local_test_cfa` , `stirling_cfa` , etc. ...

Two-Sample-CFA, to detect discriminating pattern between two (sub-) samples, can be performed with the function `S2CFA`

For further description see description of the respective functions.

A good introduction into the theory and applications of Configural Frequencies Analysis is given in the Textbook 'Person-Centered Methods' by Mark Stemmler (see references).

Additional Information: Some users running R on 'Linux like' OS distributions (like e.g. Ubuntu – and in rare cases MAC OS) might report trouble during installation of `confreq` due to the package dependency `gmp`, which is used in `confreq` to perform the exact binomial test. This (miss-)behavior can usually traced back to a missing of 'the GNU Multiple Precision Arithmetic Library' in the respective OS installation. To fix this, users might consider to run the following Ubuntu Linux command in a terminal to install the latest GMP Library:

'`sudo apt-get install libgmp3-dev`'.

For additional information see also https://www.mathemagix.org/www/mmdoc/doc/html/external/gmp.en.html and https://gmplib.org/.

Annotation: The foundations for this R-Package were established and discussed in Rothenberge (2011) and (finally) in Klagenfurt at FGME 2013 with Rainer Alexandrowicz and Mark Stemmler ...

Author(s)

- Joerg-Henrik Heine <jhheine@googlemail.com>
- R.W. Alexandrowicz (function `stirling_cfa()`)

References

von Eye, A. (2002). *Configural Frequency Analysis. Methods, Models, and Applications.* Mahwah, NJ, LEA.

Krauth, J., & Lienert, G. A. (1973). *Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin: ein multivariates nichtparametrisches Verfahren zur Aufdeckung von Typen und Syndromen; mit 70 Tabellen*. Freiburg; München: Alber Karl.

Lazarsfeld, P. F., & Henry, N. W. (1968). *Latent structure analysis*. Boston: Houghton Mifflin.

Lienert, G. A. (1978). *Verteilungsfreie Methoden in der Biostatistik (Band II)* [Non-parametrical methods in the field of biometrics (Vol. II)]. Meisenheim am Glan, Germany: Hain.

Lienert, G. A. (1971). Die Konfigurationsfrequenzanalyse: I. Ein neuer Weg zu Typen und Syndromen. *Zeitschrift für Klinische Psychologie und Psychotherapie, 19*(2), 99-115.

Stemmler, M. (2020). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables*. Cham Heidelberg New York Dordrecht London: Springer.

Stemmler, M., & Hammond, S. (1997). Configural frequency analysis of dependent samples for intra-patient treatment comparisons. *Studia Psychologica, 39*, 167–175.

Examples

```
#######################################
######### some examples ########
data(LienertLSD)
LienertLSD
CFA(LienertLSD)
## testing with (full) interactions
CFA(LienertLSD,form="~ C + T + A + C:T + C:A + T:A + C:T:A")
## testing the null model
CFA(LienertLSD,form="null")
######################
data(suicide)
suicide
# suicide data is in non tabulated data representation
# so it must be tabulated !
CFA(dat2fre(suicide))
```

---

dat2cov *Summarize Covariates by Configuration*

Description

Compute one numeric summary per covariate and configuration, in the order of the corresponding complete pattern-frequency table. Given a dataset `x`, this function returns summary values for some (`numeric`) covariate variables in `x` for each pattern (configuration) defined by a set of factor variables in `x`.

Usage

```
dat2cov(
  x,
  FUN = "mean",
  ...,
  notobs = 0,
  katorder = FALSE,
  caseorder = TRUE,
  wgt = NULL,
  variables = NULL,
  covariates = NULL
)
```

Arguments

`x`	A data frame or matrix containing configuration variables and numeric covariates.
`FUN`	A function or function name returning one numeric value per group and covariate. Defaults to `"mean"`. When `wgt` is supplied, only `mean` is supported and weighted means are computed.

...	Additional arguments passed to FUN, or to weighted.mean for weighted means, e.g. na.rm = TRUE. Not forwarded to dat2fre().
notobs	Numeric replacement for unobserved configurations. Defaults to zero. Supply one value for all covariates or one value per selected covariate, in the order of covariates. NA is allowed.
katorder	Logical; see dat2fre. Use the same setting for the corresponding pattern-frequency table.
caseorder	Logical; see dat2fre. Use the same setting for the corresponding pattern-frequency table.
wgt	Optional vector of finite, nonnegative case weights, one per input row. Zero weights are allowed.
variables	Optional character vector of configuration column names. By default, all factor columns of x are selected.
covariates	Optional character vector of numeric covariate column names. By default, all numeric columns not selected in variables are used. Explicitly exclude identifiers and weight columns if necessary.

Details

At least one configuration variable and one numeric covariate are required. Explicit selection allows numeric or character configuration variables without first converting them to factors. Factor levels can specify unobserved categories. Configuration and covariate selections must be disjoint and contain distinct existing column names.

notobs applies only when no input record belongs to a configuration. Missing or undefined summaries in observed groups remain NA or NaN; their handling is controlled by FUN and its arguments. A group containing records with zero weights is still observed, even if its weighted mean is undefined. Missing configuration values are excluded according to the default dat2fre() behavior.

Names passed to variables and covariates must be written in full because these arguments follow .... For use with CFA(), use the same configuration columns and ordering options in dat2fre().

Value

A data frame of class c("data.frame", "Pcov"), containing the covariate summaries. The configurations attribute stores the corresponding configuration columns. Row names are readable labels made unique if necessary, not keys used for internal matching.

See Also

dat2fre, CFA

Examples

```
d <- data.frame(A = factor(c("yes", "yes", "no")),
                B = factor(c("low", "low", "high")),
                Age = c(20, 40, 60), Weight = c(1, 3, 2))
cov <- dat2cov(d, variables = c("A", "B"), covariates = "Age",
```

```
                wgt = d$Weight, notobs = NA_real_)
cov
attr(cov, "configurations")
```

---

dat2fre *Convert Individual Data to Pattern Frequencies*

Description

Tabulate individual records as a complete table of configurations, including configurations with zero observed frequency. Given a dataset this function returns a (response) pattern frequencies table representation of it.

Usage

```
dat2fre(
  x,
  katorder = FALSE,
  caseorder = TRUE,
  kat = NULL,
  codes = NULL,
  wgt = NULL,
  ...
)
```

Arguments

x	A data frame containing factor, character, numeric or logical columns, or a numeric matrix of positive integer category codes. Column names must be unique; `Freq` is reserved for the output.
katorder	Logical. Reorder variables by increasing number of categories? Defaults to `FALSE`.
caseorder	Logical. If `TRUE` (default), order configurations by category order with the rightmost variable changing fastest. If `FALSE`, retain the order produced by `table()`.
kat	For numeric matrices only: a vector specifying the number of categories per column. Codes must be in `1:kat[j]`. If omitted, column maxima determine the category counts.
codes	For numeric matrices only: a list with one entry per column and one unique label per category. For named entries, names are used as labels; otherwise, entry values are used as labels.
wgt	Optional numeric vector of finite, nonnegative case weights, one per input row. Zero weights are allowed.
...	Named options passed to table: `exclude`, `useNA`, `dnn` and `deparse.level`. Other or unnamed arguments are rejected.

Details

For data frames, factor levels define available categories and their order, including unused levels. Other columns use observed categories in the order determined by `table()`. Use factor levels to specify additional categories. The arguments `kat` and `codes` are rejected for data frames rather than silently ignored.

Numeric matrices retain the historical assumption of codes starting at one. Use a data frame for other coding schemes or character labels. Missing matrix codes are rejected. For data frames, the default table options exclude missing categories; `useNA = "ifany"` includes them.

The `Freq` column always contains unweighted counts. If weights are supplied, `wgtfreq` stores cell sums of weights, `WGT` stores the mean weight per observed cell (one for unobserved cells), and `wgtdat` stores the original weight vector. A comment marks the object as weighted for compatibility with `CFA()`. The additional attributes `wgtfreq2` (cell sums of squared weights) and `wgtpositive` (cell counts of positive-weight records) support the independent-unit survey variance calculation in `S2CFA()`. Both attributes follow exactly the row order of the frequency table. Use the default `caseorder = TRUE` for `CFA()` and use identical variable selection and ordering options when preparing covariates.

Value

A data frame of class `c("Pfreq", "data.frame")`, with one factor column per configuration variable and a final `Freq` column. Weighted inputs additionally produce the attributes described above.

References

No references in the moment

See Also

`fre2dat`, `dat2cov`, `fre2tab`, `CFA`, `S2CFA`

Examples

```
#######################################
d <- data.frame(A = factor(c("yes", "no", "yes")),
                B = factor(c("low", "low", "high")))
pf <- dat2fre(d)
pf
weighted <- dat2fre(d, wgt = c(1, 2, 3))
attr(weighted, "wgtfreq")
#######################################

data(suicide)# loading data in data frame (702 cases) representation
dat2fre(suicide) # converting it into a pattern frequencies table

###########
#######################################
data(LienertLSD)# loading example pattern frequencies table ..
test<-fre2dat(LienertLSD)# and coverting it into a simple (data) matrix
test<-test[sample(c(1:65),65),] # making a messy order
############
```

```
dat2fre(test) # making a proper ordered pattern frequencies table again
##### try it with a data.frame too!
#######################################
```

`design_cfg_cfa` *Designmatrix for log linear CFA models*

Description

Calculates the designmatrix corresponding to a dataset with `length(kat)` columns (variables).

Usage

```
design_cfg_cfa(
  kat,
  form = paste("~", paste(paste("V", 1:length(kat), sep = ""), collapse = " + ")),
  ...
)
```

Arguments

`kat`	a numerical vector containing kardinal numbers, giving the number of categories for each variable of a dataset (in the respective order of the variables in such a dataset) which corresponds to the requested designmatrix. So the length of this numerical vector represents the number of variables.
`form`	a character string which can be coerced into a model formulae with the function `as.formula` in the package `stats`. If this argument is left empty the function `design_cfg_cfa()` will return a designmatrix coding only main effects and no interactions – for a designmatrix refering to three variables for example, leaving the argument `form` empty will be equivalent to assigning the character `"~ V1 + V2 + V3"` to the argument (`form="~ V1 + V2 + V3"`). A special Case is to define a null-model or rather a cfa model of order zero. In such a model no (main) effects are considered. This can be achieved bei passing the character expression `"null"` to the argument `form` – so: `form = "null"`
`...`	additional parameters passed through to function `model.matrix` in package `stats`.

Details

This function internaly calls the function `pos_cfg_cfa`.

For further information on designmatrices see decription on function `model.matrix` in the package `stats`.

Value

A designmatrix - an object of class `c("matrix","design_cfg_cfa")` - for the formula therm given in argument`form`.

References

No references in the moment

Examples

```
#####################################
# designmatrix with three main effects.
# three variables with two categories each.
design_cfg_cfa(kat=c(2,2,2))
# two variables with two categories each and one variable
# with 7 categories (Lienert LSD example).
design_cfg_cfa(kat=c(2,2,7))
###########
# designmatrix with three main effects an three interactions.
# three variables with two categories each.
design_cfg_cfa(kat=c(2,2,2),form="~ V1 + V2 + V3 + V1:V2 + V1:V3 + V2:V3")
# two variables with two categories each and one variable
# with 7 categories (e.g. The Krauth & Lienert suicide Data).
design_cfg_cfa(kat=c(2,2,7),form="~ V1 + V2 + V3 + V1:V2 + V1:V3 + V2:V3")
#####################################
```

---

df_des_cfa *Degrees of freedom*

Description

Calculates the degrees of freedom based on an designmatrix for a (log liniear) CFA model.

Usage

```
df_des_cfa(des)
```

Arguments

des	a designmatrix (object of class "matrix") as returned by function design_cfg_cfa.

Details

No details

Value

An object of class "integer" giving the degrees of freedom for the designmatrix defined in argument des.

References

No references in the moment

Examples

```
#######################################
# degrees of freedom for designmatrix with three main effects.
# three variables with two categories each.
df_des_cfa(design_cfg_cfa(kat=c(2,2,2)))
# two variables with two categories each and one variable
# with 7 categories (e.g. The Krauth & Lienert suicide Data).
df_des_cfa(design_cfg_cfa(kat=c(2,2,7)))
###########
# degrees of freedom for designmatrix with three main effects
# and three 'two by two'interactions.
# and tripple interaction --> saturated model --> df=0
# three variables with two categories each.
df_des_cfa(design_cfg_cfa(kat=c(2,2,2),form="~ V1 + V2 + V3 + V1:V2 + V1:V3 + V2:V3 + V1:V2:V3"))
#######################################
```

`expected_cfa` *Expected frequencies with glm*

Description

Calculates the expected frequencies of counts using log liniear model.

Usage

```
expected_cfa(des, observed, family = poisson(), intercept = FALSE, ...)
```

Arguments

`des`	a designmatrix (object of class `"matrix"`) as returned by function `design_cfg_cfa`.
`observed`	a integer vector with `lenght(observed) == dim(des)[1]`. WARNING: The observed frequencies counts must be in an order corresponding to the coding sheme in designmatix (see argument `des`).
`family`	argument passed to `glm.fit` with default set to `poisson()`
`intercept`	argument passed to `glm.fit` with default set to `FALSE`
`...`	aditional arguments optional passed to `glm.fit`

Details

No details

Value

An vector object giving the expected counts.

References

No references in the moment

Examples

```
#####################################
# expected counts for LienertLSD data example.
designmatrix<-design_cfg_cfa(kat=c(2,2,2)) # generate an designmatrix (only main effects)
data(LienertLSD) # load example data
observed<-LienertLSD[,4] # extract observed counts
expected_cfa(des=designmatrix, observed=observed) # calculation of expected counts
#####################################
```

`expected_margin_cfa` *Expected frequencies using margins*

Description

Calculates the expected frequencies of counts based on the margins of the k-dimensional contingency table.

Usage

```
expected_margin_cfa(Pfreq, blank = NULL)
```

Arguments

`Pfreq`	Object of class "Pfreq" (see. function `dat2fre`).
`blank`	Either (1) character vector defining the pattern (with spaces between variable categories), which will be ignored for calculation of expected frequencies; or (2) a numeric vector defining the position(s) of the pattern in object of class `"Pfreq"`, which will be ignored for calculation of expected frequencies. At default (`blank=NULL`) all possible pattern, as listed in object of class `"Pfreq"`, are included for calculation of expected frequencies.

Details

only main effects are considered.

Value

An vector object giving the expected counts.

References

No references in the moment

Examples

```
#####################################
# expected counts for LienertLSD data example.
data(LienertLSD) # load example data
expected_margin_cfa(Pfreq = LienertLSD) # calculation of expected counts (only main effects).
#####################################
```

`fre2dat` *pattern frequency to dataset conversion*

Description

Given a (response) pattern frequencies table this function returns a dataset representation of it.

Usage

```
fre2dat(x, fact = TRUE, ...)
```

Arguments

`x`	A pattern-frequency data frame or matrix, with configuration variables in all but the last column. The last column must contain finite, nonnegative integer frequencies.
`fact`	Logical. If `TRUE` (default), return a data frame with factor columns. If `FALSE`, return a numeric matrix using `data.matrix`; factors are represented by integer codes.
`...`	Arguments passed to `factor` for each column when `fact = TRUE`. Ignored when `fact = FALSE`. WARNING using this option will only work correct when all 'pattern' columns (variables) in the frequencies table share the same number of categories

Details

The order of configurations in the input is retained; each row is repeated according to its frequency. Zero-frequency rows produce no records, and incomplete configuration tables can be expanded directly. With `fact = TRUE` and no additional factor arguments, existing factor levels and labels are retained. Other columns are converted to factors. Additional factor arguments apply to every column, so a shared `labels` argument must be appropriate for all variables.

Weighted-frequency attributes are not expanded into individual weights. Expansion always uses the final frequency column. Fractional frequencies are rejected rather than rounded. A round trip need not restore the original individual row order.

Value

A data frame with factor columns or a numeric matrix, according to `fact`, with one row per reconstructed individual record.

References

No references in the moment

See Also

`dat2fre`, `fre2tab`

Examples

```
#####################################
pf <- data.frame(A = factor(c("yes", "no")),
                 B = factor(c("low", "high")), Freq = c(2, 1))
fre2dat(pf)
fre2dat(pf, fact = FALSE)
dat2fre(fre2dat(pf))
#################################
data(LienertLSD)# loading example pattern frequencies table
fre2dat(LienertLSD)# coverting it into a (data) matrix
# for a matrix without colnames
colnames(LienertLSD)<-NULL # first removing the colnames
fre2dat(LienertLSD) # conversion with automatic new colnames
# requesting a data.frame using factor levels
fre2dat(LienertLSD,fact=TRUE,labels=c("yes","no"))
```

---

fre2tab *pattern frequency to table conversion*

Description

Given data as pattern frequencies (object of class class c("data.frame","Pfreq", see function dat2fre) this function returns a typical array representation (class "table" , see table) of it.

Usage

```
fre2tab(patternfreq, form = NULL, weighted = FALSE)
```

Arguments

patternfreq	an object of class c("data.frame","Pfreq")
form	a formula object with possibly both left and right hand sides specifying the order of the variables in the resulting table. At default ((formula=NULL)) all variables in (x) are used in their respective order. With two sides, dimensions on the right precede those on the left, matching the previous structable() conversion. A left-hand side of Freq denotes frequencies rather than a configuration dimension, e.g. Freq ~ A + B. Omitting variables sums over those dimensions. Only variable names joined by + are supported; transformations and interactions are rejected.
weighted	Logical. If FALSE (default), use the final frequency column. If TRUE, use the wgtfreq attribute and report an error if this attribute is absent or invalid.

Details

This function was introduced in order to connect the typical confreq data representation in the objects of the class c("data.frame","Pfreq"), see function dat2fre, to the R-typical array representation as it exists in objects of the "table" class, see table. This array representation of

multi-dimensional contingency tables is used more universally in R – e.g. also in the R package vcd, see the examples section below.

It is assumed, that the last column of the object patternfreq represents the frequencies of the (response) pattern represented by the other columns in patternfreq.

Unused factor levels are retained. Duplicate configurations are summed. Missing configuration categories are retained as table levels. No Pfreq class assignment is needed for ordinary frequency data. The choice of weighted affects only this conversion; it does not change the handling of weights by CFA() or S2CFA().

Value

An object of class "table", with named dimensions in the input order or the order specified by form.

An object of class "table", with named dimensions in the input order or the order specified by form; see table.

References

No references at the moment

See Also

dat2fre, fre2dat, xtabs

Examples

```
########################################
d <- data.frame(A = factor(c("yes", "no", "yes")),
                B = factor(c("low", "low", "high")))
pf <- dat2fre(d, wgt = c(1, 2, 3))
fre2tab(pf)
fre2tab(pf, weighted = TRUE)
fre2tab(pf, form = ~ B + A)
fre2tab(pf, form = A ~ B)
fre2tab(pf, form = Freq ~ A + B)
fre2tab(pf, form = ~ A)
#########
data(LienertLSD)# loading example pattern frequencies table
fre2tab(LienertLSD)# coverting it into a table

### examples using functions from package vcd
data(Lienert1978)# loading example pattern frequencies table
fre2tab(Lienert1978)# coverting it into a table
strucplot(fre2tab(Lienert1978))# plotting data with 'vcd'
structable(fre2tab(Lienert1978),direction = "v")# flatten table (vertical) with 'vcd'

# changing the vertical grouping when flattening the table by unsing a 'formula':
structable(fre2tab(Lienert1978, form=~Group + Student + Teacher),direction = "v")# flatten table
```

ftab *Tabulating Answer Categories in Data*

Description

Function tabulating (answer) categories in X.

Usage

```
ftab(X, catgories = NULL, na.omit = FALSE)
```

Arguments

X	Data as a "matrix", a "data.frame" or even a "vector" or "factor". "vector" or "factor" are coerced to a "data.frame" with one column.
catgories	optional a vector ("numeric" or "character") containig the categories to tabulate. At default (catgories=NULL) the fuction looks for unique categories in X.
na.omit	logical (default: na.omit=FALSE) wether to return frequencies for missing values, NAs.

Details

X can either be a ("numeric" or "character") "matrix" containing response vectors of persons (rows) or a "data.frame" containing "numeric", "character" or "factor" variables (columns).

Value

a "matrix" with category frequencies

Examples

```
########
data(suicide)
ftab(suicide)
```

lazar *The Data Example from Lazarsfeld and Henry*

Description

data example by Lazarsfeld and Henry (1968) where n = 1000 subjects need to solve questions or problems (i.e., A,B, and C). They either '1' = solved or '2' = did not solve the problems. The data is in pattern frequencies table representation (object of class c("data.frame", "Pfreq")). This data example is used in the textbook by Mark Stemmler (2020, Table 6.6, p. 81).

Usage

```
data(lazar)
```

Format

A matrix with 4 columns and 8 rows. The last column gives the frequencies for the (response) pattern in column 1:3.

Details

No detail in the moment

References

Lazarsfeld, P. F., & Henry, N. W. (1968). *Latent structure analysis*. Boston: Houghton Mifflin.

Stemmler, M. (2020). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables*. Cham Heidelberg New York Dordrecht London: Springer.

Examples

```
############################################################
data(lazar)
dim(lazar)
############################################################
```

Lienert1978 *The Lienert (1978) Data*

Description

Data used as an example for two-sample CFA in the textbook by Mark Stemmler (2020, Table 7.7, p.97) taken from Lienert (1978, p. 978). The data is in pattern frequencies table representation (object of class c("data.frame", "Pfreq")).

Usage

```
data(Lienert1978)
```

Format

A data frame (object of class `c("data.frame", "Pfreq")`) with 4 columns and 12 rows. The last column gives the frequencies for the (response) pattern in column 1:2 of the respective 'Group' given in column 3.

Details

no details at the moment ...

References

Lienert, G. A. (1978). *Verteilungsfreie Methoden in der Biostatistik (Band II)* [Non-parametrical 168 methods in the field of biometrics (Vol. II)]. Meisenheim am Glan, Germany: Hain.

Stemmler, M. (2020). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables*. Cham Heidelberg New York Dordrecht London: Springer.

Examples

```
data(Lienert1978)
dim(Lienert1978)
############################################################
colnames(Lienert1978) # show all variable names of Lienert1978
```

`LienertLSD` *The Lienert LSD Data*

Description

Data from the classical Lienert LSD trial as an example for CFA (see Lienert, 1971, p. 103, 'Tabelle 1'). The data is in pattern frequencies table representation (object of class c("data.frame", "Pfreq")).

Usage

```
data(LienertLSD)
```

Format

A data frame (object of class `c("data.frame", "Pfreq")`) with 4 columns and 8 rows. The last column gives the frequencies for the observed pattern of the psychotoxic basic syndrome (in column 1:3) due to the intake of lysergic acid diethylamide (LSD).

Details

The first three columns are named C, T and A which are abbreviations for the observed symptoms after taking LSD:

C = narrowed consciousness

T = thought disturbance

A = affective disturbance

The coding of the observations is for all symptoms: present='+' and absent='-'

References

Lienert, G. A. (1971). Die Konfigurationsfrequenzanalyse: I. Ein neuer Weg zu Typen und Syndromen. *Zeitschrift für Klinische Psychologie und Psychotherapie, 19*(2), 99-115.

Examples

```
data(LienertLSD)
dim(LienertLSD)
############################################################
colnames(LienertLSD) # show all variable names of matrix LienertLSD
```

`lr` *Likelihood Ratio Chi-square (LR)*

Description

Calculates the likelihod ratio chi-square statistic based on observed and expected counts.

Usage

```
lr(observed, expected)
```

Arguments

`observed`	a vector giving the observed frequencies.
`expected`	a vector giving the expected frequencies.

Details

No details in the moment.

Value

numeric giving the likelihood ratio chi-square statistic.

References

Stemmler, M. (2014). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables*. Cham Heidelberg New York Dordrecht London: Springer.

Examples

```
#######################################
######### some examples ########
data(newborns)
newborns
designmatrix <- design_cfg_cfa(kat=c(2,2)) # generate an designmatrix (only main effects)
observed <- newborns[,3] # extract observed counts
expected <- expected_cfa(des=designmatrix, observed=observed) # calculation of expected counts
lr(observed,expected) # calculation of the likelihood ratio chi-square statistic
```

`newborns` *The Data Example from Stemmler 2020*

Description

data example by Stemmler (2020, table 4.1, p. 33) where n = 56 newborns 'with seizures' = 1 or 'without seizures' = 2 (coded in the in first column named 'A') were tested with an intelligence test while they attended kindergarten. Children's intelligence was divided into 'average or above' = 1 and 'below average' = 2 (coded in the in second column named 'B'). The third column gives the frequencies of the respective pattern.

Usage

```
data(newborns)
```

Format

A data.frame with 3 columns and 4 rows. The last column gives the frequencies for the observed pattern in column 1:2. The data is in pattern frequencies table representation (object of class c("data.frame", "Pfreq"))

Details

No detail in the moment

References

Stemmler, M. (2020). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables*. Cham Heidelberg New York Dordrecht London: Springer.

Examples

```
############################################################
data(newborns)
dim(newborns)
newborns
############################################################
```

plot.CFA *S3 plot for CFA*

Description

S3 plot method for object of class"CFA"

Usage

```
## S3 method for class 'CFA'
plot(
  x,
  type = "z.pChi",
  fill = c("red", "blue", "grey"),
  adjalpha = "bonferroni",
  ...
)
```

Arguments

x	object of class "CFA"
type	character indicating which test to use for visualizing whether the observed pattern are 'Types', 'Antitypes' or not significant at all. Possible options for type are "pChi", "ex.bin.test", "z.pChi", "z.pBin" and "p.stir".
fill	a vector of (three) colors defining the coloring of significant 'Types' (default "red"), 'Antitypes' (default "blue") or not significant cells (default "grey") in the plot.
adjalpha	Adjustment method: "bonferroni" (default), "holm", "hochberg", "hommel", "BH", "BY", "fdr" (alias for BH), or "none".
...	other parameters passed trough

Details

Plot frequencies use the same raw or weighted frequency basis as the CFA result. The existing CFA inferential calculations are unchanged; this plotting correction does not supply a design-based survey variance. Additional adjustment methods are "hochberg", "hommel", "BH", "BY" and "fdr" (an alias for "BH"). These methods use p.adjust on computable p-values only, following the same test-family rule as Holm. Hochberg and Hommel require appropriate dependence assumptions; BH and BY control the false discovery rate rather than the family-wise error rate. See p.adjust for assumptions and statistical interpretation. The historical Bonferroni denominator and the "none" behavior are retained. In particular, the reduced test-family rule for Holm and the new methods is not silently applied to the existing Bonferroni decisions. With adjalpha = "holm", decisions use stats::p.adjust(p, method = "holm") on the non-missing p-values only. Missing values (NA/NaN) do not contribute to the number of comparisons. The historical strict rule, adjusted p-value less than alpha, is retained. Decisions use full precision before rounding. Only finite values in [0, 1] are computable p-values. Other numeric values are excluded with a warning and treated

as missing decisions. Missing decisions are drawn using the neutral fill color. Functional cells are excluded from the Holm test family and retain their "b" label. Other pre-existing adjustment methods retain their decisions.

Value

a plot visualizing the results.

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Bonferroni, C. E. (1935). Il calcolo delle assicurazioni su gruppi di teste. In S.O. Carboni (Ed.), *Studi in Onore del Professore Salvatore Ortu Carboni* (S. 13–60). Roma, Tipografia del Senato: Bardi.

Examples

```
data(LienertLSD)
res <- CFA(LienertLSD, bintest = FALSE)
plot(res, type = "pChi", adjalpha = "BH")
```

---

plot.S2CFA *S3 plot for S2CFA*

Description

S3 plot method for object of class"S2CFA"

Usage

```
## S3 method for class 'S2CFA'
plot(
  x,
  type = "ex.fisher.test",
  fill = c("red", "grey"),
  adjalpha = "bonferroni",
  ...
)
```

Arguments

x	object of class"S2CFA"
type	Test column: "ex.fisher.test" (unweighted default), "pChi", or "p.wald". For survey-weighted results, an omitted type automatically selects "p.wald"; count-table tests cannot be requested for these weighted results. Conversely, p.wald is unavailable for unweighted results. Unavailable tests produce an informative error before adjustment or plotting starts.

`fill`	a vector of (two) colors defining the coloring of discriminating 'Types' (default "red"), or not discriminating cells (default "grey") in the plot.
`adjalpha`	Adjustment method: `"bonferroni"` (default), `"holm"`, `"hochberg"`, `"hommel"`, `"BH"`, `"BY"`, `"fdr"` (alias for BH), or `"none"`.
`...`	other parameters passed trough.

Details

Survey-weighted frequencies and Wald p-values are taken from the weighted `S2CFA()` result; no integer rounding is performed. The test assumes independent sampling units, without cluster or stratum information. Non-computable p-values do not contribute to the Holm and additional p.adjust test families. Summary and plot select the same test. Additional adjustment methods are `"hochberg"`, `"hommel"`, `"BH"`, `"BY"` and `"fdr"` (an alias for `"BH"`). These methods use `p.adjust` on computable p-values only, following the same test-family rule as Holm. Hochberg and Hommel require appropriate dependence assumptions; BH and BY control the false discovery rate rather than the family-wise error rate. See `p.adjust` for assumptions and statistical interpretation. The historical Bonferroni denominator and the `"none"` behavior are retained. In particular, the reduced test-family rule for Holm and the new methods is not silently applied to the existing Bonferroni decisions. With `adjalpha = "holm"`, decisions use `stats::p.adjust(p, method = "holm")` on the non-missing p-values only. Missing values (`NA`/`NaN`) do not contribute to the number of comparisons. The historical strict rule, adjusted p-value less than `alpha`, is retained. Decisions use full precision before rounding. Only finite values in `[0, 1]` are computable p-values. Other numeric values are excluded with a warning and treated as missing decisions. Missing decisions are drawn using the neutral fill color.

Value

Draws a plot and invisibly returns a list containing the table, fill array, significance decisions, configuration mapping, and additional entries `p.adj` and `adjalpha`.

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Bonferroni, C. E. (1935). Il calcolo delle assicurazioni su gruppi di teste. In S.O. Carboni (Ed.), *Studi in Onore del Professore Salvatore Ortu Carboni* (S. 13–60). Roma, Tipografia del Senato: Bardi.

Examples

```
data(Lienert1978)
res <- S2CFA(Lienert1978)
plot(res, adjalpha = "BY")
```

pos_cfg_cfa *Possible configurations*

Description

Calculates all possible configuartions for some variables with different numbers of categories.

Usage

```
pos_cfg_cfa(kat, fact = FALSE)
```

Arguments

kat	a numerical vector containing kardinal numbers, giving the number of categories for each variable. So the length of this numerical vector represents the number of variables.
fact	logical, default is (fact=FALSE). If this argument is set to (fact=TRUE) the result is coerced to a data.frame with factor variables.

Details

No details

Value

An object of class "matrix" or "data.frame" (depending on the argument fact) containing all possible configurations for lenght(kat) variables with the respective number of categories given as kardinal numbers in the vector kat.

References

No references in the moment

Examples

```
#######################################
# possible configurations for ...
# three variables with two categories each (Lienert LSD example).
pos_cfg_cfa(kat=c(2,2,2))
#######################################
```

print.Pfreq *S3 print for Pfreq*

Description

S3 print method for object of class"Pfreq"

Usage

```
## S3 method for class 'Pfreq'
print(
  x,
  ...,
  digits = NULL,
  quote = FALSE,
  right = TRUE,
  row.names = TRUE,
  max = NULL
)
```

Arguments

x	object of class "Pfreq"
...	further arguments passed to or from other methods.
digits	minimal number of significant digits, see print.default.
quote	logical, indicating whether or not strings should be printed with surrounding quotes.
right	logical, indicating whether or not strings should be right aligned.
row.names	logical (or character vector), indicating whether (or what) row names should be printed.
max	numeric or NULL, specifying the maximal number of entries to be printed. By default, when NULL, getOption("max.print") used.

Value

output printed to the console

`Q_determination_bound_cfa` *Theoretical Upper Bound for the Determination Coefficient*

Description

Calculates the theoretical upper bound of the local determination coefficient Q for a table with `T` variables (dimensions), following Theorem 1 in Betzin & Bollmann-Sdorra (2003).

Usage

```
Q_determination_bound_cfa(D)
```

Arguments

`D`	a numeric vector giving the number of variables (dimensions) of the table(s), `T` >= 1.

Details

For a T-dimensional table (i.e. 2^T cells) the empirical value of the determination coefficient `Q_determination_cfa` is bounded by

$$Q_I < 2^{(T-1)/T} - 1$$

and this bound is sharp (attained in the limit, for any $\epsilon > 0$ a table configuration exists that gets arbitrarily close to it). The bound increases with `T` but stays clearly below 1 even for moderately large `T` (e.g. ~0.682 for T = 4, ~0.866 for T = 10, ~0.986 for T = 100). The authors explicitly warn against using this bound to *normalize* Q (see Section 4 / Example 1 (p. 409) of the paper, where normalizing even reverses the ranking of two Q values from tables of different dimension). Comparisons of Q across tables of different dimension therefore remain problematic even after normalization.

Value

a numeric vector (same length as `T`) with the upper bound for Q.

References

Betzin, J., & Bollmann-Sdorra, P. (2003). On the determination coefficient in CFA. *Psychology Science, 45*(2), 400-420.

See Also

`Q_determination_cfa`

Examples

```
#######################################
# reproduces Table 1 of Betzin & Bollmann-Sdorra (2003)
T <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50, 75, 100)
round(Q_determination_bound_cfa(D = T), 3)
```

Q_determination_cfa *Local Determination Coefficient (Q) for CFA*

Description

Calculates Lienert's local determination coefficient (Q) for each cell of a contingency table, based on the observed and expected cell frequencies. The coefficient quantifies how strongly a single cell deviates from the independence model, relative to the largest deviation that is theoretically possible for that cell.

Usage

```
Q_determination_cfa(observed, expected, n = sum(observed))
```

Arguments

observed	numeric vector of observed (absolute) cell frequencies h_I
expected	numeric vector of expected (absolute) cell frequencies e_I of the same length as observed
n	a numeric defining the sample size with default set to sum(observed)

Details

The determination coefficient for a cell I is defined (Betzin & Bollmann-Sdorra, 2003, eq. 1; going back to Lienert, 1971) as

$$Q_I = \frac{h_I - e_I}{\max\{e_I,\, N - e_I\}}$$

where h_I is the observed and e_I the expected (absolute) frequency of cell I, and N is the total sample size. In relative frequencies ($f_I = h_I/N$, $p_I = e_I/N$) this is equivalent to

$$Q_I = \frac{f_I - p_I}{\max\{p_I,\, 1 - p_I\}}.$$

Q_I lies between -1 and 1 for a single cell and coincides with Lienert's (1971) original formula $Q_I = (h_I - e_I)/e_I$ whenever $2e_I \geq N$.

Note that values of Q_I from tables of different dimension are not directly comparable, even after normalizing by the bound – see Q_determination_bound_cfa.

Value

a numeric vector with the local determination coefficient Q_I for each cell, in the same order as observed / expected.

References

Betzin, J., & Bollmann-Sdorra, P. (2003). On the determination coefficient in CFA. *Psychology Science, 45*(2), 400-420.

Lienert, G. A. (1971). Die Konfigurationsfrequenzanalyse.*Zeitschrift für Klinische Psychologie und Psychotherapie, 19*, 99-115.

See Also

Q_determination_bound_cfa

Examples

```
#######################################
# first calculate expected counts for the LienertLSD data example.
# designmatrix with three main effects
designmatrix <- design_cfg_cfa(kat = c(2, 2, 2))
# load example data
data(LienertLSD)
# extract observed counts
observed <- LienertLSD[, 4]
expected <- expected_cfa(des = designmatrix, observed = observed)
Q_determination_cfa(observed, expected)
#######################################
# an easy, hand-checkable 2x2 example:
# margins: row totals e1. = 60 and e2. = 40,
# column totals e.1 = 50 and e.2 = 50, n = 100 -> e11 = 30
Q_determination_cfa(observed = 45, expected = 30, n = 100)
# (45-30) / max(30, 70) = 0.2142857
```

S2CFA *Configural Frequencies Analysis for two Samples.*

Description

Calculates coefficients for the two-sample CFA. Instead of differentiating between 'Types' and 'Antitypes', two-sample CFA looks for discrimination types, that is configurations with significant differences in frequencies between two sub samples.

Usage

```
S2CFA(patternfreq, alpha = 0.05, ccor = FALSE, ..., weighted = NULL)
```

Arguments

patternfreq	an object of class "Pfreq", which is data in pattern frequencies representation - see function dat2fre. The variable defining the two sub samples (a variable with max. two categories) must be located in the last but one column of the object of class "Pfreq"

alpha	a numeric giving the alpha level for testing (default set to alpha=.05)
ccor	For unweighted analysis only; TRUE is rejected for survey weights. A logical (TRUE / FALSE) determining whether to apply a continuity correction or not. When set to ccor=TRUE continuity correction is applied. For ccor=FALSE no continuity correction is applied.
...	Reserved for compatibility; not used by this implementation.
weighted	Logical or NULL. By default, use survey weights if the wgtfreq attribute is present. TRUE explicitly requests weighted analysis; FALSE explicitly uses raw frequencies. This argument follows ... and must be named in full.

Details

With survey weights, observed and expected frequencies use sums of weights, without rounding. Inference compares the two weighted proportions by an asymptotic Wald chi-squared test (one degree of freedom) using a linearized variance for independent sampling units, with replacement. No clustering, stratification, calibration or finite-population correction can be incorporated from the weight vector alone.

The influence for record i in group g is `w_i * (I_i - p_g) / sum(w_g)`. The two group contributions are disjoint. Their squared influences are summed and multiplied by `n / (n - 1)`, where n counts all positive-weight records in both groups. The variance is obtained from wgtfreq2 and wgtpositive, stored by the revised `dat2fre()`. Zero-variance configurations are not tested and have an NA p-value. For old weighted objects without these attributes, only weighted descriptive results are returned with a warning; recreate them with `dat2fre()`.

The weighted p-value is p.wald, not a Fisher exact test and not a Pearson test on population expansion totals. Classical test columns remain NA in weighted results. Multiplying all weights by a common positive factor leaves the Wald statistic and p-value unchanged. The unweighted statistical calculation retains its historical behavior. Both groups are paired by categories, even after a row permutation. Incomplete or duplicated configuration grids are rejected explicitly.

Value

An object of class S2CFA. Weighted results additionally contain `weighted = TRUE` and a weighting description. The first ten local result columns retain the historical layout; survey columns are Wald, df.wald, p.wald, prop.diff and SE.diff. Frequencies in weighted results are sums of weights.

References

Stemmler, M. (2020). *Person-Centered Methods – Configural Frequency Analysis (CFA) and Other Methods for the Analysis of Contingency Tables.* Cham Heidelberg New York Dordrecht London: Springer.

Stemmler, M., & Hammond, S. (1997). Configural frequency analysis of dependent samples for intra-patient treatment comparisons. *Studia Psychologica, 39*, 167–175.

Examples

```
#######################################
############### some examples #########
######### example from Marks Textbook
```

```
data(Lienert1978)
res1 <- S2CFA(Lienert1978)
summary(res1)
res2 <- S2CFA(Lienert1978, ccor=TRUE) # with continuity correction
summary(res2)
######### example with biger numbers
data(suicide)
ftab(suicide) # 'Epoche' may divide the sample into 2 subsamples
suicide_2s <- suicide[, c(1,3,2) ] # reorder data that 'Epoche' is the last column
ftab(suicide_2s) # check reordering
suicide_2s_fre <- dat2fre(suicide_2s)
res3 <- S2CFA(suicide_2s_fre)
summary(res3)
res4 <- S2CFA(suicide_2s_fre, ccor=TRUE) # with continuity correction
summary(res4)
```

`stirling_cfa` *Approximation to the binomial using Stirling's Formula*

Description

Calculates the binomial aproximation using stirling's formula (Version of function: V 1.0 - November 2013)

Usage

```
stirling_cfa(
  observed,
  expected = NULL,
  n = sum(observed),
  p = NULL,
  cum = T,
  verb = T
)
```

Arguments

`observed`	a integer vector with observed freqencies
`expected`	a vector giving the expected frequencies. expected can be set to `expected=NULL` if an vector of cell probabilities is given in argument `p`.
`n`	number of trials (scalar) default is `n = sum(observed)` .
`p`	a vector of cell probabilities. If p is not NULL the argument `expected` is ignored and this vector p of cell probabilities is used for calculatio instead of expected counts
`cum`	a logical - computation of cumulative density. If `cum=TRUE` (default) computes tail probability. If `cum=FALSE` computes prob. only for one cell (i.e. execute stircore only).
`verb`	logical - verbose results: If `verb=TRUE` (default) builds a results table. If `verb=FALSE` returns vector of cell p-values only.

Details

- Vector p must be of same length as observed _or_ p may be a scalar (e.g. in case of the zero-order CFA).
- The routine autoselects the upper or lower tail:
  - if obs > exp then sum obs:n
  - else sum 0:obs
- The stirling approximation cannot be evaluated if the observed frequency is 0 or n. Therefore, the proposal of A. von Eye (20xx) is adopted, taking the sum up to 1 or n-1, respectively.

Author(s)

R.W. Alexandrowicz

References

von Eye, A. (2002). *Configural Frequency Analysis. Methods, Models, and Applications.* Mahwah, NJ, LEA.

---

suicide *The Krauth & Lienert suicide Data*

Description

Data from the Krauth & Lienert suicide example for CFA (see Tables 39a and 39b; Krauth & Lienert, 1973). The data describe suicide patterns in pre- and post-WWII Germany – see von Eye, A., (2002); p. 385.

Usage

```
data(suicide)
```

Format

A `data.frame` with 3 columns (as factors). The data is in data list representation – each row represents one case.

Details

The three columns are named 'Geschlecht', 'Epoche' and 'Suizidart' which is 'gender', 'epoch' and 'type od suicide'. each of the variables are factors with the following levels:

Geschlecht: 'm' = 1 (male); 'w' = 2 (female)

Epoche: '44' = 1 (the epoch 1944); '52' = 2 (the epoch 1952)

Suizidart: 'Eh' = 1(hang); 'Es' = 2 (shoot); 'Et' = 3(drown); 'G' = 4(gas); 'H' = 5(crashing down); 'P' = 6(open vein); 'S' = 7(barbiturate);

References

Krauth, J., & Lienert, G. A. (1973). *Die Konfigurationsfrequenzanalyse (KFA) und ihre Anwendung in Psychologie und Medizin: ein multivariates nichtparametrisches Verfahren zur Aufdeckung von Typen und Syndromen; mit 70 Tabellen*. Freiburg; München: Alber Karl.

von Eye, A. (2002). Configural Frequency Analysis: Methods, models, and applications. Mahwah, N.J.: Lawrence Erlbaum Associates.

Examples

```
############################################################
data(suicide) # to load the data.frame included in the package
class(suicide)
dim(suicide)
str(suicide)
```

`summary.CFA` *S3 Summary for CFA*

Description

S3 summary method for object of class`"CFA"`

Usage

```
## S3 method for class 'CFA'
summary(
  object,
  digits = 3,
  type = "z.pChi",
  sorton = NULL,
  decreasing = FALSE,
  showall = TRUE,
  holm = FALSE,
  wide = FALSE,
  adjalpha = "bonferroni",
  ...
)
```

Arguments

`object`	object of class`"CFA"`
`digits`	integer rounds the values to the specified number of decimal places, default is `digits=3`.
`type`	character indicating which test to use for inference whether the observed pattern are 'Types', 'Antitypes' or not significant at all. Possible options for `type` are `"pChi"`, `"ex.bin.test"`, `"z.pChi"`, `"z.pBin"` and `"p.stir"`.

sorton	sort results of local test by any column. By default the output is not sorted. Other options may be "pat.", "obs.", "exp.", "Type", "Chi", etc. ... So all column names that can potentially appear in the result.
decreasing	logical. Should the sort be increasing or decreasing? see order
showall	logical with default showall = TRUE. To return only significant pattern ('Types' / 'Antitypes') set it to showall = FALSE.
holm	logical with default holm = FALSE. If set to holm = TRUE, significance testing is based on the holm procedure – see references. This argument is deprecated (since version 1.5.6) and kept only for downward compatibility. Use argument adjalpha for any type of alpha adjustment.
wide	logical with default wide = FALSE. If set to wide = TRUE, results for all significance tests are returned.
adjalpha	Adjustment method: "bonferroni" (default), "holm", "hochberg", "hommel", "BH", "BY", "fdr" (alias for BH), or "none". "holland" remains available in this summary method.
...	other parameters passed trough.

Details

Additional adjustment methods are "hochberg", "hommel", "BH", "BY" and "fdr" (an alias for "BH"). These methods use p.adjust on computable p-values only, following the same test-family rule as Holm. Hochberg and Hommel require appropriate dependence assumptions; BH and BY control the false discovery rate rather than the family-wise error rate. See p.adjust for assumptions and statistical interpretation. The historical Bonferroni denominator and the "none" behavior are retained. In particular, the reduced test-family rule for Holm and the new methods is not silently applied to the existing Bonferroni decisions. The final column p.adj contains the selected adjusted p-values (raw p-values for "none"); old columns keep their positions. Existing Holm.adj, Holm.crit and, where applicable, Holland.adj remain available. Classification precedes rounding. With adjalpha = "holm", decisions use stats::p.adjust(p, method = "holm") on the non-missing p-values only. Missing values (NA/NaN) do not contribute to the number of comparisons. The historical strict rule, adjusted p-value less than alpha, is retained. Decisions use full precision before rounding. Only finite values in [0, 1] are computable p-values. Other numeric values are excluded with a warning and treated as missing decisions. The additional column Holm.adj contains adjusted p-values. The existing Holm.crit column is retained as a diagnostic value: alpha / (m - r + 1), where m is the number of tested values and r is their rank (minimum rank for ties). A row-wise comparison with this column alone does not implement the sequential Holm procedure. Missing decisions are displayed as NA and are omitted when showall = FALSE. The existing Holm columns retain their positions. Functional cells are excluded from the Holm test family and retain their "b" label. Other pre-existing adjustment methods retain their decisions.

Value

A data frame of local results, with printed explanatory output. Includes a final p.adj column. For Holm adjustment, includes Holm.crit and Holm.adj.

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Bonferroni, C. E. (1935). Il calcolo delle assicurazioni su gruppi di teste. In S.O. Carboni (Ed.), *Studi in Onore del Professore Salvatore Ortu Carboni* (S. 13–60). Roma, Tipografia del Senato: Bardi.

Holland, B. S., & Copenhaver, M. D. (1987). An Improved Sequentially Rejective Bonferroni Test Procedure. *Biometrics, 43*(2), 417. https://doi.org/10.2307/2531823

Examples

```
data(LienertLSD)
res <- CFA(LienertLSD, bintest = FALSE)
summary(res, type = "pChi", adjalpha = "BH")
```

`summary.S2CFA` *S3 Summary for S2CFA*

Description

S3 summary method for object of class`"S2CFA"`

Usage

```
## S3 method for class 'S2CFA'
summary(
  object,
  digits = 3,
  type = "ex.fisher.test",
  sorton = NULL,
  decreasing = FALSE,
  showall = TRUE,
  adjalpha = "bonferroni",
  ...
)
```

Arguments

`object`	object of class`"S2CFA"`
`digits`	integer rounds the values to the specified number of decimal places, default is `digits=3`.
`type`	Test column: `"ex.fisher.test"` (unweighted default), `"pChi"`, or `"p.wald"`. For survey-weighted results, an omitted type automatically selects `"p.wald"`; count-table tests cannot be requested for these weighted results. Conversely, `p.wald` is unavailable for unweighted results. Unavailable tests produce an informative error before adjustment or plotting starts.

sorton	sort results of local test by any column. By default the output is not sorted. Other options may be "pat.", "disc.Type", etc. ... So all column names that can potentially appear in the result.
decreasing	logical. Should the sort be increasing or decreasing? see order
showall	logical with default showall = TRUE. To return only significant pattern (discriminating types) set it to showall = FALSE.
adjalpha	Adjustment method: "bonferroni" (default), "holm", "hochberg", "hommel", "BH", "BY", "fdr" (alias for BH), or "none".
...	other parameters passed trough.

Details

Survey-weighted frequencies and Wald p-values are taken from the weighted S2CFA() result; no integer rounding is performed. The test assumes independent sampling units, without cluster or stratum information. Non-computable p-values do not contribute to the Holm and additional p.adjust test families. Summary and plot select the same test. Additional adjustment methods are "hochberg", "hommel", "BH", "BY" and "fdr" (an alias for "BH"). These methods use p.adjust on computable p-values only, following the same test-family rule as Holm. Hochberg and Hommel require appropriate dependence assumptions; BH and BY control the false discovery rate rather than the family-wise error rate. See p.adjust for assumptions and statistical interpretation. The historical Bonferroni denominator and the "none" behavior are retained. In particular, the reduced test-family rule for Holm and the new methods is not silently applied to the existing Bonferroni decisions. The final column p.adj contains the selected adjusted p-values (raw p-values for "none"); old columns keep their positions. Existing Holm.adj, Holm.crit and, where applicable, Holland.adj remain available. Classification precedes rounding. With adjalpha = "holm", decisions use stats::p.adjust(p, method = "holm") on the non-missing p-values only. Missing values (NA/NaN) do not contribute to the number of comparisons. The historical strict rule, adjusted p-value less than alpha, is retained. Decisions use full precision before rounding. Only finite values in [0, 1] are computable p-values. Other numeric values are excluded with a warning and treated as missing decisions. The additional column Holm.adj contains adjusted p-values. The existing Holm.crit column is retained as a diagnostic value: alpha / (m - r + 1), where m is the number of tested values and r is their rank (minimum rank for ties). A row-wise comparison with this column alone does not implement the sequential Holm procedure. Missing decisions are displayed as NA and are omitted when showall = FALSE. The existing Holm columns retain their positions.

Value

A data frame of local results, with printed explanatory output. Includes a final p.adj column. For Holm adjustment, includes Holm.crit and Holm.adj.

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Bonferroni, C. E. (1935). Il calcolo delle assicurazioni su gruppi di teste. In S.O. Carboni (Ed.), *Studi in Onore del Professore Salvatore Ortu Carboni* (S. 13–60). Roma, Tipografia del Senato: Bardi.

Examples

```
data(Lienert1978)
res <- S2CFA(Lienert1978)
summary(res, adjalpha = "BY")
```

`z_tests_cfa` *Two z-Approximation Tests*

Description

Calculates the Chi-square approximation to the z-test and the binomial approximation to the z-test.

Usage

```
z_tests_cfa(observed, expected, ccor = FALSE, ntotal = sum(observed))
```

Arguments

`observed`	a vector giving the observed frequencies.
`expected`	a vector giving the expected frequencies.
`ccor`	either a logical (TRUE / FALSE) determining wether to apply a continuity correction or not to the Binomial Approximation of the z-Test. When set to `ccor=TRUE` continuity correction is applied for expected values 5 =< expected =< 10. For `ccor=FALSE` no continuity correction is applied. Another option is to set `ccor=c(x,y)` where x is the lower and y the upper bound for expected values where continuity correction is applied. So `ccor=c(5,10)` is equivalent to `ccor=TRUE`.
`ntotal`	optional a numeric giving the total number of observations. By default ntotal is calculated as `ntotal=sum(observed)`.

Details

An continuity correction can be applied to the binomial approximation – see argument `ccor`.

Value

a list with z and p-values.

References

No references in the moment

Examples

```
#######################################
# expected counts for LienertLSD data example.
designmatrix<-design_cfg_cfa(kat=c(2,2,2)) # generate an designmatrix (only main effects)
data(LienertLSD) # load example data
observed<-LienertLSD[,4] # extract observed counts
expected<-expected_cfa(des=designmatrix, observed=observed) # calculation of expected counts
z_tests_cfa(observed,expected)
#######################################
```

Index

∗ **datasets**
  lazar, 22
  Lienert1978, 23
  LienertLSD, 24
  newborns, 26
  suicide, 37
∗ **mainfunction**
  CFA, 5
  S2CFA, 34
∗ **methods**
  anova.CFA, 3
  coef.CFA, 9
  plot.CFA, 27
  plot.S2CFA, 28
  print.Pfreq, 31
  summary.CFA, 38
  summary.S2CFA, 40
∗ **misc**
  binomial_test_cfa, 4
  chi_local_test_cfa, 8
  design_cfg_cfa, 15
  df_des_cfa, 16
  expected_cfa, 17
  expected_margin_cfa, 18
  lr, 25
  pos_cfg_cfa, 30
  Q_determination_bound_cfa, 32
  Q_determination_cfa, 33
  stirling_cfa, 36
  z_tests_cfa, 42
∗ **utilities**
  [.Pfreq, 2
  dat2cov, 11
  dat2fre, 13
  fre2dat, 19
  fre2tab, 20
  ftab, 22
[.Pfreq, 2

anova *(*anova.CFA*)*, 3

anova.CFA, 3
anova.glm, *4*

binomial_test_cfa, 4

CFA, 5, *9, 10*, *12*, *14*
chi_local_test_cfa, 8, *10*
coef.CFA, 9
coefficients *(*coef.CFA*)*, 9
confreq-package *(*confreq_PACKAGE*)*, 9
confreq_PACKAGE, 9

dat2cov, *6*, 11, *14*
dat2fre, *3*, *6, 7*, *12*, 13, *18–21*, *34*
data.matrix, *19*
design_cfg_cfa, *10*, 15
df_des_cfa, 16

expected_cfa, *10*, 17
expected_margin_cfa, 18

factor, *19*
fre2dat, *14*, 19, *21*
fre2tab, *3*, *14*, *19*, 20
ftab, 22

getOption, *31*
glm, *5, 6*, *9*
glm.fit, *6*, *17*

lazar, 22
Lienert1978, 23
LienertLSD, 24
lr, 25

newborns, 26

order, *39*, *41*

p.adjust, *27*, *29*, *39*, *41*
plot.CFA, *7*, 27

plot.S2CFA, 28
pos_cfg_cfa, *15*, 30
print.default, *31*
print.Pfreq, 31

Q_determination_bound_cfa, 32, *33, 34*
Q_determination_cfa, *32*, 33

S2CFA, *9, 10*, *14*, 34
stirling_cfa, *10*, 36
suicide, 37
summary.CFA, *7*, 38
summary.S2CFA, 40

table, *13*, *20, 21*

weighted.mean, *12*

xtabs, *21*

z_tests_cfa, 42