Package 'DiscreteTests'

January 20, 2025

Type Package

Title Vectorised Computation of P-Values and Their Supports for Several Discrete Statistical Tests

Version 0.2.1

Date 2024-10-27

Description Provides vectorised functions for computing p-values of various common discrete statistical tests, as described e.g. in Agresti (2002) <doi:10.1002/0471249688>, including their distributions. Exact and approximate computation methods are provided. For exact p-values, several procedures of determining two-sided p-values are included, which are outlined in more detail in Hirji (2006) <doi:10.1201/9781420036190>.

License GPL-3

Encoding UTF-8

Language en-GB

Imports R6, checkmate, lifecycle

URL https://github.com/DISOhda/DiscreteTests

BugReports https://github.com/DISOhda/DiscreteTests/issues

RoxygenNote 7.3.2

NeedsCompilation no

Author Florian Junge [cre, aut] (<https://orcid.org/0009-0001-6856-6938>), Christina Kihn [aut], Sebastian Döhler [ctb] (<https://orcid.org/0000-0002-0321-6355>)

Maintainer Florian Junge <diso.fbmn@h-da.de>

Repository CRAN

Date/Publication 2024-10-27 17:40:02 UTC

Contents

`DiscreteTests-package` *Vectorised Computation of P-Values and Their Supports for Several Discrete Statistical Tests*

Description

This package provides vectorised functions for computing p-values of various discrete statistical tests. Exact and approximate computation methods are provided. For exact p-values, several procedures of determining two-sided p-values are included.

Additionally, these functions are capable of returning the discrete p-value supports, i.e. all observable p-values under a null hypothesis. These supports can be used for multiple testing procedures in the `DiscreteFDR` and `FDX` packages.

Author(s)

Maintainer: Florian Junge <diso.fbmn@h-da.de> (ORCID)

Authors:

- Christina Kihn

Other contributors:

- Sebastian Döhler <sebastian.doehler@h-da.de> (ORCID) [contributor]

References

Fisher, R. A. (1935). The logic of inductive inference. *Journal of the Royal Statistical Society Series A*, **98**, pp. 39–54. doi:10.2307/2342435

Agresti, A. (2002). *Categorical data analysis* (2nd ed.). New York: John Wiley & Sons. doi:10.1002/0471249688

Blaker, H. (2000) Confidence curves and improved exact confidence intervals for discrete distributions. *Canadian Journal of Statistics*, **28**(4), pp. 783-798. doi:10.2307/3315916

Hirji, K. F. (2006). *Exact analysis of discrete data*. New York: Chapman and Hall/CRC. pp. 55-83. doi:10.1201/9781420036190

See Also

Useful links:

- https://github.com/DISOhda/DiscreteTests
- Report bugs at https://github.com/DISOhda/DiscreteTests/issues

---

`binom_test_pv` *Binomial Tests*

Description

`binom_test_pv()` performs an exact or approximate binomial test about the probability of success in a Bernoulli experiment. In contrast to `stats::binom.test()`, it is vectorised, only calculates p-values and offers a normal approximation of their computation. Furthermore, it is capable of returning the discrete p-value supports, i.e. all observable p-values under a null hypothesis. Multiple tests can be evaluated simultaneously. In two-sided tests, several procedures of obtaining the respective p-values are implemented.

[Deprecated]
Note: Please use `binom_test_pv()`! The older `binom.test.pv()` is deprecated in order to migrate to snake case. It will be removed in a future version.

Usage

```
binom_test_pv(
  x,
  n,
  p = 0.5,
  alternative = "two.sided",
  ts_method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple_output = FALSE
)

binom.test.pv(
  x,
  n,
  p = 0.5,
  alternative = "two.sided",
  ts.method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple.output = FALSE
)
```

Arguments

x	integer vector giving the number of successes.
n	integer vector giving the number of trials.
p	numerical vector of hypothesised probabilities of success.
alternative	character vector that indicates the alternative hypotheses; each value must be one of "two.sided" (the default), "less" or "greater".
ts_method, ts.method	single character string that indicates the two-sided p-value computation method (if any value in alternative equals "two.sided") and must be one of "minlike" (the default), "blaker", "absdist" or "central" (see details). Ignored, if exact = FALSE.
exact	logical value that indicates whether p-values are to be calculated by exact computation (exact = TRUE; the default) or by a continuous approximation.
correct	logical value that indicates if a continuity correction is to be applied (correct = TRUE; the default) or not. Ignored, if exact = TRUE.
simple_output, simple.output	logical value that indicates whether an R6 class object, including the tests' parameters and support sets, i.e. all observable p-values under each null hypothesis, is to be returned (see below).

Details

The parameters x, n, p and alternative are vectorised. They are replicated automatically to have the same lengths. This allows multiple hypotheses to be tested simultaneously.

If p = NULL, it is tested if the probability of success is 0.5 with the alternative being specified by alternative.

For exact computation, various procedures of determining two-sided p-values are implemented.

"minlike" The standard approach in stats::fisher.test() and stats::binom.test(). The probabilities of the likelihoods that are equal or less than the observed one are summed up. In Hirji (2006), it is referred to as the *Probability-based* approach.

"blaker" The minima of the observations' lower and upper tail probabilities are combined with the opposite tail not greater than these minima. More details can be found in Blaker (2000) or Hirji (2006), where it is referred to as the *Combined Tails* method.

"absdist" The probabilities of the absolute distances from the expected value that are greater than or equal to the observed one are summed up. In Hirji (2006), it is referred to as the *Distance from Center* approach.

"central" The smaller values of the observations' simply doubles the minimum of lower and upper tail probabilities. In Hirji (2006), it is referred to as the *Twice the Smaller Tail* method.

For non-exact (i.e. continuous approximation) approaches, ts_method is ignored, since all its methods would yield the same p-values. More specifically, they all converge to the doubling approach as in ts_mthod = "central".

Value

If `simple.output = TRUE`, a vector of computed p-values is returned. Otherwise, the output is a `DiscreteTestResults` R6 class object, which also includes the p-value supports and testing parameters. These have to be accessed by public methods, e.g. `$get_pvalues()`.

References

Agresti, A. (2002). *Categorical data analysis* (2nd ed.). New York: John Wiley & Sons. pp. 14-15. doi:10.1002/0471249688

Blaker, H. (2000) Confidence curves and improved exact confidence intervals for discrete distributions. *Canadian Journal of Statistics*, **28**(4), pp. 783-798. doi:10.2307/3315916

Hirji, K. F. (2006). *Exact analysis of discrete data*. New York: Chapman and Hall/CRC. pp. 55-83. doi:10.1201/9781420036190

See Also

`stats::binom.test()`

Examples

```
# Constructing
k <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
n <- c(18, 12, 10)
p <- c(0.5, 0.2, 0.3)

# Computation of exact two-sided p-values ("blaker") and their supports
results_ex  <- binom_test_pv(k, n, p, ts_method = "blaker")
raw_pvalues <- results_ex$get_pvalues()
pCDFlist    <- results_ex$get_pvalue_supports()

# Computation of normal-approximated one-sided p-values ("less") and their supports
results_ap  <- binom_test_pv(k, n, p, "less", exact = FALSE)
raw_pvalues <- results_ap$get_pvalues()
pCDFlist    <- results_ap$get_pvalue_supports()
```

DiscreteTestResults *Discrete Test Results Class*

Description

This is the class used by the statistical test functions of this package for returning not only p-values, but also the supports of their distributions and the parameters of the respective tests. Objects of this class are obtained by setting the `simple.output` parameter of a test function to `FALSE` (the default). All data members of this class are private to avoid inconsistencies by deliberate or inadvertent changes by the user. However, the results can be read by public methods.

Methods

Public methods:

- DiscreteTestResults$new()
- DiscreteTestResults$get_pvalues()
- DiscreteTestResults$get_inputs()
- DiscreteTestResults$get_pvalue_supports()
- DiscreteTestResults$get_support_indices()
- DiscreteTestResults$print()
- DiscreteTestResults$clone()

Method `new()`**:** Creates a new `DiscreteTestResults` object.

Usage:

```
DiscreteTestResults$new(
  test_name,
  inputs,
  p_values,
  pvalue_supports,
  support_indices,
  data_name
)
```

Arguments:

`test_name` single character string with the name of the test(s).

`inputs` named list of **exactly three** elements containing the observations, test parameters and hypothesised null values **as data frames**; names of these list fields must be `observations`, `nullvalues` and `parameters`. See details for further information about the requirements for these fields.

`p_values` numeric vector of the p-values calculated by each hypothesis test.

`pvalue_supports` list of **unique** numeric vectors containing all p-values that are observable under the respective hypothesis; each value of `p_values` must occur in its respective p-value support.

`support_indices` list of numeric vectors containing the test indices that indicates to which individual testing scenario each unique parameter set and each unique support belongs.

`data_name` single character string with the name of the variable that contains the observed data.

Details: The fields of the `inputs` have the following requirements:

`$observations` data frame that contains the observed data; if the observed data is a matrix, it must be converted to a data frame; must not be `NULL`, only numerical and character values are allowed.

`$nullvalues` data frame that contains the hypothesised values of the tests, e.g. the rate parameters for Poisson tests; must not be `NULL`, only numerical values are allowed.

`$parameters` data frame that holds the parameter combinations of the null distribution of each test (e.g. numbers of Bernoulli trials for binomial tests, or `m`, `n` and `k` for the hypergeometric distribution used by Fisher's Exact Test, which have to be derived from the observations first); **must** include a mandatory column named `alternative`; only numerical and character values are allowed.

Missing values or NULLs are not allowed for any of these fields. All data frames must have the same number of rows. Their column names are used by the print method for producing text output, therefore they should be informative, i.e. short and (if necessary) non-syntactic, like e.g. `number of success`.

Method get_pvalues(): Returns the computed p-values.

Usage:

```
DiscreteTestResults$get_pvalues(named = TRUE)
```

Arguments:

named single logical value that indicates whether the vector is to be returned as a named vector (if names are present)

Returns: A numeric vector of the p-values of all null hypotheses.

Method get_inputs(): Return the list of the test inputs.

Usage:

```
DiscreteTestResults$get_inputs(unique = FALSE)
```

Arguments:

unique single logical value that indicates whether only unique combinations of parameter sets and null values are to be returned. If unique = FALSE (the default), the returned data frames may contain duplicate sets.

Returns: A list of three elements. The first one contains a data frame with the observations for each tested null hypothesis, while the second is another data frame with the hypothesised null values (e.g. p for binomial tests). The third list field holds the parameter sets (e.g. n in case of a binomial test). If unique = TRUE, only unique combinations of parameter sets and null values are returned, but observations remain unchanged.

Method get_pvalue_supports(): Returns the p-value supports, i.e. all observable p-values under the respective null hypothesis of each test.

Usage:

```
DiscreteTestResults$get_pvalue_supports(unique = FALSE)
```

Arguments:

unique single logical value that indicates whether only unique p-value supports are to be returned. If unique = FALSE (the default), the returned supports may be duplicated.

Returns: A list of numeric vectors containing the supports of the p-value null distributions.

Method get_support_indices(): Returns the indices that indicate to which testing scenario each unique support belongs.

Usage:

```
DiscreteTestResults$get_support_indices()
```

Returns: A list of numeric vectors. Each one contains the indices of the null hypotheses to which the respective support and/or unique parameter set belongs.

Method print(): Prints the computed p-values.

Usage:

```
DiscreteTestResults$print(
  inputs = TRUE,
  pvalues = TRUE,
  supports = FALSE,
  test_idx = NULL,
  limit = 10,
  ...
)
```

Arguments:

`inputs` single logical value that indicates if the inputs values (i.e. observations and parameters) are to be printed; defaults to `TRUE`.

`pvalues` single logical value that indicates if the resulting p-values are to be printed; defaults to `TRUE`.

`supports` single logical value that indicates if the p-value supports are to be printed; defaults to `FALSE`.

`test_idx` integer vector giving the indices of the tests whose results are to be printed; if `NULL` (the default), results of every test up to the index specified by `limit` (see below) are printed

`limit` single integer that indicates the maximum number of test results to be printed; if `limit = 0`, results of every test are printed; ignored if `test_idx` is not set to `NULL`

`...` further arguments passed to `print.default`.

Returns: Prints a summary of the tested null hypotheses. The object itself is invisibly returned.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DiscreteTestResults$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## one-sided binomial test
#  parameters
x <- 2:4
n <- 5
p <- 0.4
m <- length(x)
#  support (same for all three tests) and p-values
support <- sapply(0:n, function(k) binom.test(k, n, p, "greater")$p.value)
pv <- support[x + 1]
#  DiscreteTestResults object
res <- DiscreteTestResults$new(
  # string with name of the test
  test_name = "Exact binomial test",
  # list of data frames
  inputs = list(
    observations = data.frame(
      `number of successes` = x,
      # no name check of column header to have a speaking name for 'print'
```

```
      check.names = FALSE
    ),
    parameters = data.frame(
      # parameter 'n', needs to be replicated to length of 'x'
      `number of trials` = rep(n, m),
      # mandatory parameter 'alternative', needs to be replicated to length of 'x'
      alternative = rep("greater", m),
      # no name check of column header to have a speaking name for 'print'
      check.names = FALSE
    ),
    nullvalues = data.frame(
      # here: only one null value, 'p'; needs to be replicated to length of 'x'
      `probability of success` = rep(p, m),
      # no name check of column header to have a speaking name for 'print'
      check.names = FALSE
    )
  ),
  # numerical vector of p-values
  p_values = pv,
  # list of supports (here: only one support); values must be sorted and unique
  pvalue_supports = list(unique(sort(support))),
  # list of indices that indicate which p-value/hypothesis each support belongs to
  support_indices = list(1:m),
  # name of input data variables
  data_name = "x, n and p"
)

#  print results without supports
print(res)
#  print results with supports
print(res, supports = TRUE)
```

`DiscreteTestResultsSummary`

Discrete Test Results Summary Class

Description

This is the class used by `DiscreteTests` for summarising `DiscreteTestResults` objects. It contains the summarised objects itself, as well as a summary data frame as private members. Both can be read by public methods.

Methods

Public methods:

- `DiscreteTestResultsSummary$new()`
- `DiscreteTestResultsSummary$get_test_results()`
- `DiscreteTestResultsSummary$get_summary_table()`

- DiscreteTestResultsSummary$print()
- DiscreteTestResultsSummary$clone()

Method `new()`**:** Creates a new `summary.DiscreteTestResults` object.

Usage:

```
DiscreteTestResultsSummary$new(test_results)
```

Arguments:

`test_results` the `DiscreteTestResults` class object to be summarised.

Method `get_test_results()`**:** Returns the underlying DiscreteTestResults object.

Usage:

```
DiscreteTestResultsSummary$get_test_results()
```

Returns: A DiscreteTestResults R6 class object.

Method `get_summary_table()`**:** Returns the summary table of the underlying DiscreteTestResults object.

Usage:

```
DiscreteTestResultsSummary$get_summary_table()
```

Returns: A data frame.

Method `print()`**:** Prints the summary.

Usage:

```
DiscreteTestResultsSummary$print(...)
```

Arguments:

`...` further arguments passed to `print.data.frame`.

Returns: Prints a summary table of the tested null hypotheses. The object itself is invisibly returned.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DiscreteTestResultsSummary$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# binomial tests
obj <- binom.test.pv(0:5, 5, 0.5)
# create DiscreteTestResultsSummary object
res <- DiscreteTestResultsSummary$new(obj)
# print summary
print(res)
# extract summary table
res$get_summary_table()
```

`fisher_test_pv` *Fisher's Exact Test for Count Data*

Description

`fisher_test_pv()` performs Fisher's exact test or a chi-square approximation to assess if rows and columns of a 2-by-2 contingency table with fixed marginals are independent. In contrast to `stats::fisher.test()`, it is vectorised, only calculates p-values and offers a normal approximation of their computation. Furthermore, it is capable of returning the discrete p-value supports, i.e. all observable p-values under a null hypothesis. Multiple tables can be analysed simultaneously. In two-sided tests, several procedures of obtaining the respective p-values are implemented.

[Deprecated]
Note: Please use `fisher_test_pv()`! The older `fisher.test.pv()` is deprecated in order to migrate to snake case. It will be removed in a future version.

Usage

```
fisher_test_pv(
  x,
  alternative = "two.sided",
  ts_method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple_output = FALSE
)

fisher.test.pv(
  x,
  alternative = "two.sided",
  ts.method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple.output = FALSE
)
```

Arguments

`x`	integer vector with four elements, a 2-by-2 matrix or an integer matrix (or data frame) with four columns, where each line represents a 2-by-2 table to be tested.
`alternative`	character vector that indicates the alternative hypotheses; each value must be one of `"two.sided"` (the default), `"less"` or `"greater"`.
`ts_method`, `ts.method`	single character string that indicates the two-sided p-value computation method (if any value in `alternative` equals `"two.sided"`) and must be one of `"minlike"` (the default), `"blaker"`, `"absdist"` or `"central"` (see details). Ignored, if `exact = FALSE`.

`exact` logical value that indicates whether p-values are to be calculated by exact computation (`exact = TRUE`; the default) or by a continuous approximation.

`correct` logical value that indicates if a continuity correction is to be applied (`correct = TRUE`; the default) or not. Ignored, if `exact = TRUE`.

`simple_output, simple.output` logical value that indicates whether an R6 class object, including the tests' parameters and support sets, i.e. all observable p-values under each null hypothesis, is to be returned (see below).

Details

The parameters `x` and `alternative` are vectorised. They are replicated automatically, such that the number of `x`'s rows is the same as the length of `alternative`. This allows multiple null hypotheses to be tested simultaneously. Since `x` is (if necessary) coerced to a matrix with four columns, it is replicated row-wise.

If `exact = TRUE`, Fisher's exact test is performed (the specific hypothesis depends on the value of `alternative`). Otherwise, if `exact = FALSE`, a chi-square approximation is used for two-sided hypotheses or a normal approximation for one-sided tests, based on the square root of the chi-squared statistic. This is possible because the degrees of freedom of chi-squared tests on 2-by-2 tables are limited to 1.

For exact computation, various procedures of determining two-sided p-values are implemented.

`"minlike"` The standard approach in `stats::fisher.test()` and `stats::binom.test()`. The probabilities of the likelihoods that are equal or less than the observed one are summed up. In Hirji (2006), it is referred to as the *Probability-based* approach.

`"blaker"` The minima of the observations' lower and upper tail probabilities are combined with the opposite tail not greater than these minima. More details can be found in Blaker (2000) or Hirji (2006), where it is referred to as the *Combined Tails* method.

`"absdist"` The probabilities of the absolute distances from the expected value that are greater than or equal to the observed one are summed up. In Hirji (2006), it is referred to as the *Distance from Center* approach.

`"central"` The smaller values of the observations' simply doubles the minimum of lower and upper tail probabilities. In Hirji (2006), it is referred to as the *Twice the Smaller Tail* method.

For non-exact (i.e. continuous approximation) approaches, `ts_method` is ignored, since all its methods would yield the same p-values. More specifically, they all converge to the doubling approach as in `ts_mthod = "central"`.

Value

If `simple.output = TRUE`, a vector of computed p-values is returned. Otherwise, the output is a `DiscreteTestResults` R6 class object, which also includes the p-value supports and testing parameters. These have to be accessed by public methods, e.g. `$get_pvalues()`.

References

Fisher, R. A. (1935). The logic of inductive inference. *Journal of the Royal Statistical Society Series A*, **98**, pp. 39–54. doi:10.2307/2342435

Agresti, A. (2002). *Categorical data analysis* (2nd ed.). New York: John Wiley & Sons. pp. 91–97. doi:10.1002/0471249688

Blaker, H. (2000) Confidence curves and improved exact confidence intervals for discrete distributions. *Canadian Journal of Statistics*, **28**(4), pp. 783-798. doi:10.2307/3315916

Hirji, K. F. (2006). *Exact analysis of discrete data*. New York: Chapman and Hall/CRC. pp. 55-83. doi:10.1201/9781420036190

See Also

`stats::fisher.test()`

Examples

```
# Constructing
S1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
S2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
F1 <- N1 - S1
F2 <- N2 - S2
df <- data.frame(S1, F1, S2, F2)

# Computation of Fisher's exact p-values (default: "minlike") and their supports
results_f   <- fisher_test_pv(df)
raw_pvalues <- results_f$get_pvalues()
pCDFlist    <- results_f$get_pvalue_supports()

# Computation of p-values of chi-square tests and their supports
results_c   <- fisher_test_pv(df, exact = FALSE)
raw_pvalues <- results_c$get_pvalues()
pCDFlist    <- results_c$get_pvalue_supports()
```

`mcnemar_test_pv` *McNemar's Test for Count Data*

Description

Performs McNemar's chi-square test or an exact variant to assess the symmetry of rows and columns in a 2-by-2 contingency table. In contrast to `stats::mcnemar.test()`, it is vectorised, only calculates p-values and offers their exact computation. Furthermore, it is capable of returning the discrete p-value supports, i.e. all observable p-values under a null hypothesis. Multiple tables can be analysed simultaneously. In two-sided tests, several procedures of obtaining the respective p-values are implemented. It is a special case of the binomial test.

[Deprecated]
Note: Please use `mcnemar_test_pv()`! The older `mcnemar.test.pv()` is deprecated in order to migrate to snake case. It will be removed in a future version.

Usage

```
mcnemar_test_pv(
  x,
  alternative = "two.sided",
  exact = TRUE,
  correct = TRUE,
  simple_output = FALSE
)

mcnemar.test.pv(
  x,
  alternative = "two.sided",
  exact = TRUE,
  correct = TRUE,
  simple.output = FALSE
)
```

Arguments

`x`	integer vector with four elements, a 2-by-2 matrix or an integer matrix (or data frame) with four columns where each line represents a 2-by-2 table to be tested.
`alternative`	character vector that indicates the alternative hypotheses; each value must be one of `"two.sided"` (the default), `"less"` or `"greater"`.
`exact`	logical value that indicates whether p-values are to be calculated by exact computation (`exact = TRUE`; the default) or by a continuous approximation.
`correct`	logical value that indicates if a continuity correction is to be applied (`correct = TRUE`; the default) or not. Ignored, if `exact = TRUE`.
`simple_output, simple.output`	logical value that indicates whether an R6 class object, including the tests' parameters and support sets, i.e. all observable p-values under each null hypothesis, is to be returned (see below).

Details

The parameters `x` and `alternative` are vectorised. They are replicated automatically, such that the number of `x`'s rows is the same as the length of `alternative`. This allows multiple null hypotheses to be tested simultaneously. Since 'x is (if necessary) coerced to a matrix with four columns, it is replicated row-wise.

It can be shown that McNemar's test is a special case of the binomial test. Therefore, its computations are handled by `binom_test_pv()`. In contrast to that function, `mcnemar_test_pv()` does not allow specifying exact two-sided p-value calculation procedures. The reason is that McNemar's exact test always tests for a probability of 0.5, in which case all these exact two-sided p-value computation methods yield exactly the same results.

Value

If `simple.output = TRUE`, a vector of computed p-values is returned. Otherwise, the output is a `DiscreteTestResults` R6 class object, which also includes the p-value supports and testing

parameters. These have to be accessed by public methods, e.g. `$get_pvalues()`.

References

Agresti, A. (2002). *Categorical data analysis* (2nd ed.). New York: John Wiley & Sons. pp. 411–413. doi:10.1002/0471249688

See Also

`stats::mcnemar.test()`, `binom_test_pv()`

Examples

```
# Constructing
S1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
S2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
F1 <- N1 - S1
F2 <- N2 - S2
df <- data.frame(S1, F1, S2, F2)

# Computation of exact p-values and their supports
results_ex  <- mcnemar_test_pv(df)
raw_pvalues <- results_ex$get_pvalues()
pCDFlist    <- results_ex$get_pvalue_supports()

# Computation of chisquare p-values and their supports
results_cs  <- mcnemar_test_pv(df, exact = FALSE)
raw_pvalues <- results_cs$get_pvalues()
pCDFlist    <- results_cs$get_pvalue_supports()
```

`poisson_test_pv` *Poisson Test*

Description

`poisson_test_pv()` performs an exact or approximate Poisson test about the rate parameter of a Poisson distribution. In contrast to `stats::poisson.test()`, it is vectorised, only calculates p-values and offers a normal approximation of their computation. Furthermore, it is capable of returning the discrete p-value supports, i.e. all observable p-values under a null hypothesis. Multiple tests can be evaluated simultaneously. In two-sided tests, several procedures of obtaining the respective p-values are implemented.

[Deprecated]
Note: Please use `poisson_test_pv()`! The older `poisson.test.pv()` is deprecated in order to migrate to snake case. It will be removed in a future version.

Usage

```
poisson_test_pv(
  x,
  lambda = 1,
  alternative = "two.sided",
  ts_method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple_output = FALSE
)

poisson.test.pv(
  x,
  lambda = 1,
  alternative = "two.sided",
  ts.method = "minlike",
  exact = TRUE,
  correct = TRUE,
  simple.output = FALSE
)
```

Arguments

`x`	integer vector giving the number of events.
`lambda`	non-negative numerical vector of hypothesised rate(s).
`alternative`	character vector that indicates the alternative hypotheses; each value must be one of `"two.sided"` (the default), `"less"` or `"greater"`.
`ts_method`, `ts.method`	single character string that indicates the two-sided p-value computation method (if any value in `alternative` equals `"two.sided"`) and must be one of `"minlike"` (the default), `"blaker"`, `"absdist"` or `"central"` (see details). Ignored, if `exact = FALSE`.
`exact`	logical value that indicates whether p-values are to be calculated by exact computation (`exact = TRUE`; the default) or by a continuous approximation.
`correct`	logical value that indicates if a continuity correction is to be applied (`correct = TRUE`; the default) or not. Ignored, if `exact = TRUE`.
`simple_output`, `simple.output`	logical value that indicates whether an R6 class object, including the tests' parameters and support sets, i.e. all observable p-values under each null hypothesis, is to be returned (see below).

Details

The parameters `x`, `lambda` and `alternative` are vectorised. They are replicated automatically to have the same lengths. This allows multiple null hypotheses to be tested simultaneously.

Since the Poisson distribution itself has an infinite support, so do the p-values of exact Poisson tests. Thus supports only contain p-values that are not rounded off to 0.

For exact computation, various procedures of determining two-sided p-values are implemented.

`"minlike"` The standard approach in `stats::fisher.test()` and `stats::binom.test()`. The probabilities of the likelihoods that are equal or less than the observed one are summed up. In Hirji (2006), it is referred to as the *Probability-based* approach.

`"blaker"` The minima of the observations' lower and upper tail probabilities are combined with the opposite tail not greater than these minima. More details can be found in Blaker (2000) or Hirji (2006), where it is referred to as the *Combined Tails* method.

`"absdist"` The probabilities of the absolute distances from the expected value that are greater than or equal to the observed one are summed up. In Hirji (2006), it is referred to as the *Distance from Center* approach.

`"central"` The smaller values of the observations' simply doubles the minimum of lower and upper tail probabilities. In Hirji (2006), it is referred to as the *Twice the Smaller Tail* method.

For non-exact (i.e. continuous approximation) approaches, `ts_method` is ignored, since all its methods would yield the same p-values. More specifically, they all converge to the doubling approach as in `ts_mthod = "central"`.

Value

If `simple.output = TRUE`, a vector of computed p-values is returned. Otherwise, the output is a `DiscreteTestResults` R6 class object, which also includes the p-value supports and testing parameters. These have to be accessed by public methods, e.g. `$get_pvalues()`.

References

Blaker, H. (2000) Confidence curves and improved exact confidence intervals for discrete distributions. *Canadian Journal of Statistics*, **28**(4), pp. 783-798. doi:10.2307/3315916

Hirji, K. F. (2006). *Exact analysis of discrete data*. New York: Chapman and Hall/CRC. pp. 55-83. doi:10.1201/9781420036190

See Also

`stats::poisson.test()`, `binom_test_pv()`

Examples

```
# Constructing
k <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
lambda <- c(3, 2, 1)

# Computation of exact two-sided p-values ("blaker") and their supports
results_ex  <- poisson_test_pv(k, lambda, ts_method = "blaker")
raw_pvalues <- results_ex$get_pvalues()
pCDFlist    <- results_ex$get_pvalue_supports()

# Computation of normal-approximated one-sided p-values ("less") and their supports
results_ap  <- poisson_test_pv(k, lambda, "less", exact = FALSE)
raw_pvalues <- results_ap$get_pvalues()
pCDFlist    <- results_ap$get_pvalue_supports()
```

summary.DiscreteTestResults

Summarizing Discrete Test Results

Description

summary method for class DiscreteTestResults.

Usage

```
## S3 method for class 'DiscreteTestResults'
summary(object, ...)
```

Arguments

object	object of class DiscreteTestResults to be summarised; usually created by using one of the packages test functions, e.g. binom.test.pv(), with simple.output = FALSE.
...	further arguments passed to or from other methods.

Value

A summary.DiscreteTestResults R6 class object.

Examples

```
# binomial tests
obj <- binom.test.pv(0:5, 5, 0.5)
# print summary
summary(obj)
# extract summary table
smry <- summary(obj)
smry$get_summary_table()
```

Index