# Package ‘confreq’

November 6, 2014

**Type** Package

**Title** Configural Frequencies Analysis Using Log-linear Modeling

**Version** 1.3

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 2.10.1), stats, gmp

**Date** 2014-11-6

**Author** Joerg-Henrik Heine, R.W. Alexandrowicz (function stirling_cfa() ) and some package testing by Mark Stemmler

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

**Description** Several functions for Configural Frequencies Analysis - introduced by G. A. Lienert as 'Konfigurations Frequenz Analyse' (KFA)

## R topics documented:

---

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 Texbook 'Person-Centered Methods' by Mark Stemmler (see references).

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

### 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. (2014). *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))
```

---

`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 (optionaly) 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 formular definition given in argument `form`.

## Usage

```
CFA(patternfreq, alpha = 0.05, form = NULL, ccor = FALSE,
  family = poisson(), intercept = FALSE, ...)
```

## 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 refering 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` musst be of class `"matrix"` and must logicaly match to the argument `patternfreq`, which is currently not checked! - but simply assumed. |
| `ccor` | either a logical (TRUE / FALSE) determining wether to apply a continuity correction or not. 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.fit` with default set to `poisson()` |
| `intercept` | argument passed to `glm.fit` with default set to `FALSE` |
| `...` | additional parameters passed through to other functions. |

## Details

This is the main function of the package. It internaly calls several functions of the package `confreq` which are also available as single functions. For clasification of the observed patterns into 'Types' and 'Antitypes' according to Linert (1971), a S3 summary method for the resulting object of class `"CFA"` can be applied - see `summary.CFA`.

## 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.

## Examples

```
#######################################
######### some examples ########
data(LienertLSD)
LienertLSD
res1 <- CFA(LienertLSD)
summary(res1)
## testing with (full) interactions
res2 <- CFA(LienertLSD,form="~ C + T + A + C:T + C:A + T:A + C:T:A")
summary(res2)
#' ## testing the null model
res3 <- CFA(LienertLSD,form="null")
summary(res3)
#######################
data(suicide)
suicide
# suicide data is in non tabulated data representation - so it must be tabulated !
res4 <- CFA(dat2fre(suicide))
summary(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)
#######################################
```

---

## `dat2fre` *dataset to pattern frequency conversion*

---

### Description

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)
```

### Arguments

| | |
|---|---|
| `x` | an object of class "matrix" or "data.frame". If `x` is a "data.frame" each variable (column) must be an `integer` or a `factor`. If `x` is a "matrix" it is assumed that the categories for each variable in `x` start with `1` – there is no check for that !!! |
| `katorder` | logical with default set to `katorder==FALSE`. When set to `katorder==TRUE` variables are ordered according to their number of categories (variable with most categories is the rightmost variable) in the resulting object. |
| `caseorder` | logical with default set to `caseorder==TRUE`. When set to `caseorder==FALSE` configurations are only ordered according to the categories of the rightmost variable in the resulting object. |

| | |
|---|---|
| kat | ignored when x is a data.frame! If x is a "matrix" the optional argument kat must be an integer vector defining the number of categories for every variable in x (in the respective order). If left empty the (max) number of categories ist estimated from the data given in x. |
| codes | a list with character vectors containing coding for integers in matrix (if x is a numeric matrix). If codes is not empty (and the argument x is an object of class "matrix") the return object will be pattern frequencies table as data.frame. |

## Details

No further details

## Value

An object of class c("data.frame","Pfreq") containing the (response) pattern frequencies table representation of the given dataset in the argument x.

## References

No references in the moment

## Examples

```
#######################################
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 (Linert 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 (Linert LSD example).
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 (Linert LSD example).
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
#######################################
```

`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 = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object of class "matrix" which is a (response) pattern frequencies table. It is assumed, that the last column of the object `x` represents the frequencies of the (respose) patern represented by the other columns in `x`. |
| `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. |
| `...` | additional parameters passed trough. This is an option to assign factor labels to the resulting `data.frame` (when setting argument `fact=TRUE`) –> see `factor` in the `base` package and examples.<br>WARNING using this option will only work correct when all 'pattern' columns (variables) in the frequencies table share the same number of categories |

## Details

No details

## Value

An object of class "matrix" or "data.frame" (depending on the argument `fact`) containing the dataset representation of the (response) pattern frequencies table give in the argument `x`.

## References

No references in the moment

## Examples

```
########################################
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"))
```

## 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, `NA`s. |

### 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" )).

### 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.

### 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 (2014) taken from Linert (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. (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

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

---

LienertLSD *The Linert LSD Data*

---

## Description

Data from the classical Linert LSD trial as an example for CFA. 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 (response) pattern in column 1:3.

## Details

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

C = narrowed consciousness

T = thought disturbance

A = affective disturbance

## 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 2014*

### Description

data example by Stemmler (2014) p. 26 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. (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

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

## 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 (Linert LSD example).
pos_cfg_cfa(kat=c(2,2,2))
#######################################
```

---

## 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 subsamples.

### Usage

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

### Arguments

patternfreq
: an object of class "Pfreq", which is data in pattern frequencies representation - see function dat2fre. The variable defining the two subsaples (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
: a logical (TRUE / FALSE) determining wether to apply a continuity correction or not. When set to ccor=TRUE continuity correction is applied. For ccor=FALSE no continuity correction is applied.

...
: additional parameters passed through to other functions.

### Details

no details at the moment ...

### Value

an object of class S2CFA with results.

### 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.

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 Linert suicide Data*

---

### Description

Data from the Linert suicide example for CFA. The data is in data list representation (each row is one case).

### Usage

```
data(suicide)
```

### Format

A `data.frame` with 3 columns (as factors).

### 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.

### 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", ...)
```

### Arguments

| | |
|---|---|
| `object` | object of class`"CFA"` |
| `digits` | integer rounds the values to the specified number of decimal places, default is `digits=3`. |
| `type` | character with default `type="z.pChi"`, to return wether 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"`. |
| `...` | other parameters passed trough |

## 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", ...)
```

### Arguments

| | |
|---|---|
| object | object of class"S2CFA" |
| digits | integer rounds the values to the specified number of decimal places, default is digits=3. |
| type | character with default type="ex.fisher.test", to return wether the observed pattern are 'discriminating Types' or not significant at all based on the respective p-value. Another option for type is type="pChi". |
| ... | other parameters passed trough |

## 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. 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 an 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