Package ‘contigencyTable2’

December 10, 2022

Title Create Complete Tables to Show Statistics of Contigency Tables

Version 1.4

Description Visualization of contigency tables and calculate statistics of contigency table like exact test for 2x2 table and make beauty table with use html base package like kableExtra.

URL https://github.com/stats9/contigencyTable2

BugReports https://github.com/stats9/contigencyTable2/issues/1

License GPL (>=3)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports magrittr, epitools, kableExtra, htmltools, Hmisc, vcd, dplyr, DescTools, tibble, ggforce, ggplot2, rstatix

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.2

NeedsCompilation no

Author habib ezatabadi [aut, cre]

Maintainer habib ezatabadi <habibezati88@gmail.com>

R topics documented:

`check_package` *This function is prepared to check whether the package is installed by entering the name of a package as a string.*

Description

This function is prepared to check whether the package is installed by entering the name of a package as a string.

Usage

```
check_package(pak)
```

Arguments

pak	name of package as string format

Value

return a string ("this package is not installed") or a vector with two element, name and version of vector

Examples

```
## Not run:
    pak <- "ggplot2"
    check_package(pak)
## End(Not run)
```

`create_dat_two` *create a function to create original data from a table 2x2*

Description

create a function to create original data from a table 2x2

Usage

```
create_dat_two(tab, name1, name2)
```

Arguments

`tab`	contigency table 2x2
`name1`	A string that name of first variable into table
`name2`	A string that name of second variable into table

Value

res a dataframe that has two column which column 1 is first variable and column 2 is second variable

Examples

```
## Not run:
create_dat_two(mytable, "Expose", "Disease")

## End(Not run)
```

`draw_ellipse_ci` *Confidence ellipse for bivariate normal*

Description

To draw confidence ellipses for the mean of a two-variable normal distribution, there are functions in R, but none of them are theoretically accurate. Here, we have tried to draw this confidence ellipse in accordance with the academic texts.

Usage

```
draw_ellipse_ci(S, xbar, alpha = .05, n = 100)
```

Arguments

`S`	The Covariance Matrix
`xbar`	sample Mean
`alpha`	$1 - \alpha$ is level of CI.
`n`	number of observaions

Value

A plot that draw a ellipse with its diagonals and show sample Mean in center of that.

Author(s)

Habib Ezatabadi

References

Johnson, R. A., & Wichern, D. W. (1992). Applied multivariate statistical analysis. New Jersey, 405.

Examples

```
## Not run:
    S <- matrix(c(1, -1, -1, 4), 2, 2)
    xbar <- c(0, 0)
    draw_ellipse_ci(S = S, xbar = xbar, alpha = .05, n = 100)

## End(Not run)
```

---

get_contigency_result *Function to create a complete table results for contigency table*

Description

Function to create a complete table results for contigency table

Usage

```
get_contigency_result(n11, n12, n21, n22,
    varname1 = "Expose", varname2 = "Disease",
    levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"), show_table_results = TRUE)
```

Arguments

n11	The number that shows this is that the first variable of the table is at its first level and the second variable of the table is also at its first level
n12	The numbers that indicate this, the first variable of the table is on its first level and the second variable of the table is on its second level
n21	The numbers that indicate this, the first variable of the table is on its second level and the second variable of the table is on its first level
n22	The numbers that indicate this, the first variable of the table is on its second level and the second variable of the table is also on its second level
varname1	name of first variable
varname2	name of second variable
levels_var1	levels of first variable
levels_var2	levels of second variable

show_table_results
: A logical variable that takes two values, FALSE and TRUE, when in the TRUE state, is displayed in the output of a complete table as an HTML page.

Value

Table_results A list containing 8 output tables in html format, showing the outputs for each table.

stat_R_results list of 8 table as dataframe format for show result of table that generate from contigency table.

Examples

```
## Not run: get_contigency_result(
    n11 = 475, n12 = 461, n21 = 7, n22 = 61,
    varname1 = "Expose", varname2 = "Disease",
    levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"),
    show_table_results = TRUE)
## End(Not run)
```

`get_dat_from_tab` *This function is designed so that, according to the user's request, from an Contigency table based on two variables, a data set with type; Create a matrix or dataframe or list.*

Description

This function is designed so that, according to the user's request, from an Contigency table based on two variables, a data set with type; Create a matrix or dataframe or list.

Usage

```
get_dat_from_tab(tab, Levels = NULL , idLevel = 0, data_type = "Matrix",
    varnames = c("Var1", "Var2"))
```

Arguments

`tab`	contigency table based on Two Variables.
`Levels`	A list with two members, the first member of the variable levels that is distributed in the rows of the contigency table and the second Member in its columns, the default value is NULL. And level two and ... for two variables.
`idLevel`	indicator variable, if the `Levels` argument is entered, this argument must take the value 1, otherwise 0.
`data_type`	According to the user's request, if you want the format of the output data to be in the form of a matrix, the value of the `"Matrix"` is entered, for the dataframe, `"dataframe"` and for the list entered `"list"`.
`varnames`	A vector with two members, which are the names of the first variable (the variable whose levels are distributed in the rows of the contigency table) and the second.

Value

The output is a list with two members, input table (`original_table`) and dataset (`Data`).

Examples

```
## Not run: data(table_2)
    get_dat_from_tab(tab = table_2, data_type = "dataframe")
## End(Not run)
```

---

homogenity_test_or — *this function created for get mantel-haenszel and test homogenty of OR*

Description

this function created for get mantel-haenszel and test homogenty of OR

Usage

```
homogenity_test_or(x, partial_oddsratio_method = "wald",
    confront_var = "age")
```

Arguments

`x`	is array with Atleast 3 dimension
`partial_oddsratio_method`	method The odds ratio estimation method has three state `"midp"`,`"wald"`, `"exact"`
`confront_var`	confounding variable is A factor variable

Value

odd_ratio_result result

test_result resut results

tabe_test t table

Examples

```
## Not run: homogenity_test_or(x, partial_oddsratio_method = "wald", confront_var = "age")
```

`HypterTension` *HyperTension Data*

Description

A dataset with two variables and 200 observations.

- value: The variable in which blood pressure measurement values are stored in two groups of standard treatment and new treatment.
- Groups: The name of the treatment group

Usage

```
data(HyperTension)
```

Format

dataframe

`h_fisher` *This function is designed to implement Fisher's algorithm for exact testing in a 2x2 contigency table. Although the `stats::fisher.test()` function is a very fast and good function, this function is also suitable.*

Description

This function is designed to implement Fisher's algorithm for exact testing in a 2x2 contigency table. Although the `stats::fisher.test()` function is a very fast and good function, this function is also suitable.

Usage

```
h_fisher(tab, alternative = "two-sided")
```

Arguments

`tab`	contigency table $$2 \times 2$$
`alternative`	argumment that can take 3 value ("two-sided", "less", "greater")

Value

a vector with two element `"p-value"`, `"p-table"` that, `"p-value" is`

$$p_{value}$$

of test and `p-table` is probablity of original table.

Examples

```
## Not run: tab2 <- matrix(c(1, 9, 11, 3), 2, 2,
    byrow = T)
    h_fisher(tab2, alternative = "two-sided")
## End(Not run)
```

---

ifel *The base Function of R for applying a condition on a vector at the same time is in the form that return is the first value of the vector This function is designed to return a vector by applying a condition on a vector.*

Description

The base Function of R for applying a condition on a vector at the same time is in the form that return is the first value of the vector This function is designed to return a vector by applying a condition on a vector.

Usage

```
ifel(cond, x, y)
```

Arguments

`cond`	A logical value, that is TRUE or FALSE
`x`	if cond = `TRUE` return x
`y`	if cond = `FALSE` return y

See Also

`base::ifelse()`

Examples

```
## Not run: ifel(TRUE, c(1, 2, 5), c(4, 1, 3))
```

Inferiority_superiority_test_pa *Non-Inferiority and superiority Test in cilinical Trials, for compare two Treatment*

Description

In order to implement non-inferiority tests and superiority tests, in randomized clinical trials, when we want to compare two types of treatment or two types of drugs or a drug with a placebo, there are many softwares, there are various functions in R, some From the functions used in R, they do not have optimal outputs and even sometimes, their outputs are not very correct. In this function, we have tried to have integrated outputs with a completely simple table, as well as adding a graph, two non-derogatory tests. and superiority in studies designed for parallel groups.

Usage

```
Inferiority_superiority_test_pa(
    dataType  = "binary",  Dat, alpha = .05,
    Method_estimate_for_binary_data = "fm",
    margin, reff = 1, better = "right", Test_Method = "N",
    Name_groups = c(group1 = "standard", group2 = "new")
    )
```

Arguments

`dataType` It can take two values, `c("binary", "continuous")`, to tell the function whether our data is to compare rates or continuous values.

`Dat` is our data set, the thing about it is that the data must have two columns, the first column is related to values and the second column is related to factors, that is, there must be a specific factor in the second column, which determines that observation It is related to which treatment, the next point is that the agents can only be of two types. And also for binary data, the value column should have only `TRUE` (success) and `FALSE` (failure) values for each observation.

`alpha` $0 < \alpha < 1$, level of test.

`Method_estimate_for_binary_data` It can take 3 values, `c("fm", "ha", "wald")`, it should be noted that `"fm"` abbreviated of `"Farrington-Manning"` Method, `"ha"` abbreviated of `"Hauck-Anderson"` method, and `"wald"` implements the famous `"wald"` method for binary data.

`margin` $\delta,\ H_0 : p_1 - p_2 \leq \delta\ Vs\ H_1 : p_1 - p_2 > \delta$ Null-value for more information about margin in Non-Inferiority
and Superiority test go to `details`.

`reff` It takes two values `reff = 1`, `reff = 2`, 1 means that we want to subtract the average of the second community from the first and 2 means that we want to subtract the difference of the first community from the second. if `reff = 1` then $H_0 : \mu_1 - \mu_2 \leq \delta$ `reff = 2` then $H_0 : \mu_2 - \mu_1 \leq \delta$.

`better` It takes two values `c("left", "right")`. Maybe for certain tests, a lower value indicates a better treatment, in which case we should give this argument the `"left"` value, and if a higher value indicates a better treatment, by default, this argument contains the `"right"` value.

`Test_Method` getting two value `c("N", "S")` N infer to `"Non-Inferiority Test"` and S infer to `"Superiority Test"`.

`Name_groups` A Vector with Two element, name of group 1 and name of group 2. It is necessary that the values of this argument be quantified. By default, the values of this argument are `c("standard", "new")` standard for group 1 and new for group two.

Details

for Non-Inferiority and Superiority Test for rates and continuous values like Hypertension, We set a null value based on previous clinical studies or articles and studies already done, then:

Non-Inferiority If a treatment is better, i.e. that treatment has a larger mean in recorded observations or a higher rates, our test is defined as:

$$\text{for rates :} \quad \begin{cases} H_0 : & p_1 - p_2 \leq \delta \\ H_1 : & p_1 - p_2 > \delta, \end{cases} \quad \delta < 0.$$

$$\text{or for continuous values}: \quad \begin{cases} H_0: & \text{Treat}_1 - \text{Treat}_2 \leq \delta \\ H_1: & \text{Treat}_1 - \text{Treat}_2 > \delta \end{cases} \quad \delta < 0.$$

If a treatment is better, i.e. that treatment has a smaller mean in the recorded observations or a smaller rates, our test is defined as:

$$\text{for rates}: \quad \begin{cases} H_0: & p_1 - p_2 \geq \delta \\ H_1: & p_1 - p_2 < \delta, \end{cases} \quad \delta > 0.$$

$$\text{or for continuous values}: \quad \begin{cases} H_0: & \text{Treat}_1 - \text{Treat}_2 \geq \delta \\ H_1: & \text{Treat}_1 - \text{Treat}_2 < \delta \end{cases} \quad \delta > 0.$$

Superiority If a treatment is better, i.e. that treatment has a larger mean in recorded observations or a higher rates, our test is defined as:

$$\text{for rates}: \quad \begin{cases} H_0: & p_1 - p_2 \leq \delta \\ H_1: & p_1 - p_2 > \delta, \end{cases} \quad \delta > 0.$$

$$\text{or for continuous values}: \quad \begin{cases} H_0: & \text{Treat}_1 - \text{Treat}_2 \leq \delta \\ H_1: & \text{Treat}_1 - \text{Treat}_2 > \delta \end{cases} \quad \delta > 0.$$

If a treatment is better, i.e. that treatment has a smaller mean in the recorded observations or a smaller rates, our test is defined as:

$$\text{for rates}: \quad \begin{cases} H_0: & p_1 - p_2 \geq \delta \\ H_1: & p_1 - p_2 < \delta, \end{cases} \quad \delta < 0.$$

$$\text{or for continuous values}: \quad \begin{cases} H_0: & \text{Treat}_1 - \text{Treat}_2 \geq \delta \\ H_1: & \text{Treat}_1 - \text{Treat}_2 < \delta \end{cases} \quad \delta < 0.$$

Rates for rates we add three method in this code; `"wald"`, `"Farrington-Manning"` and `"Hauck-Anderson"` for get more information about this methods you can go to this link.

Continuous Values for continuous values we implement a `T Test`:

$$\bar{x}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} x_{i_1}, \quad \bar{x}_2 = \frac{1}{n_2} \sum_{i=1}^{n_2} x_{i_2}$$

$$\bar{x}_1 \sim \mathcal{N}(\mu_1, \frac{\sigma_1}{n_1}), \quad \bar{x}_2 \sim \mathcal{N}(\mu_2, \frac{\sigma_2}{n_2}),$$

$$\text{we want to test}: \quad \begin{cases} H_0: & \mu_1 - \mu_2 \leq \delta \\ H_1: & \mu_1 - \mu_2 > \delta \end{cases},$$

$$\text{if } \sigma_1 = \sigma_2 \implies S^2_{pooled} = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2},$$

$$S_1^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_{i_1} - \bar{x}_1)^2,$$

$$S_2^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (x_{i_2} - \bar{x}_2)^2, \implies$$

$$\text{Test Statistics}: \frac{\bar{x}_1 - \bar{x}_2 - \delta}{s_{pooled}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \underset{\text{if } H_0 \text{ is TRUE}}{\sim} T_{(n_1+n_2-2)}$$

$$\text{if } \sigma_1 \neq \sigma_2 \implies$$

We have to use Welch-Satterthwaite, so for more information about this statistics, go to this link.

Value

A List of Three Components

`TestResult`:	"htest" class that contains result of test
`plotResult`:	Plot of Test
`TestTable`:	A Html table for show results

TestResult that from class "htest" contains below members:\

`statistic`:	the value of the Z-statistic
`parameter`:	delta, rate difference (group 1 - group 2) under the null hypothesis
`p.value`:	the p-value for the Farrington-Manning test
`null.value`:	rate difference (group 1 - group 2) under the null
`alternative`:	a character string indicating the alternative hypothesis
`method`:	a character string indicating the exact method employed
`data.name`:	a character string giving the names of the data used
`estimate`:	the estimated rate difference (maximum likelihood)
`conf.int`:	a confidence interval for the rate difference
`sample.size`:	the total sample size used for the test

Author(s)

Habib Ezatabadi

Examples

```
## Not run:
    dat <- data(HyperTension)
    Inferiority_superiority_test(dataType = "continuous", Dat = dat, alpha = 0.05,
    margin = 5, reff = 1, better = "right", Test_Method = "N",
    Name_groups  = c(group1 = "standard", group2 = "new"))

## End(Not run)
```

---

lambda_coef_contigency *define function for get Lambda coefficients*

Description

define function for get Lambda coefficients

Usage

```
lambda_coef_contigency(n11, n12, n21, n22,
    varname1 = "Expose", varname2 = "Diseasee", levels_var1 = c("Exposed",
    "UnExposed"), levels_var2 = c("Disease", "UnDisease"))
```

Arguments

`n11`	see also `get_contigency_result`
`n12`	see also `get_contigency_result`
`n21`	see also `get_contigency_result`
`n22`	see also `get_contigency_result`
`varname1`	see also `get_contigency_result`
`varname2`	see also `get_contigency_result`
`levels_var1`	see also `get_contigency_result`
`levels_var2`	see also `get_contigency_result`

Value

table of lambda result, for more detail of what is lambda

Examples

```
## Not run: lambda_coef_contigency(475, 461, 7, 61, "Expose", "Disease",
    levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"))
## End(Not run)
```

`list_to_dataframe` *this function created for convert a list to dataframe, List members must be vectors with equal number of members.*

Description

this function created for convert a list to dataframe, List members must be vectors with equal number of members.

Usage

```
list_to_dataframe(List)
```

Arguments

`List`	a list; List members must be vectors with equal number of members

Value

a dataframe, A dataframe whose columns are members of the input list.

Examples

```
## Not run:
    List = list(a = c(1, 2, 3, 4), b = c("a", "b", "c", "d"))
    list_to_dataframe(List)
## End(Not run)
```

odr — *for get oddsRatio based on Column 1, Column 2 from a contigency table*

Description

for get oddsRatio based on Column 1, Column 2 from a contigency table

Usage

```
odr(n11, n12, n21, n22,
     varname1 = "Expose", varname2 = "Disease",
   levels_var1 = c("Exposed", "UnExposed"), levels_var2 = c("Disease", "UnDisease"),
     method = "wald", conf_level = 0.95, show_table_result = TRUE)
```

Arguments

n11	see get_contigency_result
n12	see get_contigency_result
n21	see get_contigency_result
n22	see get_contigency_result
varname1	see get_contigency_result
varname2	see get_contigency_result
levels_var1	see get_contigency_result
levels_var2	see get_contigency_result
method	The odds ratio estimation method has three state "midp", "wald", "exact"
conf_level	level of confidence Interval
show_table_result	see also get_contigency_result

Value

two table of oddsratio results, a html table and a r table

Examples

```
## Not run:
    odr(n11 = 475, n12 = 461, n21 = 7, n22 = 61, varname1 = "Expose",
    varname2 = "Disease", levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"),
    method = "wald", conf_level = 0.95, show_table_result = TRUE)

## End(Not run)
```

rct_binary_data_1 *rct_binary_data_1 Data*

Description

```
A dataset with two variables and 114 observations.
```

- values: Recovery or non-recovery for the treatment group.
- group: The name of the treatment group

Usage

```
data(rct_binary_data_1)
```

Format

dataframe

rct_binary_data_2 *rct_binary_data_2 Data*

Description

```
A Table with 3 variables.
```

- drug: A type of drug
- alive: alive = 1, death = 0
- count number of patients that alive or death.

Usage

```
data(rct_binary_data_2)
```

Format

dataframe

rct_binary_data_3 *rct_binary_data_3 Data*

Description

A Table with 3 variables.

- method_treat: A type of Treatment
- state_life: alive = 1, death = 0
- count number of patients that alive or death.

Usage

```
data(rct_binary_data_3)
```

Format

dataframe

rct_continuous_data_2 *rct_continuous_data_2 Data*

Description

A dataset with two variables and 21 observations.

- values: Measured criteria for each patient's recovery
- treat: The Type of Treatment

Usage

```
data(rct_continuous_data_2)
```

Format

dataframe

rct_continuous_data_3 *rct_continuous_data_3 Data*

Description

A dataset with two variables and 22 observations.

- values: Measured criteria for each patient's recovery
- treat: The Type of Treatment

Usage

```
data(rct_continuous_data_3)
```

Format

dataframe

rr *define function for get relative risk results*

Description

define function for get relative risk results

Usage

```
rr(n11, n12, n21, n22,
    varname1 = "Expose", varname2 = "Diseasee", levels_var1 = c("Exposed",
    "UnExposed"), levels_var2 = c("Disease", "UnDisease"),
    method = "wald", conf_level = 0.95, nboot = 1000)
```

Arguments

n11	see also get_contigency_result
n12	see also get_contigency_result
n21	see also get_contigency_result
n22	see also get_contigency_result
varname1	see also get_contigency_result
varname2	see also get_contigency_result
levels_var1	see also get_contigency_result
levels_var2	see also get_contigency_result
method	It has two modes: "wald", "boot" which is the "boot" mode based on resampling Method.
conf_level	see odr
nboot	when method = "boot" therefore nboot is number of replicates that make resampling. resamplingMethods.

Value

two table for RiskRatio results.

Examples

```
## Not run: rr(475, 461, 7, 61, "Expose", "Disease", c("Exposed", "UnExposed"),
    c("Disease", "UnDisease"), method = "boot", conf_level = 0.95, nboot = 1000)
## End(Not run)
```

---

stats_round — *Rounding vectors that have multi-type values (characters and numbers)*

Description

To round a vector that has both numeric values and character values, we know that if a vector contains characters, all the values have character format, but sometimes we need to round the numeric values to several decimal places when displaying the vector. This function is implemented to round numerical values in vectors that contain characters.

Usage

```
stats_round(x, ndigit = 4)
```

Arguments

`x`	A vector, with numeric and character elements
`ndigit`	The number of decimal digits we want to round the numbers inside the vector

Value

A vector of the same size but with numbers rounded to an arbitrary number of decimal places.

Author(s)

Habib Ezatabadi

See Also

`base::round()`

Examples

```
## Not run: stats_round(x = c("a", 2.342341, "stats9", 3.324234235), ndigit = 2)
```

table_1 — *table_1 contigency table with 3 variables*

Description

A dataset containing a contigency table with 3 variable The variables are as follows:

Usage

```
data(table_1)
```

Format

contigency table with 3 variables

Details

- exposure: The variable that shows how many were exposed, which is a binary variable with two levels of exposure (1) or no exposure (0).
- Group: A binary variable that is leveled at the level of the treated group (1) and the control group (0).
- age: A categorical variable, which is divided into three levels: 1, 2, and 3.

table_2 — *table_2 contigency table with 2 variables*

Description

A contigency table based on the number of case-control study for ovarian cancer patients and its association with contraceptive use and duration of use.

Usage

```
data(table_2)
```

Format

contigency table with 2 variables

Details

- Disease: The variable that shows how many were Disease (case) on Not Disease (control), which is a binary variable with two levels of Disease (case) or Not Disease (control).
- OC Duration time: How long the person in question has been using contraceptives. which has 4 levels, no use (`None`), between 0 and 5 years of use (`0-5`), between 5 and 10 years of use (`50-10` and more than 10 years of use (`>10`).

`Table_Test_Result` *This function has been prepared for the purpose of performing three valid tests to check the connection or non-connection of the columns and rows of a contigency table and to output the test statistics as well as the expected values of the table and to check whether the exact test should also be performed or not. bring.*

Description

This function has been prepared for the purpose of performing three valid tests to check the connection or non-connection of the columns and rows of a contigency table and to output the test statistics as well as the expected values of the table and to check whether the exact test should also be performed or not. bring.

Usage

```
Table_Test_Result(tab, Levels, idLevel = 0)
```

Arguments

`tab`	contigency table with two variable, that any variable have I (I >= 2) levels.
`Levels`	see `get_dat_from_tab`
`idLevel`	see `get_dat_from_tab`

Details

for calculate test statistics values, we use this formulas:

$$\text{Contigency Table} = \left[\begin{array}{c|c|c|c} n_{(1,\ 1)} & n_{(1,\ 2)} & \cdots & n_{(1,\ J)} \\ n_{(2,\ 1)} & n_{(2,\ 2)} & \cdots & n_{(2,\ J)} \\ \vdots & \ddots & \ddots & \vdots \\ n_{(I,\ 1)} & n_{(2,\ 2)} & \cdots & n_{(I,\ J)} \end{array}\right]$$

$$\Lambda = \frac{\prod_i \prod_j (n_{i+} \times n_{+j})^{n_{ij}}}{n \prod_i \prod_j n_{ij}^{n_{ij}}}$$

$$G^2 = -2\log(\Lambda) = 2\sum_i \sum_j n_{ij} \log\left(\frac{n_{ij}}{\hat{\mu}_{ij}}\right)$$

$$\hat{\mu}_{ij} = \frac{n_{i+} \times n_{+j}}{n}$$

$$\underset{\text{If } H_0 \text{ is TRUE}}{G^2} \approx \chi^2_{(I-1)\times(J-1)}$$

$$\chi^2_{\text{pearson}} = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(n_{(i,\ j)} - \hat{\lambda}_{(i,\ j))})^2}{\hat{\lambda}_{(i,\ j))}}$$

$$\underset{\text{If } H_0 \text{ is TRUE}}{\chi^2_{(\text{pearson})}} \approx \chi^2_{(I-1)\times(J-1)}$$

$$\text{Trend Test Statistics} = M^2 = r^2 \times (n-1)$$

$$\underset{\text{If } H_0 \text{ Is TRUE}}{M^2} \approx \chi^2_{(1)}$$

$$n = \sum_{i=1}^{I} \sum_{j=1}^{J} n_{(i,\ j)},$$

$$r = \text{Corr}(X_1,\ X_2), \quad X_1,\ X_2 \text{ Are two variables of contigency table}$$

Value

ExpEcted_Vals table of expected values of a contigency table (tab)

test_result table of test results

Total_results table of Total results (expected values, test resutls and input table)

table_results html table for total results

Examples

```
## Not run: data(table_2)
    Table_Test_Result(tab = table_2)
## End(Not run)
```

`uncertainty_get` *Uncertainty coefficient function*

Description

Uncertainty coefficient function

Usage

```
uncertainty_get(n11, n12, n21, n22,
    varname1 = "Expose", varname2 = "Disease",
    levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"))
```

Arguments

`n11`	see also `get_contigency_result`
`n12`	see also `get_contigency_result`
`n21`	see also `get_contigency_result`
`n22`	see also `get_contigency_result`
`varname1`	see also `get_contigency_result`
`varname2`	see also `get_contigency_result`
`levels_var1`	see also `get_contigency_result`
`levels_var2`	see also `get_contigency_result`

Value

table of uncertainty coefficienty results

Examples

```
## Not run: uncertainty_get(n11 = 475, n12 = 461,
    n21 = 7, n22 = 61, varname1 = "Expose",
    varname2 = "Disease",
    levels_var1 = c("Exposed", "UnExposed"),
    levels_var2 = c("Disease", "UnDisease"))
## End(Not run)
```

Index