Search all packages and functions

stats (version 3.6.2)

chisq.test: Pearson's Chi-squared Test for Count Data

Description

`chisq.test` performs chi-squared contingency table tests and goodness-of-fit tests.

Usage

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

Arguments

x

a numeric vector or matrix. `x` and `y` can also both be factors.

y

a numeric vector; ignored if `x` is a matrix. If `x` is a factor, `y` should be a factor of the same length.

correct

a logical indicating whether to apply continuity correction when computing the test statistic for 2 by 2 tables: one half is subtracted from all $|O - E|$ differences; however, the correction will not be bigger than the differences themselves. No correction is done if `simulate.p.value = TRUE`.

p

a vector of probabilities of the same length of `x`. An error is given if any entry of `p` is negative.

rescale.p

a logical scalar; if TRUE then `p` is rescaled (if necessary) to sum to 1. If `rescale.p` is FALSE, and `p` does not sum to 1, an error is given.

simulate.p.value

a logical indicating whether to compute p-values by Monte Carlo simulation.

B

an integer specifying the number of replicates used in the Monte Carlo test.

Value

A list with class `"htest"` containing the following components:

`statistic`

the value the chi-squared test statistic.

`parameter`

the degrees of freedom of the approximate chi-squared distribution of the test statistic, `NA` if the p-value is computed by Monte Carlo simulation.

`p.value`

the p-value for the test.

`method`

a character string indicating the type of test performed, and whether Monte Carlo simulation or continuity correction was used.

`data.name`

a character string giving the name(s) of the data.

`observed`

the observed counts.

`expected`

the expected counts under the null hypothesis.

`residuals`

the Pearson residuals, `(observed - expected) / sqrt(expected)`.

`stdres`

standardized residuals, `(observed - expected) / sqrt(V)`, where `V` is the residual cell variance (Agresti, 2007, section 2.4.5 for the case where `x` is a matrix, `n * p * (1 - p)` otherwise).

Details

If `x` is a matrix with one row or column, or if `x` is a vector and `y` is not given, then a *goodness-of-fit test* is performed (`x` is treated as a one-dimensional contingency table). The entries of `x` must be non-negative integers. In this case, the hypothesis tested is whether the population probabilities equal those in `p`, or are all equal if `p` is not given.

If `x` is a matrix with at least two rows and columns, it is taken as a two-dimensional contingency table: the entries of `x` must be non-negative integers. Otherwise, `x` and `y` must be vectors or factors of the same length; cases with missing values are removed, the objects are coerced to factors, and the contingency table is computed from these.

Then Pearson's chi-squared test is performed of the null hypothesis that the joint distribution of the cell counts in a 2-dimensional contingency table is the product of the row and column marginals.

If `simulate.p.value` is `FALSE`, the p-value is computed from the asymptotic chi-squared distribution of the test statistic; continuity correction is only used in the 2-by-2 case (if `correct` is `TRUE`, the default). Otherwise the p-value is computed for a Monte Carlo test (Hope, 1968) with `B` replicates.

In the contingency table case simulation is done by random sampling from the set of all contingency tables with given marginals, and works only if the marginals are strictly positive. Continuity correction is never used, and the statistic is quoted without it. Note that this is not the usual sampling situation assumed for the chi-squared test but rather that for Fisher's exact test.

In the goodness-of-fit case simulation is done by random sampling from the discrete distribution specified by `p`, each sample being of size `n = sum(x)`. This simulation is done in R and may be slow.

References

Hope, A. C. A. (1968). A simplified Monte Carlo significance test procedure. *Journal of the Royal Statistical Society Series B*, **30**, 582--598. http://www.jstor.org/stable/2984263.

Patefield, W. M. (1981). Algorithm AS 159: An efficient method of generating r x c tables with given row and column totals. *Applied Statistics*, **30**, 91--97. 10.2307/2346669.

Agresti, A. (2007). *An Introduction to Categorical Data Analysis*, 2nd ed. New York: John Wiley & Sons. Page 38.

See Also

For goodness-of-fit testing, notably of continuous distributions, `ks.test`.

Examples

Run this code

```
# NOT RUN {
## From Agresti(2007) p.39
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(gender = c("F", "M"),
                    party = c("Democrat","Independent", "Republican"))
(Xsq <- chisq.test(M))  # Prints test summary
Xsq$observed   # observed counts (same as M)
Xsq$expected   # expected counts under the null
Xsq$residuals  # Pearson residuals
Xsq$stdres     # standardized residuals


## Effect of simulating p-values
x <- matrix(c(12, 5, 7, 7), ncol = 2)
chisq.test(x)$p.value           # 0.4233
chisq.test(x, simulate.p.value = TRUE, B = 10000)$p.value
                                # around 0.29!

## Testing for population probabilities
```

```r
## Testing for population probabilities
## Case A. Tabulated data
x <- c(A = 20, B = 15, C = 25)
chisq.test(x)
chisq.test(as.table(x))             # the same

x <- c(89,37,30,28,2)
p <- c(40,20,20,15,5)
try(
chisq.test(x, p = p)                # gives an error
)
chisq.test(x, p = p, rescale.p = TRUE)
                                    # works
p <- c(0.40,0.20,0.20,0.19,0.01)
                                    # Expected count in category 5
                                    # is 1.86 < 5 ==> chi square approx.
chisq.test(x, p = p)                #               maybe doubtful, but is ok!
chisq.test(x, p = p, simulate.p.value = TRUE)

## Case B. Raw data
x <- trunc(5 * runif(100))
chisq.test(table(x))            # NOT 'chisq.test(x)'!
# }
```

Run the code above in your browser using DataCamp Workspace

Powered By DataCamp