Package ‘tolerance’

April 17, 2024

Type Package

Title Statistical Tolerance Intervals and Regions

Version 3.0.0

Date 2024-04-18

Maintainer Derek S. Young <derek.young@uky.edu>

Depends R (>= 3.5.0)

Imports MASS, plotly, stats4

Description Statistical tolerance limits provide the limits between which we can expect to find a specified proportion of a sampled population with a given level of confidence. This package provides functions for estimating tolerance limits (intervals) for various univariate distributions (binomial, Cauchy, discrete Pareto, exponential, two-parameter exponential, extreme value, hypergeometric, Laplace, logistic, negative binomial, negative hypergeometric, normal, Pareto, Poisson-Lindley, Poisson, uniform, and Zipf-Mandelbrot), Bayesian normal tolerance limits, multivariate normal tolerance regions, nonparametric tolerance intervals, tolerance bands for regression settings (linear regression, nonlinear regression, nonparametric regression, and multivariate regression), and analysis of variance tolerance intervals. Visualizations are also available for most of these settings.

License GPL (>= 2)

NeedsCompilation no

Author Derek S. Young [aut, cre] (<https://orcid.org/0000-0002-3048-3803>),
Kedai Cheng [aut] (<https://orcid.org/0009-0004-2720-1299>)

Repository CRAN

Date/Publication 2024-04-17 21:30:03 UTC

R topics documented:

`tolerance-package` *Statistical Tolerance Intervals and Regions*

Description

Statistical tolerance limits provide the limits between which we can expect to find a specified proportion of a sampled population with a given level of confidence. This package provides functions for estimating tolerance limits (intervals) for various univariate distributions (binomial, Cauchy, discrete Pareto, exponential, two-parameter exponential, extreme value, hypergeometric, Laplace, logistic, negative binomial, negative hypergeometric, normal, Pareto, Poisson-Lindley, Poisson, uniform, and Zipf-Mandelbrot), Bayesian normal tolerance limits, multivariate normal tolerance regions, nonparametric tolerance intervals, tolerance bands for regression settings (linear regression, nonlinear regression, nonparametric regression, and multivariate regression), and analysis of variance tolerance intervals. Visualizations are also available for most of these settings.

Details

Package:	tolerance
Type:	Package
Version:	2.0.0
Date:	2020-02-04
Imports:	MASS, rgl, stats4
License:	GPL (>= 2)

Author(s)

Derek S. Young, Ph.D.

Maintainer: Derek S. Young <derek.young@uky.edu>

References

Hahn, G. J. and Meeker, W. Q. (1991), *Statistical Intervals: A Guide for Practitioners*, Wiley-Interscience.

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Patel, J. K. (1986), Tolerance Intervals - A Review, *Communications in Statistics - Theory and Methodology*, **15**, 2719–2762.

Young, D. S. (2010), `tolerance`: An `R` Package for Estimating Tolerance Intervals, *Journal of Statistical Software*, **36**(5), 1–39.

Young, D. S. (2014), Computing Tolerance Intervals and Regions in `R`. In M. B. Rao and C. R. Rao (eds.), *Handbook of Statistics, Volume 32: Computational Statistics with* `R`, 309–338. North-Holland, Amsterdam.

See Also

`confint`

`acc.samp` *Acceptance Sampling*

Description

Provides an upper bound on the number of acceptable rejects or nonconformities in a process. This is similar to a 1-sided upper tolerance bound for a hypergeometric random variable.

Usage

```
acc.samp(n, N, alpha = 0.05, P = 0.99, AQL = 0.01, RQL = 0.02)
```

Arguments

`n`	The sample size to be drawn from the inventory.
`N`	The total inventory (or lot) size.
`alpha`	`1-alpha` is the confidence level for bounding the probability of accepting the inventory.
`P`	The proportion of items in the inventory which are to be accountable.
`AQL`	The acceptable quality level, which is the largest proportion of defects in a process considered acceptable. Note that `0` < `AQL` < `1`.
`RQL`	The rejectable quality level, which is the largest proportion of defects in an independent lot that one is willing to tolerate. Note that `AQL` < `RQL` < `1`.

Value

`acc.samp` returns a matrix with the following quantities:

`acceptance.limit`	The number of items in the sample which may be unaccountable, yet still be able to attain the desired confidence level `1-alpha`.
`lot.size`	The total inventory (or lot) size `N`.
`confidence`	The confidence level `1-alpha`.
`P`	The proportion of accountable items specified by the user.
`AQL`	The acceptable quality level as specified by the user. If the sampling were to be repeated numerous times as a process, then this quantity specifies the proportion of missing items considered acceptable from the process as a whole. Conditioning on the calculated value for `acceptance.limit`, the `AQL` is used to estimate the producer's risk (see `prod.risk` below).
`RQL`	The rejectable quality level as specified by the user. This is the proportion of individual items in a sample one is willing to tolerate missing. Conditioning on the calculated value for `acceptance.limit`, the `RQL` is used to estimate the consumer's risk (see `cons.risk` below).
`sample.size`	The sample size drawn as specified by `n`.
`prod.risk`	The producer's risk at the specified `AQL`. This is the probability of rejecting an audit of a good inventory (also called the Type I error). A good inventory can be rejected if an unfortunate random sample is selected (e.g., most of the missing items happened to be selected for the audit). `1-prod.risk` gives the confidence level of this sampling plan for the specified `AQL` and `RQL`. If it is lower than the confidence level desired (e.g., because the `AQL` is too high), then a warning message will be displayed.
`cons.risk`	The consumer's risk at the specified `RQL`. This is the probability of accepting an audit of a bad inventory (also called the Type II error). A bad inventory can be accepted if a fortunate random sample is selected (e.g., most of the missing items happened to not be selected for the audit).

References

Montgomery, D. C. (2005), *Introduction to Statistical Quality Control*, Fifth Edition, John Wiley & Sons, Inc.

See Also

`Hypergeometric`

Examples

```
## A 90%/90% acceptance sampling plan for a sample of 450
## drawn from a lot size of 960.

acc.samp(n = 450, N = 960, alpha = 0.10, P = 0.90, AQL = 0.07,
        RQL = 0.10)
```

anovatol.int *Tolerance Intervals for ANOVA*

Description

Tolerance intervals for each factor level in a balanced (or nearly-balanced) ANOVA.

Usage

```
anovatol.int(lm.out, data, alpha = 0.05, P = 0.99, side = 1,
             method = c("HE", "HE2", "WBE", "ELL", "KM",
             "EXACT", "OCT"), m = 50)
```

Arguments

`lm.out`	An object of class `lm` (i.e., the results from the linear model fitting routine such that the `anova` function can act upon).
`data`	A data frame consisting of the data fitted in `lm.out`. Note that `data` must have one column for each main effect (i.e., factor) that is analyzed in `lm.out` and that these columns must be of class `factor`.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. `"HE"` is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. `"HE2"` is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. `"WBE"` is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. `"ELL"` is the Ellison correction to the Weissberg-Beatty method when `f` is appreciably larger than `n^2`. A warning message is displayed if `f` is not larger than `n^2`. `"KM"` is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. `"EXACT"` computes the k-factor exactly by finding the integral solution to the problem via the `integrate` function. Note the computation time of this method is largely determined by `m`. `"OCT"` is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
`m`	The maximum number of subintervals to be used in the `integrate` function. This is necessary only for `method = "EXACT"` and `method = "OCT"`. The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for `method = "EXACT"`.

Value

`anovatol.int` returns a list where each element is a data frame corresponding to each main effect (i.e., factor) tested in the ANOVA and the rows of each data frame are the levels of that factor. The columns of each data frame report the following:

`mean`	The mean for that factor level.
`n`	The effective sample size for that factor level.
`k`	The k-factor for constructing the respective factor level's tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Howe, W. G. (1969), Two-Sided Tolerance Limits for Normal Populations - Some Improvements, *Journal of the American Statistical Association*, **64**, 610–620.

Weissberg, A. and Beatty, G. (1969), Tables of Tolerance Limit Factors for Normal Distributions, *Technometrics*, **2**, 483–500.

See Also

`K.factor`, `normtol.int`, `lm`, `anova`

Examples

```
## 90%/95% 2-sided tolerance intervals for a 2-way ANOVA
## using the "warpbreaks" data.

attach(warpbreaks)

lm.out <- lm(breaks ~ wool + tension)
out <- anovatol.int(lm.out, data = warpbreaks, alpha = 0.10,
                    P = 0.95, side = 2, method = "HE")
out

plottol(out, x = warpbreaks)
```

bayesnormtol.int *Bayesian Normal Tolerance Intervals*

Description

Provides 1-sided or 2-sided Bayesian tolerance intervals under the conjugate prior for data distributed according to a normal distribution.

Usage

```
bayesnormtol.int(x = NULL, norm.stats = list(x.bar = NA,
                 s = NA, n = NA), alpha = 0.05, P = 0.99,
                 side = 1, method = c("HE", "HE2", "WBE",
                 "ELL", "KM", "EXACT", "OCT"), m = 50,
                 hyper.par = list(mu.0 = NULL,
                 sig2.0 = NULL, m.0 = NULL, n.0 = NULL))
```

Arguments

`x`	A vector of data which is distributed according to a normal distribution.
`norm.stats`	An optional list of statistics that can be provided in-lieu of the full dataset. If provided, the user must specify all three components: the sample mean (`x.bar`), the sample standard deviation (`s`), and the sample size (`n`).
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. `"HE"` is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. `"HE2"` is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. `"WBE"` is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. `"ELL"` is the Ellison correction to the Weissberg-Beatty method when `f` is appreciably larger than `n^2`. A warning message is displayed if `f` is not larger than `n^2`. `"KM"` is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. `"EXACT"` computes the k-factor exactly by finding the integral solution to the problem via the `integrate` function. Note the computation time of this method is largely determined by `m`. `"OCT"` is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
`m`	The maximum number of subintervals to be used in the `integrate` function. This is necessary only for `method = "EXACT"` and `method = "OCT"`. The larger the number, the more accurate the solution. Too low of a value can result in

	an error. A large value can also cause the function to be slow for `method = "EXACT"`.
`hyper.par`	A list consisting of the hyperparameters for the conjugate prior: the hyperparameters for the mean (`mu.0` and `n.0`) and the hyperparameters for the variance (`sig2.0` and `m.0`).

Details

Note that if one considers the non-informative prior distribution, then the Bayesian tolerance intervals are the same as the classical solution, which can be obtained by using `normtol.int`.

Value

`bayesnormtol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`x.bar`	The sample mean.
`1-sided.lower`	The 1-sided lower Bayesian tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper Bayesian tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower Bayesian tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper Bayesian tolerance bound. This is given only if `side = 2`.

References

Aitchison, J. (1964), Bayesian Tolerance Regions, *Journal of the Royal Statistical Society, Series B*, **26**, 161–175.

Guttman, I. (1970), *Statistical Tolerance Regions: Classical and Bayesian*, Charles Griffin and Company.

Young, D. S., Gordon, C. M., Zhu, S., and Olin, B. D. (2016), Sample Size Determination Strategies for Normal Tolerance Intervals Using Historical Data, *Quality Engineering*, **28**, 337–351.

See Also

`Normal`, `normtol.int`, `K.factor`

Examples

```
## 95%/85% 2-sided Bayesian normal tolerance limits for
## a sample of size 100.

set.seed(100)
x <- rnorm(100)
out <- bayesnormtol.int(x = x, alpha = 0.05, P = 0.85,
                        side = 2, method = "EXACT",
                        hyper.par = list(mu.0 = 0,
                        sig2.0 = 1, n.0 = 10, m.0 = 10))
```

```
out

plottol(out, x, plot.type = "both", side = "upper",
        x.lab = "Normal Data")
```

bintol.int *Binomial Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for binomial random variables. From a statistical quality control perspective, these limits use the proportion of defective (or acceptable) items in a sample to bound the number of defective (or acceptable) items in future productions of a specified quantity.

Usage

```
bintol.int(x, n, m = NULL, alpha = 0.05, P = 0.99, side = 1,
           method = c("LS", "WS", "AC", "JF", "CP", "AS",
           "LO", "PR", "PO", "CL", "CC", "CWS"),
           a1 = 0.5, a2 = 0.5)
```

Arguments

`x`	The number of defective (or acceptable) units in the sample. Can be a vector of length `n`, in which case the sum of `x` is used.
`n`	The size of the random sample of units selected for inspection.
`m`	The quantity produced in future groups. If `m = NULL`, then the tolerance limits will be constructed assuming `n` for this quantity.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the defective (or acceptable) units in future samples of size `m` to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the lower and upper confidence bounds, which are used in the calculation of the tolerance bounds. The default method is `"LS"`, which is the large-sample method. `"WS"` is Wilson's method, which is just the score confidence interval. `"AC"` gives the Agresti-Coull method, which is also appropriate when the sample size is large. `"JF"` is Jeffreys' method, which is a Bayesian approach to the estimation. `"CP"` is the Clopper-Pearson (exact) method, which is based on beta percentiles and provides a more conservative interval. `"AS"` is the arcsine method, which is appropriate when the sample proportion is not too close to 0 or 1. `"LO"` is the logit method, which also is appropriate when the sample proportion is not too close to 0 or 1, but yields a more conservative interval. `"PR"` uses a probit transformation and is accurate for large sample sizes. `"PO"` is based on a Poisson parameterization, but it tends

	to be more erratic compared to the other methods. "CL" is the complementary log transformation and also tends to perform well for large sample sizes. "CC" gives a continuity-corrected version of the large-sample method. "CWS" gives a continuity-corrected version of Wilson's method. More information on these methods can be found in the "References".
a1	This specifies the first shape hyperparameter when using Jeffreys' method.
a2	This specifies the second shape hyperparameter when using Jeffreys' method.

Value

bintol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of defective (or acceptable) units in future samples of size m.
p.hat	The proportion of defective (or acceptable) units in the sample, calculated by x/n.
1-sided.lower	The 1-sided lower tolerance bound. This is given only if side = 1.
1-sided.upper	The 1-sided upper tolerance bound. This is given only if side = 1.
2-sided.lower	The 2-sided lower tolerance bound. This is given only if side = 2.
2-sided.upper	The 2-sided upper tolerance bound. This is given only if side = 2.

References

Brown, L. D., Cai, T. T., and DasGupta, A. (2001), Interval Estimation for a Binomial Proportion, *Statistical Science*, **16**, 101–133.

Hahn, G. J. and Chandra, R. (1981), Tolerance Intervals for Poisson and Binomial Variables, *Journal of Quality Technology*, **13**, 100–110.

Newcombe, R. G. (1998), Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods, *Statistics in Medicine*, **17**, 857–872.

See Also

Binomial, umatol.int

Examples

```
## 85%/90% 2-sided binomial tolerance intervals for a future
## lot of 2500 when a sample of 230 were drawn from a lot of
## 1000.  All methods but Jeffreys' method are compared
## below.

bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "LS")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "WS")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "AC")
```

```
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "CP")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "AS")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "LO")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "PR")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "PO")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "CL")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "CC")
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 2, method = "CWS")

## Using Jeffreys' method to construct the 85%/90% 1-sided
## binomial tolerance limits.  The first calculation assumes
## a prior on the proportion of defects which places greater
## density on values near 0.  The second calculation assumes
## a prior on the proportion of defects which places greater
## density on values near 1.

bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 1, method = "JF", a1 = 2, a2 = 10)
bintol.int(x = 230, n = 1000, m = 2500, alpha = 0.15, P = 0.90,
           side = 1, method = "JF", a1 = 5, a2 = 1)
```

bonftol.int — *Approximate 2-Sided Tolerance Intervals that Control the Tails Using Bonferroni's Inequality*

Description

This function allows the user to control what proportion of the population is to be in the tails of the given distribution for a 2-sided tolerance interval. The result is a conservative approximation based on Bonferroni's inequality.

Usage

```
bonftol.int(fn, P1 = 0.005, P2 = 0.005, alpha = 0.05, ...)
```

Arguments

`fn`	The function name for the 2-sided tolerance interval to be calculated.

`P1`	The proportion of the population not covered in the lower tail of the distribution.
`P2`	The proportion of the population not covered in the upper tail of the distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`...`	Additional arguments passed to `fn`, including the data. All arguments that would be specified in `fn` must also be specified here.

Value

The results for the 2-sided tolerance interval procedure are reported. See the corresponding help file for `fn` about specific output. Note that the (minimum) proportion of the population to be covered by this interval is `1 - (P1 + P2)`.

Note

This function can be used with any 2-sided tolerance interval function, including the regression tolerance interval functions.

References

Jensen, W. A. (2009), Approximations of Tolerance Intervals for Normally Distributed Data, *Quality and Reliability Engineering International*, **25**, 571–580.

Patel, J. K. (1986), Tolerance Intervals - A Review, *Communications in Statistics - Theory and Methodology*, **15**, 2719–2762.

Examples

```
## 95%/97% tolerance interval for normally distributed
## data controlling 1% of the data is in the lower tail
## and 2% of the data in the upper tail.

set.seed(100)
x <- rnorm(100, 0, 0.2)
bonftol.int(normtol.int, x = x, P1 = 0.01, P2 = 0.02,
            alpha = 0.05, method = "HE")
```

`cautol.int` *Cauchy Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for Cauchy distributed data.

Usage

```
cautol.int(x, alpha = 0.05, P = 0.99, side = 1)
```

Arguments

`x`	A vector of data which is Cauchy distributed.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).

Value

`cautol.int` returns a data.frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Bain, L. J. (1978), *Statistical Analysis of Reliability and Life-Testing Models*, Marcel Dekker, Inc.

See Also

`Cauchy`

Examples

```
## 95%/90% 2-sided Cauchy tolerance interval for a sample
## of size 1000.

set.seed(100)
x <- rcauchy(1000, 100000, 10)
out <- cautol.int(x = x, alpha = 0.05, P = 0.90, side = 2)
out

plottol(out, x, plot.type = "both", x.lab = "Cauchy Data")
```

diffnormtol.int — *1-Sided Tolerance Limits for the Distribution of the Difference Between Two Independent Normal Random Variables*

Description

Provides 1-sided tolerance limits for the difference between two independent normal random variables. If the ratio of the variances is known, then an exact calculation is performed. Otherwise, approximation methods are implemented.

Usage

```
diffnormtol.int(x1, x2, var.ratio = NULL, alpha = 0.05,
                P = 0.99, method = c("HALL", "GK", "RG"))
```

Arguments

`x1`	A vector of sample data which is distributed according to a normal distribution (sample 1).
`x2`	Another vector of sample data which is distributed according to a normal distribution (sample 2). It can be of a different sample size than the sample specified by `x1`.
`var.ratio`	A specified, known value of the variance ratio (i.e., the ratio of the variance for population 1 to the variance of population 2). If `NULL`, then the variance ratio is estimated according to one of the three methods specified in the `method` argument.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by the tolerance limits.
`method`	The method for estimating the variance ratio. This only needs to be specified in the case when `var.ratio` is not `NULL`. `"HALL"` is Hall's method, which takes a bias-corrected version of the ratio between the sample variance for sample 1 to the sample variance for sample 2. `"GK"` is the Guo-Krishnamoorthy method, which first calculates a bias-corrected version of the ratio between the sample variance for sample 2 to the sample variance for sample 1. The resulting limit is then compared to the limit from Hall's method and the most conservative limit is chosen. `"RG"` is the Reiser-Guttman method, which is a biased version of the variance ratio that is calculated by taking the sample variance for sample 1 to the sample variance for sample 2. Typically, Hall's method or the Guo-Krishnamoorthy method are preferred to the Reiser-Guttman method.

Details

Satterthwaite's approximation for the degrees of freedom is used when the variance ratio is unknown.

Value

`diffnormtol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`diff.bar`	The difference between the sample means.
`1-sided.lower`	The 1-sided lower tolerance bound.
`1-sided.upper`	The 1-sided upper tolerance bound.

Note

Unlike other tolerance interval functions, the output from `diffnormtol.int` cannot be passed to `plottol`.

References

Guo, H. and Krishnamoorthy, K. (2004), New Approximate Inferential Methods for the Reliability Parameter in a Stress-Strength Model: The Normal Case, *Communications in Statistics - Theory and Methods*, **33**, 1715–1731.

Hall, I. J. (1984), Approximate One-Sided Tolerance Limits for the Difference or Sum of Two Independent Normal Variates, *Journal of Quality Technology*, **16**, 15–19.

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Reiser, B. J. and Guttman, I. (1986), Statistical Inference for Pr(Y < X): The Normal Case, *Technometrics*, **28**, 253–257.

See Also

`Normal`, `K.factor`, `normtol.int`

Examples

```
## 90%/99% tolerance limits for the difference between two
## simulated normal data sets.  This data is taken from
## Krishnamoorthy and Mathew (2009).  Note that there is a
## calculational error in their example, which yields different
## results with the output below.

x1 <- c(10.166, 5.889, 8.258, 7.303, 8.757)
x2 <- c(-0.204, 2.578, 1.182, 1.892, 0.786, -0.517, 1.156,
        0.980, 0.323, 0.437, 0.397, 0.050, 0.812, 0.720)

diffnormtol.int(x1, x2, alpha = 0.10, P = 0.99, method = "HALL")
diffnormtol.int(x1, x2, alpha = 0.10, P = 0.99, method = "GK")
diffnormtol.int(x1, x2, alpha = 0.10, P = 0.99, method = "RG")
diffnormtol.int(x1, x2, var.ratio = 3.8, alpha = 0.10, P = 0.99)
```

`DiffProp` *Difference Between Two Proportions Distribution*

Description

Density (mass), distribution function, quantile function, and random generation for the difference between two proportions. This is determined by taking the difference between two independent beta distributions.

Usage

```
ddiffprop(x, k1, k2, n1, n2, a1 = 0.5, a2 = 0.5,
          log = FALSE, ...)
pdiffprop(q, k1, k2, n1, n2, a1 = 0.5, a2 = 0.5,
          lower.tail = TRUE, log.p = FALSE, ...)
qdiffprop(p, k1, k2, n1, n2, a1 = 0.5, a2 = 0.5,
          lower.tail = TRUE, log.p = FALSE, ...)
rdiffprop(n, k1, k2, n1, n2, a1 = 0.5, a2 = 0.5)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	The number of observations. If `length>1`, then the length is taken to be the number required.
`k1, k2`	The number of successes drawn from groups 1 and 2, respectively.
`n1, n2`	The sample sizes for groups 1 and 2, respectively.
`a1, a2`	The shift parameters for the beta distributions. For the fiducial approach, we know that the lower and upper limits are set at `a1 = a2 = 0` and `a1 = a2 = 1`, respectively, for the true `p1` and `p2`. While computations can be performed on real values outside the unit interval, a `warning` message will be returned if such values are specified. For practical purposes, the default value of 0.5 should be used for each parameter.
`log, log.p`	Logical vectors. If `TRUE`, then the probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.
`...`	Additional arguments passed to the Appell `F1` function.

Details

The difference between two proportions distribution has a fairly complicated functional form. Please see the article by Chen and Luo (2011), who corrected a typo in the article by Nadarajah and Kotz (2007), for the functional form of this distribution.

Value

`ddiffprop` gives the density (mass), `pdiffprop` gives the distribution function, `qdiffprop` gives the quantile function, and `rdiffprop` generates random deviates.

References

Chen, Y. and Luo, S. (2011), A Few Remarks on 'Statistical Distribution of the Difference of Two Proportions', *Statistics in Medicine*, **30**, 1913–1915.

Nadarajah, S. and Kotz, S. (2007), Statistical Distribution of the Difference of Two Proportions, *Statistics in Medicine*, **26**, 3518–3523.

See Also

`runif` and `.Random.seed` about random number generation.

Examples

```
## Randomly generated data from the difference between
## two proportions distribution.

set.seed(100)
x <- rdiffprop(n = 100, k1 = 2, k2 = 10, n1 = 17, n2 = 13)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- ddiffprop(x = x.1, k1 = 2, k2 = 10, n1 = 17, n2 = 13)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pdiffprop(q = x.1, k1 = 2, k2 = 10, n1 = 17,
     n2 = 13), type = "l", xlab = "x",
     ylab = "Cumulative Probabilities")

qdiffprop(p = 0.20, k1 = 2, k2 = 10, n1 = 17, n2 = 13,
          lower.tail = FALSE)
qdiffprop(p = 0.80, k1 = 2, k2 = 10, n1 = 17, n2 = 13)
```

`DiscretePareto` *Discrete Pareto Distribution*

Description

Density (mass), distribution function, quantile function, and random generation for the discrete Pareto distribution.

Usage

```
ddpareto(x, theta, log = FALSE)
pdpareto(q, theta, lower.tail = TRUE, log.p = FALSE)
qdpareto(p, theta, lower.tail = TRUE, log.p = FALSE)
rdpareto(n, theta)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	The number of observations. If `length>1`, then the length is taken to be the number required.
`theta`	The shape parameter, which must be greater than 0 and less than 1.
`log, log.p`	Logical vectors. If `TRUE`, then the probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.

Details

The discrete Pareto distribution has mass

$$p(x) = \theta^{\log(1+x)} - \theta^{\log(2+x)},$$

where $x = 0, 1, \ldots$ and $0 < \theta < 1$ is the shape parameter.

Value

`ddpareto` gives the density (mass), `pdpareto` gives the distribution function, `qdpareto` gives the quantile function, and `rdpareto` generates random deviates for the specified distribution.

References

Krishna, H. and Pundir, P. S. (2009), Discrete Burr and Discrete Pareto Distributions, *Statistical Methodology*, **6**, 177–188.

Young, D. S., Naghizadeh Qomi, M., and Kiapour, A. (2019), Approximate Discrete Pareto Tolerance Limits for Characterizing Extremes in Count Data, *Statistica Neerlandica*, **73**, 4–21.

See Also

`runif` and `.Random.seed` about random number generation.

Examples

```
## Randomly generated data from the discrete Pareto
## distribution.

set.seed(100)
x <- rdpareto(n = 150, theta = 0.2)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- ddpareto(x = x.1, theta = 0.2)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pdpareto(q = x.1, theta = 0.2), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")
```

```
qdpareto(p = 0.80, theta = 0.2, lower.tail = FALSE)
qdpareto(p = 0.95, theta = 0.2)
```

distfree.est — *Estimating Various Quantities for Distribution-Free Tolerance Intervals*

Description

When providing two of the three quantities `n`, `alpha`, and `P`, this function solves for the third quantity in the context of distribution-free tolerance intervals.

Usage

```
distfree.est(n = NULL, alpha = NULL, P = NULL, side = 1)
```

Arguments

`n`	The necessary sample size to cover a proportion `P` of the population with confidence `1-alpha`. Can be a vector.
`alpha`	1 minus the confidence level attained when it is desired to cover a proportion `P` of the population and a sample size `n` is provided. Can be a vector.
`P`	The proportion of the population to be covered with confidence `1-alpha` when a sample size `n` is provided. Can be a vector.
`side`	Whether a 1-sided or 2-sided tolerance interval is assumed (determined by `side = 1` or `side = 2`, respectively).

Value

When providing two of the three quantities `n`, `alpha`, and `P`, `distfree.est` returns the third quantity. If more than one value of a certain quantity is specified, then a table will be returned.

References

Natrella, M. G. (1963), *Experimental Statistics: National Bureau of Standards - Handbook No. 91*, United States Government Printing Office, Washington, D.C.

See Also

`nptol.int`

Examples

```
## Solving for 1 minus the confidence level.

distfree.est(n = 59, P = 0.95, side = 1)

## Solving for the sample size.

distfree.est(alpha = 0.05, P = 0.95, side = 1)

## Solving for the proportion of the population to cover.

distfree.est(n = 59, alpha = 0.05, side = 1)

## Solving for sample sizes for many tolerance specifications.

distfree.est(alpha = seq(0.01, 0.05, 0.01),
             P = seq(0.80, 0.99, 0.01), side = 2)
```

`dpareto.ll` *Maximum Likelihood Estimation for the Discrete Pareto Distribution*

Description

Performs maximum likelihood estimation for the parameter of the discrete Pareto distribution.

Usage

```
dpareto.ll(x, theta = NULL, ...)
```

Arguments

`x`	A vector of raw data which is distributed according to a Poisson-Lindley distribution.
`theta`	Optional starting value for the parameter. If `NULL`, then the method of moments estimator is used.
`...`	Additional arguments passed to the `mle` function.

Details

The discrete Pareto distribution is a discretized of the continuous Type II Pareto distribution (also called the Lomax distribution).

Value

See the help file for `mle` to see how the output is structured.

References

Krishna, H. and Pundir, P. S. (2009), Discrete Burr and Discrete Pareto Distributions, *Statistical Methodology*, **6**, 177–188.

Young, D. S., Naghizadeh Qomi, M., and Kiapour, A. (2019), Approximate Discrete Pareto Tolerance Limits for Characterizing Extremes in Count Data, *Statistica Neerlandica*, **73**, 4–21.

See Also

`mle`, `DiscretePareto`

Examples

```
## Maximum likelihood estimation for randomly generated data
## from the discrete Pareto distribution.

set.seed(100)

dp.data <- rdpareto(n = 500, theta = 0.2)
out.dp <- dpareto.ll(dp.data)
stats4::coef(out.dp)
stats4::vcov(out.dp)
```

`dparetotol.int` *Discrete Pareto Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to the discrete Pareto distribution.

Usage

```
dparetotol.int(x, m = NULL, alpha = 0.05, P = 0.99, side = 1,
                  ...)
```

Arguments

`x`	A vector of raw data which is distributed according to a discrete Pareto distribution.
`m`	The number of observations in a future sample for which the tolerance limits will be calculated. By default, `m = NULL` and, thus, `m` will be set equal to the original sample size.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`...`	Additional arguments passed to the `dpareto.ll` function, which is used for maximum likelihood estimation.

Details

The discrete Pareto is a discretized of the continuous Type II Pareto distribution (also called the Lomax distribution). Discrete Pareto distributions are heavily right-skewed distributions and potentially good models for discrete lifetime data and extremes in count data. For most practical applications, one will typically be interested in 1-sided upper bounds.

Value

`dparetotol.int` returns a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`theta`	MLE for the shape parameter `theta`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Young, D. S., Naghizadeh Qomi, M., and Kiapour, A. (2019), Approximate Discrete Pareto Tolerance Limits for Characterizing Extremes in Count Data, *Statistica Neerlandica*, **73**, 4–21.

See Also

`DiscretePareto`, `dpareto.ll`

Examples

```
## 95%/95% 1-sided tolerance intervals for data assuming
## the discrete Pareto distribution.

set.seed(100)

x <- rdpareto(n = 500, theta = 0.5)
out <- dparetotol.int(x, alpha = 0.05, P = 0.95, side = 1)
out
```

`exp2tol.int` *2-Parameter Exponential Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to a 2-parameter exponential distribution. Data with Type II censoring is permitted.

Usage

```
exp2tol.int(x, alpha = 0.05, P = 0.99, side = 1,
            method = c("GPU", "DUN", "KM"), type.2 = FALSE)
```

Arguments

`x`	A vector of data which is distributed according to the 2-parameter exponential distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for how the upper tolerance bound is approximated. `"GPU"` is the Guenther-Patil-Upppuluri method. `"DUN"` is the Dunsmore method, which has been empirically shown to be an improvement for samples greater than or equal to 8. `"KM"` is the Krishnamoorthy-Mathew method, which is typically more liberal than the other methods. More information on these methods can be found in the "References", which also highlight general sample size conditions as to when these different methods should be used.
`type.2`	Select `TRUE` if Type II censoring is present (i.e., the data set is censored at the maximum value present). The default is `FALSE`.

Value

`exp2tol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Dunsmore, I. R. (1978), Some Approximations for Tolerance Factors for the Two Parameter Exponential Distribution, *Technometrics*, **20**, 317–318.

Engelhardt, M. and Bain, L. J. (1978), Tolerance Limits and Confidence Limits on Reliability for the Two-Parameter Exponential Distribution, *Technometrics*, **20**, 37–39.

Guenther, W. C., Patil, S. A., and Uppuluri, V. R. R. (1976), One-Sided β-Content Tolerance Factors for the Two Parameter Exponential Distribution, *Technometrics*, **18**, 333–340.

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

See Also

TwoParExponential

Examples

```
## 95%/90% 1-sided 2-parameter exponential tolerance intervals
## for a sample of size 50.

set.seed(100)
x <- r2exp(50, 6, shift = 55)
out <- exp2tol.int(x = x, alpha = 0.05, P = 0.90, side = 1,
                   method = "DUN", type.2 = FALSE)
out

plottol(out, x, plot.type = "both", side = "upper",
        x.lab = "2-Parameter Exponential Data")
```

---

exptol.int *Exponential Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to an exponential distribution. Data with Type II censoring is permitted.

Usage

```
exptol.int(x, alpha = 0.05, P = 0.99, side = 1, type.2 = FALSE)
```

Arguments

`x`	A vector of data which is distributed according to an exponential distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`type.2`	Select `TRUE` if Type II censoring is present (i.e., the data set is censored at the maximum value present). The default is `FALSE`.

Value

`exptol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`lambda.hat`	The mean of the data (i.e., `1/rate`).

`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Blischke, W. R. and Murthy, D. N. P. (2000), *Reliability: Modeling, Prediction, and Optimization*, John Wiley & Sons, Inc.

See Also

`Exponential`

Examples

```
## 95%/99% 1-sided exponential tolerance intervals for a
## sample of size 50.

set.seed(100)
x <- rexp(100, 0.004)
out <- exptol.int(x = x, alpha = 0.05, P = 0.99, side = 1,
                  type.2 = FALSE)
out

plottol(out, x, plot.type = "both", side = "lower",
        x.lab = "Exponential Data")
```

`exttol.int` *Weibull (or Extreme-Value) Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to either a Weibull distribution or extreme-value (also called Gumbel) distributions.

Usage

```
exttol.int(x, alpha = 0.05, P = 0.99, side = 1,
           dist = c("Weibull", "Gumbel"), ext = c("min", "max"),
           NR.delta = 1e-8)
```

Arguments

`x`	A vector of data which is distributed according to either a Weibull distribution or an extreme-value distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`dist`	Select either `dist = "Weibull"` or `dist = "Gumbel"` if the data is distributed according to the Weibull or extreme-value distribution, respectively.
`ext`	If `dist = "Gumbel"`, then select which extreme is to be modeled for the Gumbel distribution. The Gumbel distribution for the minimum (i.e., `ext = "min"`) corresponds to a left-skewed distribution and the Gumbel distribution for the maximum (i.e., `ext = "max"`) corresponds to a right-skewed distribution
`NR.delta`	The stopping criterion used for the Newton-Raphson algorithm when finding the maximum likelihood estimates of the Weibull or extreme-value distribution.

Details

Recall that the relationship between the Weibull distribution and the extreme-value distribution for the minimum is that if the random variable X is distributed according to a Weibull distribution, then the random variable $Y = ln(X)$ is distributed according to an extreme-value distribution for the minimum.

If `dist = "Weibull"`, then the natural logarithm of the data are taken so that a Newton-Raphson algorithm can be employed to find the MLEs of the extreme-value distribution for the minimum and then the data and MLEs are transformed back appropriately. No transformation is performed if `dist = "Gumbel"`. The Newton-Raphson algorithm is initialized by the method of moments estimators for the parameters.

Value

`exttol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`shape.1`	MLE for the shape parameter if `dist = "Weibull"` or for the location parameter if `dist = "Gumbel"`.
`shape.2`	MLE for the scale parameter if `dist = "Weibull"` or `dist = "Gumbel"`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Bain, L. J. and Engelhardt, M. (1981), Simple Approximate Distributional Results for Confidence and Tolerance Limits for the Weibull Distribution Based on Maximum Likelihood Estimators, *Technometrics*, **23**, 15–20.

Coles, S. (2001), *An Introduction to Statistical Modeling of Extreme Values*, Springer.

See Also

Weibull

Examples

```
## 85%/90% 1-sided Weibull tolerance intervals for a sample
## of size 150.

set.seed(100)
x <- rweibull(150, 3, 75)
out <- exttol.int(x = x, alpha = 0.15, P = 0.90, side = 1,
                  dist = "Weibull")
out

plottol(out, x, plot.type = "both", side = "lower",
        x.lab = "Weibull Data")
```

F1 *Appell's F1 Hypergeometric Function*

Description

The Appell function of the first kind, which is a two variable extension of the hypergeometric distribution.

Usage

```
F1(a, b, b.prime, c, x, y, ...)
```

Arguments

a, b, b.prime, c	Appropriate parameters for this function.
x, y	The inputted values to evaluate this function such that each is less than 1 in absolute value.
...	Additional arguments passed to the `integrate` function.

Value

`F1` returns the simple integral result for the Appell function of the first kind with the arguments specified above.

Note

This function is solved by using a simple integral representation for real numbers. While all four of the Appell functions can be extended to the complex plane, this is not an option for this code.

References

Bailey, W. N. (1935), *Generalised Hypergeometric Series*, Cambridge University Press.

See Also

`DiffProp`, `integrate`

Examples

```
## Sample calculation.

F1(a = 3, b = 4, b.prime = 5, c = 13, x = 0.2, y = 0.4)
```

`fidbintol.int` *Fiducial-Based Tolerance Intervals for the Function of Two Binomial Proportions*

Description

Provides 1-sided or 2-sided tolerance intervals for the function of two binomial proportions using fiducial quantities.

Usage

```
fidbintol.int(x1, x2, n1, n2, m1 = NULL, m2 = NULL, FUN,
              alpha = 0.05, P = 0.99, side = 1, K = 1000,
              B = 1000)
```

Arguments

`x1`	A value of observed "successes" from group 1.
`x2`	A value of observed "successes" from group 2.
`n1`	The total number of trials for group 1.
`n2`	The total number of trials for group 2.
`m1`	The total number of future trials for group 1. If `NULL`, then it is set to `n1`.

`m2`	The total number of future trials for group 2. If `NULL`, then it is set to `n2`.
`FUN`	Any reasonable (and meaningful) function taking exactly two arguments that we are interested in constructing a tolerance interval on.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`K`	The number of fiducial quantities to be generated. The number of iterations should be at least as large as the default value of 1000. See `Details` for the definition of the fiducial quantity for a binomial proportion.
`B`	The number of iterations used for the Monte Carlo algorithm which determines the tolerance limits. The number of iterations should be at least as large as the default value of 1000.

Details

If X is observed from a $Bin(n, p)$ distribution, then the fiducial quantity for p is $Beta(X + 0.5, n - X + 0.5)$.

Value

`fidbintol.int` returns a list with two items. The first item (`tol.limits`) is a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`fn.est`	A point estimate of the functional form of interest using the maximum likelihood estimates calculated with the inputted values of `x1`, `x2`, `n1`, and `n2`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

The second item (`fn`) simply returns the functional form specified by `FUN`.

References

Clopper, C. J. and Pearson, E. S. (1934), The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial, *Biometrika*, **26**, 404–413.

Krishnamoorthy, K. and Lee, M. (2010), Inference for Functions of Parameters in Discrete Distributions Based on Fiducial Approach: Binomial and Poisson Cases, *Journal of Statistical Planning and Inference*, **140**, 1182–1192.

Mathew, T. and Young, D. S. (2013), Fiducial-Based Tolerance Intervals for Some Discrete Distributions, *Computational Statistics and Data Analysis*, **61**, 38–49.

See Also

fidnegbintol.int, fidpoistol.int

Examples

```
## 95%/99% 1-sided and 2-sided tolerance intervals for
## the difference between binomial proportions.

set.seed(100)

p1 <- 0.2
p2 <- 0.4
n1 <- n2 <- 200
x1 <- rbinom(1, n1, p1)
x2 <- rbinom(1, n2, p2)
fun.ti <- function(x, y) x - y

fidbintol.int(x1, x2, n1, n2, m1 = 500, m2 = 500, FUN = fun.ti,
              alpha = 0.05, P = 0.99, side = 1)
fidbintol.int(x1, x2, n1, n2, m1 = 500, m2 = 500, FUN = fun.ti,
              alpha = 0.05, P = 0.99, side = 2)
```

---

fidnegbintol.int — *Fiducial-Based Tolerance Intervals for the Function of Two Negative Binomial Proportions*

Description

Provides 1-sided or 2-sided tolerance intervals for the function of two negative binomial proportions using fiducial quantities.

Usage

```
fidnegbintol.int(x1, x2, n1, n2, m1 = NULL, m2 = NULL, FUN,
                 alpha = 0.05, P = 0.99, side = 1, K = 1000,
                 B = 1000)
```

Arguments

x1	A value of observed "failures" from group 1.
x2	A value of observed "failures" from group 2.
n1	The target number of successes for group 1.
n2	The target number of successes for group 2.
m1	The total number of future trials for group 1. If NULL, then it is set to n1.
m2	The total number of future trials for group 2. If NULL, then it is set to n2.

`FUN`	Any reasonable (and meaningful) function taking exactly two arguments that we are interested in constructing a tolerance interval on.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`K`	The number of fiducial quantities to be generated. The number of iterations should be at least as large as the default value of 1000. See `Details` for the definition of the fiducial quantity for a negative binomial proportion.
`B`	The number of iterations used for the Monte Carlo algorithm which determines the tolerance limits. The number of iterations should be at least as large as the default value of 1000.

Details

If X is observed from a $NegBin(n, p)$ distribution, then the fiducial quantity for p is $Beta(n, X + 0.5)$.

Value

`fidnegbintol.int` returns a list with two items. The first item (`tol.limits`) is a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`fn.est`	A point estimate of the functional form of interest using the maximum likelihood estimates calculated with the inputted values of `x1`, `x2`, `n1`, and `n2`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

The second item (`fn`) simply returns the functional form specified by `FUN`.

References

Cai, Y. and Krishnamoorthy, K. (2005), A Simple Improved Inferential Method for Some Discrete Distributions, *Computational Statistics and Data Analysis*, **48**, 605–621.

Clopper, C. J. and Pearson, E. S. (1934), The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial, *Biometrika*, **26**, 404–413.

Krishnamoorthy, K. and Lee, M. (2010), Inference for Functions of Parameters in Discrete Distributions Based on Fiducial Approach: Binomial and Poisson Cases, *Journal of Statistical Planning and Inference*, **140**, 1182–1192.

Mathew, T. and Young, D. S. (2013), Fiducial-Based Tolerance Intervals for Some Discrete Distributions, *Computational Statistics and Data Analysis*, **61**, 38–49.

See Also

fidbintol.int, fidpoistol.int

Examples

```
## 95%/99% 1-sided and 2-sided tolerance intervals for
## the ratio of odds ratios for negative binomial proportions.

set.seed(100)

p1 <- 0.6
p2 <- 0.2
n1 <- n2 <- 50
x1 <- rnbinom(1, n1, p1)
x2 <- rnbinom(1, n2, p2)
fun.ti <- function(x, y) x * (1 - y) / (y * (1 - x))

fidnegbintol.int(x1, x2, n1, n2, m1 = 50, m2 = 50, FUN = fun.ti,
                 alpha = 0.05, P = 0.99, side = 1)
fidnegbintol.int(x1, x2, n1, n2, m1 = 50, m2 = 50, FUN = fun.ti,
                 alpha = 0.05, P = 0.99, side = 2)
```

---

fidpoistol.int — *Fiducial-Based Tolerance Intervals for the Function of Two Poisson Rates*

Description

Provides 1-sided or 2-sided tolerance intervals for the function of two Poisson rates using fiducial quantities.

Usage

```
fidpoistol.int(x1, x2, n1, n2, m1 = NULL, m2 = NULL, FUN,
               alpha = 0.05, P = 0.99, side = 1, K = 1000,
               B = 1000)
```

Arguments

`x1`	A value of observed counts from group 1.
`x2`	A value of observed counts from group 2.
`n1`	The length of time that `x1` was recorded over.
`n2`	The length of time that `x2` was recorded over.
`m1`	The total number of future trials for group 1. If `NULL`, then it is set to `n1`.
`m2`	The total number of future trials for group 2. If `NULL`, then it is set to `n2`.

`FUN`	Any reasonable (and meaningful) function taking exactly two arguments that we are interested in constructing a tolerance interval on.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`K`	The number of fiducial quantities to be generated. The number of iterations should be at least as large as the default value of 1000. See `Details` for the definition of the fiducial quantity for a Poisson rate.
`B`	The number of iterations used for the Monte Carlo algorithm which determines the tolerance limits. The number of iterations should be at least as large as the default value of 1000.

Details

If X is observed from a $Poi(n * \lambda)$ distribution, then the fiducial quantity for λ is $\chi^2_{2*x+1}/(2*n)$.

Value

`fidpoistol.int` returns a list with two items. The first item (`tol.limits`) is a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`fn.est`	A point estimate of the functional form of interest using the maximum likelihood estimates calculated with the inputted values of `x1`, `x2`, `n1`, and `n2`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

The second item (`fn`) simply returns the functional form specified by `FUN`.

References

Cox, D. R. (1953), Some Simple Approximate Tests for Poisson Variates, *Biometrika*, **40**, 354–360.

Krishnamoorthy, K. and Lee, M. (2010), Inference for Functions of Parameters in Discrete Distributions Based on Fiducial Approach: Binomial and Poisson Cases, *Journal of Statistical Planning and Inference*, **140**, 1182–1192.

Mathew, T. and Young, D. S. (2013), Fiducial-Based Tolerance Intervals for Some Discrete Distributions, *Computational Statistics and Data Analysis*, **61**, 38–49.

See Also

`fidbintol.int`, `fidnegbintol.int`

Examples

```
## 95%/99% 1-sided and 2-sided tolerance intervals for
## the ratio of two Poisson rates.

set.seed(100)

lambda1 <- 10
lambda2 <- 2
n1 <- 3000
n2 <- 3250
x1 <- rpois(1, n1 * lambda1)
x2 <- rpois(1, n2 * lambda2)
fun.ti <- function(x, y) x / y

fidpoistol.int(x1, x2, n1, n2, m1 = 2000, m2 = 2500,
               FUN = fun.ti, alpha = 0.05, P = 0.99, side = 1)
fidpoistol.int(x1, x2, n1, n2, m1 = 2000, m2 = 2500,
               FUN = fun.ti, alpha = 0.05, P = 0.99, side = 2)
```

`gamtol.int` *Gamma (or Log-Gamma) Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to either a gamma distribution or log-gamma distribution.

Usage

```
gamtol.int(x, alpha = 0.05, P = 0.99, side = 1,
            method = c("HE", "HE2", "WBE", "ELL", "KM", "EXACT",
            "OCT"), m = 50, log.gamma = FALSE)
```

Arguments

`x`	A vector of data which is distributed according to either a gamma distribution or a log-gamma distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. `"HE"` is the Howe method and is often viewed as being extremely accurate, even

	for small sample sizes. "HE2" is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. "WBE" is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. "ELL" is the Ellison correction to the Weissberg-Beatty method when f is appreciably larger than n^2. A warning message is displayed if f is not larger than n^2. "KM" is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. "EXACT" computes the k-factor exactly by finding the integral solution to the problem via the integrate function. Note the computation time of this method is largely determined by m. "OCT" is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
m	The maximum number of subintervals to be used in the integrate function. This is necessary only for method = "EXACT" and method = "OCT". The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for method = "EXACT".
log.gamma	If TRUE, then the data is considered to be from a log-gamma distribution, in which case the output gives tolerance intervals for the log-gamma distribution. The default is FALSE.

Details

Recall that if the random variable X is distributed according to a log-gamma distribution, then the random variable $Y = ln(X)$ is distributed according to a gamma distribution.

Value

gamtol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of the population covered by this tolerance interval.
1-sided.lower	The 1-sided lower tolerance bound. This is given only if side = 1.
1-sided.upper	The 1-sided upper tolerance bound. This is given only if side = 1.
2-sided.lower	The 2-sided lower tolerance bound. This is given only if side = 2.
2-sided.upper	The 2-sided upper tolerance bound. This is given only if side = 2.

References

Krishnamoorthy, K., Mathew, T., and Mukherjee, S. (2008), Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability, *Technometrics*, **50**, 69–78.

See Also

GammaDist, K.factor

Examples

```
## 99%/99% 1-sided gamma tolerance intervals for a sample
## of size 50.

set.seed(100)
x <- rgamma(50, 0.30, scale = 2)
out <- gamtol.int(x = x, alpha = 0.01, P = 0.99, side = 1)
out

plottol(out, x, plot.type = "both", side = "upper",
        x.lab = "Gamma Data")
```

---

hypertol.int *Hypergeometric Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for hypergeometric random variables. From a sampling without replacement perspective, these limits use the proportion of units from group A (e.g., "black balls" in an urn) in a sample to bound the number of potential units drawn from group A in a future sample taken from the universe.

Usage

```
hypertol.int(x, n, N, m = NULL, alpha = 0.05, P = 0.99,
             side = 1, method = c("EX", "LS", "CC"))
```

Arguments

x	The number of units from group A in the sample. Can be a vector, in which case the sum of x is used.
n	The size of the random sample of units selected.
N	The population size.
m	The quantity of units to be sampled from the universe for a future study. If m = NULL, then the tolerance limits will be constructed assuming n for this quantity.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of units from group A in future samples of size m to be covered by this tolerance interval.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).
method	The method for calculating the lower and upper confidence bounds, which are used in the calculation of the tolerance bounds. The default method is "EX", which is an exact-based method. "LS" is the large-sample method. "CC" gives a continuity-corrected version of the large-sample method.

Value

hypertol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of units from group A in future samples of size m.
rate	The sampling rate determined by n/N.
p.hat	The proportion of units in the sample from group A, calculated by x/n.
1-sided.lower	The 1-sided lower tolerance bound. This is given only if side = 1.
1-sided.upper	The 1-sided upper tolerance bound. This is given only if side = 1.
2-sided.lower	The 2-sided lower tolerance bound. This is given only if side = 2.
2-sided.upper	The 2-sided upper tolerance bound. This is given only if side = 2.

Note

As this methodology is built using large-sample theory, if the sampling rate is less than 0.05, then a warning is generated stating that the results are not reliable. Also, compare the functionality of this procedure with the acc.samp procedure, which is to determine a minimal acceptance limit for a particular sampling plan.

References

Brown, L. D., Cai, T. T., and DasGupta, A. (2001), Interval Estimation for a Binomial Proportion, *Statistical Science*, **16**, 101–133.

Eichenberger, P., Hulliger, B., and Potterat, J. (2011), Two Measures for Sample Size Determination, *Survey Research Methods*, **5**, 27–37.

Young, D. S. (2014), Tolerance Intervals for Hypergeometric and Negative Hypergeometric Variables, *Sankhya: The Indian Journal of Statistics, Series B*, **77**(1), 114–140.

See Also

acc.samp, Hypergeometric

Examples

```
## 90%/95% 1-sided and 2-sided hypergeometric tolerance
## intervals for a future sample of 30 when the universe
## is of size 100.

hypertol.int(x = 15, n = 50, N = 100, m = 30, alpha = 0.10,
             P = 0.95, side = 1, method = "LS")
hypertol.int(x = 15, n = 50, N = 100, m = 30, alpha = 0.10,
             P = 0.95, side = 1, method = "CC")
hypertol.int(x = 15, n = 50, N = 100, m = 30, alpha = 0.10,
             P = 0.95, side = 2, method = "LS")
hypertol.int(x = 15, n = 50, N = 100, m = 30, alpha = 0.10,
             P = 0.95, side = 2, method = "CC")
```

K.factor *Estimating K-factors for Tolerance Intervals Based on Normality*

Description

Estimates k-factors for tolerance intervals based on normality.

Usage

```
K.factor(n, f = NULL, alpha = 0.05, P = 0.99, side = 1,
         method = c("HE", "HE2", "WBE", "ELL", "KM", "EXACT",
         "OCT"), m = 50)
```

Arguments

n	The (effective) sample size.
f	The number of degrees of freedom associated with calculating the estimate of the population standard deviation. If NULL, then f is taken to be n-1.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of the population to be covered by the tolerance interval.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).
method	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. "HE" is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. "HE2" is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. "WBE" is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. "ELL" is the Ellison correction to the Weissberg-Beatty method when f is appreciably larger than n^2. A warning message is displayed if f is not larger than n^2. "KM" is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. "EXACT" computes the k-factor exactly by finding the integral solution to the problem via the integrate function. Note the computation time of this method is largely determined by m. "OCT" is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
m	The maximum number of subintervals to be used in the integrate function. This is necessary only for method = "EXACT" and method = "OCT". The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for method = "EXACT".

Value

K.factor returns the k-factor for tolerance intervals based on normality with the arguments specified above.

Note

For larger sample sizes, there may be some accuracy issues with the 1-sided calculation since it depends on the noncentral t-distribution. The code is primarily intended to be used for moderate values of the noncentrality parameter. It will not be highly accurate, especially in the tails, for large values. See `TDist` for further details.

References

Ellison, B. E. (1964), On Two-Sided Tolerance Intervals for a Normal Distribution, *Annals of Mathematical Statistics*, **35**, 762–772.

Howe, W. G. (1969), Two-Sided Tolerance Limits for Normal Populations - Some Improvements, *Journal of the American Statistical Association*, **64**, 610–620.

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Odeh, R. E. and Owen, D. B. (1980), *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*, Marcel-Dekker.

Owen, D. B. (1964), Controls of Percentages in Both Tails of the Normal Distribution, *Technometrics*, **6**, 377-387.

Wald, A. and Wolfowitz, J. (1946), Tolerance Limits for a Normal Distribution, *Annals of the Mathematical Statistics*, **17**, 208–215.

Weissberg, A. and Beatty, G. (1969), Tables of Tolerance Limit Factors for Normal Distributions, *Technometrics*, **2**, 483–500.

See Also

`integrate`, `K.table`, `normtol.int`, `TDist`

Examples

```
## Showing the k-factor under the Howe, Weissberg-Beatty,
## and exact estimation methods.

K.factor(10, P = 0.95, side = 2, method = "HE")
K.factor(10, P = 0.95, side = 2, method = "WBE")
K.factor(10, P = 0.95, side = 2, method = "EXACT", m = 20)
```

`K.factor.sim` *Estimating K-factors for Simultaneous Tolerance Intervals Based on Normality*

Description

Estimates k-factors for simultaneous tolerance intervals based on normality.

Usage

```
K.factor.sim(n, l = NULL, alpha = 0.05, P = 0.99, side = 1,
          method = c("EXACT", "BONF"), m = 50)
```

Arguments

`n`	If `method = "EXACT"`, this is the sample size of each of the `l` groups. If `method = "BONF"`, then `n` can be a vector of different sample sizes for the `l` groups.
`l`	The number of normal populations for which the k-factors will be constructed simultaneously. If `NULL`, then it is taken to be the length of `n`.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by the tolerance interval.
`side`	Whether a k-factor for a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. `"EXACT"` is an exact method that can be used when all `l` groups have the same sample size. `"BONF"` is an approximate method using the Bonferroni inequality, which can be used when the `l` groups have different sample sizes.
`m`	The maximum number of subintervals to be used in the `integrate` function. This is necessary only for `method = "EXACT"`. The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for `method = "EXACT"`.

Value

`K.factor` returns the k-factor for simultaneous tolerance intervals based on normality with the arguments specified above.

Note

For larger combinations of `n` and `l` when `side = 2` and `method = "EXACT"`, the calculation can be slow. For larger sample sizes when `method = "BONF"`, there may be some accuracy issues with the 1-sided calculation since it depends on the noncentral t-distribution. The code is primarily intended to be used for moderate values of the noncentrality parameter. It will not be highly accurate, especially in the tails, for large values. See `TDist` for further details.

Thanks to Andrew Landgraf for providing the basic code for the `method = "EXACT"` procedure.

References

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Mee, R. W. (1990), Simultaneous Tolerance Intervals for Normal Populations with Common Variance, *Technometrics*, **32**, 83-92.

See Also

`integrate`, `K.factor`

Examples

```
## Reproducing part of Table B5 from Krishnamoorthy and
## Mathew (2009).

n_sizes <- c(2:20, seq(30, 100, 10))
l_sizes <- 2:10
KM_table <- sapply(1:length(l_sizes), function(i)
                   sapply(1:length(n_sizes), function(j)
                   round(K.factor.sim(n = n_sizes[j],
                   l = l_sizes[i], side=1, alpha = 0.1,
                   P = 0.9),3)))
dimnames(KM_table) <- list(n = n_sizes, l = l_sizes)
KM_table
```

K.table *Tables of K-factors for Tolerance Intervals Based on Normality*

Description

Tabulated summary of k-factors for tolerance intervals based on normality. The user can specify multiple values for each of the three inputs.

Usage

```
K.table(n, alpha, P, side = 1, f = NULL, method = c("HE",
        "HE2", "WBE", "ELL", "KM", "EXACT", "OCT"), m = 50,
        by.arg = c("n", "alpha", "P"))
```

Arguments

n	A vector of (effective) sample sizes.
alpha	The level chosen such that 1-alpha is the confidence level. Can be a vector.
P	The proportion of the population to be covered by this tolerance interval. Can be a vector.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).
f	The number of degrees of freedom associated with calculating the estimate of the population standard deviation. If NULL, then f is taken to be n-1. Only a single value can be specified for f.
method	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. "HE" is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. "HE2" is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. "WBE" is the Weissberg-Beatty method (also called the Wald-Wolfowitz

method), which performs similarly to the first Howe method for larger sample sizes. "ELL" is the Ellison correction to the Weissberg-Beatty method when f is appreciably larger than n^2. A warning message is displayed if f is not larger than n^2. "KM" is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. "EXACT" computes the k-factor exactly by finding the integral solution to the problem via the integrate function. Note the computation time of this method is largely determined by m. "OCT" is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.

m	The maximum number of subintervals to be used in the integrate function. This is necessary only for method = "EXACT" and method = "OCT". The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for method = "EXACT".
by.arg	How you would like the output organized. If by.arg = "n", then the output provides a list of matrices sorted by the values specified in n. The matrices have rows corresponding to the values specified by 1-alpha and columns corresponding to the values specified by P. If by.arg = "alpha", then the output provides a list of matrices sorted by the values specified in 1-alpha. The matrices have rows corresponding to the values specified by n and columns corresponding to the values specified by P. If by.arg = "P", then the output provides a list of matrices sorted by the values specified in P. The matrices have rows corresponding to the values specified by 1-alpha and columns corresponding to the values specified by n.

Details

The method used for estimating the k-factors is that due to Howe as it is generally viewed as more accurate than the Weissberg-Beatty method.

Value

K.table returns a list with a structure determined by the argument by.arg described above.

References

Howe, W. G. (1969), Two-Sided Tolerance Limits for Normal Populations - Some Improvements, *Journal of the American Statistical Association*, **64**, 610–620.

Weissberg, A. and Beatty, G. (1969), Tables of Tolerance Limit Factors for Normal Distributions, *Technometrics*, **2**, 483–500.

See Also

K.factor

Examples

```
## Tables generated for each value of the sample size.
```

```
K.table(n = seq(50, 100, 10), alpha = c(0.01, 0.05, 0.10),
        P = c(0.90, 0.95, 0.99), by.arg = "n")

## Tables generated for each value of the confidence level.

K.table(n = seq(50, 100, 10), alpha = c(0.01, 0.05, 0.10),
        P = c(0.90, 0.95, 0.99), by.arg = "alpha")

## Tables generated for each value of the coverage proportion.

K.table(n = seq(50, 100, 10), alpha = c(0.01, 0.05, 0.10),
        P = c(0.90, 0.95, 0.99), by.arg = "P")
```

`laptol.int` *Laplace Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to a Laplace distribution.

Usage

```
laptol.int(x, alpha = 0.05, P = 0.99, side = 1)
```

Arguments

`x`	A vector of data which is distributed according to a Laplace distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).

Value

`laptol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Bain, L. J. and Engelhardt, M. (1973), Interval Estimation for the Two Parameter Double Exponential Distribution, *Technometrics*, **15**, 875–887.

Examples

```
## First generate data from a Laplace distribution with location
## parameter 70 and scale parameter 3.

set.seed(100)
tmp <- runif(40)
x <- rep(70, 40) - sign(tmp - 0.5)*rep(3, 40)*
              log(2*ifelse(tmp < 0.5, tmp, 1-tmp))

## 95%/90% 1-sided Laplace tolerance intervals for the sample
## of size 40 generated above.

out <- laptol.int(x = x, alpha = 0.05, P = 0.90, side = 1)
out

plottol(out, x, plot.type = "hist", side = "two",
        x.lab = "Laplace Data")
```

---

logistol.int *Logistic (or Log-Logistic) Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to a logistic or log-logistic distribution.

Usage

```
logistol.int(x, alpha = 0.05, P = 0.99, log.log = FALSE,
              side = 1)
```

Arguments

`x`	A vector of data which is distributed according to a logistic or log-logistic distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`log.log`	If `TRUE`, then the data is considered to be from a log-logistic distribution, in which case the output gives tolerance intervals for the log-logistic distribution. The default is `FALSE`.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).

Details

Recall that if the random variable X is distributed according to a log-logistic distribution, then the random variable $Y = ln(X)$ is distributed according to a logistic distribution.

Value

`logistol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side` = `1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side` = `1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side` = `2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side` = `2`.

References

Balakrishnan, N. (1992), *Handbook of the Logistic Distribution*, Marcel Dekker, Inc.

Hall, I. J. (1975), One-Sided Tolerance Limits for a Logistic Distribution Based on Censored Samples, *Biometrics*, **31**, 873–880.

See Also

`Logistic`

Examples

```
## 90%/95% 1-sided logistic tolerance intervals for a sample
## of size 20.

set.seed(100)
x <- rlogis(20, 5, 1)
out <- logistol.int(x = x, alpha = 0.10, P = 0.95,
                    log.log = FALSE, side = 1)
out

plottol(out, x, plot.type = "control", side = "two",
        x.lab = "Logistic Data")
```

`mvregtol.region` *Multivariate (Multiple) Linear Regression Tolerance Regions*

Description

Determines the appropriate tolerance factor for computing multivariate (multiple) linear regression tolerance regions based on Monte Carlo simulation.

Usage

```
mvregtol.region(mvreg, new.x = NULL, alpha = 0.05, P = 0.99,
                B = 1000)
```

Arguments

`mvreg`	A multivariate (multiple) linear regression fit, having class `mlm`.
`new.x`	An optional data frame of new values for which to approximate k-factors. This must be a data frame with named columns that match those in the data frame used for the `mvreg` fitted object.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance region.
`B`	The number of iterations used for the Monte Carlo algorithm which determines the tolerance factor. The number of iterations should be at least as large as the default value of 1000.

Details

A basic sketch of how the algorithm works is as follows:

(1) Generate independent chi-square random variables and Wishart random matrices.

(2) Compute the eigenvalues of the randomly generated Wishart matrices.

(3) Iterate the above steps to generate a set of `B` sample values such that the `100(1-alpha)`-th percentile is an approximate tolerance factor.

Value

`mvregtol.region` returns a matrix where the first column is the k-factor, the next `q` columns are the estimated responses from the least squares fit, and the final `m` columns are the predictor values. The first `n` rows of the matrix pertain to the raw data as specified by `y` and `x`. If values for `new.x` are specified, then there is one additional row appended to this output for each row in the matrix `new.x`.

Note

As of tolerance version 2.0.0, the arguments to this function have changed. This function no longer depends on inputted `y` and `x` matrices or an `int` argument. Instead, the function requires `mvreg`, which is of class "mlm", and provides all of the necessary components for the way the output is formatted. Also, `new.x` must now be a data frame with columns matching those from the data frame used in the `mvreg` fitted object.

References

Anderson, T. W. (2003) *An Introduction to Multivariate Statistical Analysis*, Third Edition, Wiley.

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Krishnamoorthy, K. and Mondal, S. (2008), Tolerance Factors in Multiple and Multivariate Linear Regressions, *Communications in Statistics - Simulation and Computation*, **37**, 546–559.

Examples

```
## 95%/95% multivariate regression tolerance factors using
## a fertilizer data set presented in Anderson (2003, p. 374).

grain <- c(40, 17, 9, 15, 6, 12, 5, 9)
straw <- c(53, 19, 10, 29, 13, 27, 19, 30)
fert <- c(24, 11, 5, 12, 7, 14, 11, 18)
DF <- data.frame(grain,straw,fert)
new.x <- data.frame(fert = c(10, 15, 20))
mvreg <- lm(cbind(grain, straw) ~ fert + I(fert^2), data = DF)

set.seed(100)
out <- mvregtol.region(mvreg, new.x = new.x, alpha = 0.05,
                       P = 0.95, B = 5000)
out
```

`mvtol.region` *Multivariate Normal Tolerance Regions*

Description

Determines the appropriate tolerance factor for computing multivariate normal tolerance regions based on Monte Carlo methods or other approximations.

Usage

```
mvtol.region(x, alpha = 0.05, P = 0.99, B = 1000, M = 1000,
             method = c("KM", "AM", "GM", "HM", "MHM", "V11",
             "HM.V11", "MC"))
```

Arguments

`x`	An nxp matrix of data assumed to be drawn from a p-dimensional multivariate normal distribution. `n` pertains to the sample size.
`alpha`	The level chosen such that `1-alpha` is the confidence level. A vector of `alpha` values may be specified.
`P`	The proportion of the population to be covered by this tolerance region. A vector of `P` values may be specified.

`B`	The number of iterations used for the Monte Carlo algorithms (i.e., when `method = "KM"` or `"MC"`), which determines the tolerance factor. The number of iterations should be at least as large as the default value of 1000.
`M`	The number of iterations used for the inner loop of the Monte Carlo algorithm specified through `method = "MC"`. The number of iterations should be at least as large as the default value of 1000. Note that this is not required for `method = "KM"` since that algorithm handles the eigenvalues differently in the estimation of the tolerance factor.
`method`	The method for estimating the tolerance factors. `"KM"` is the Krishnamoorthy-Mondal method, which is the method implemented in previous versions of the `tolerance` package. It is one of the more accurate methods available. `"AM"` is an approximation method based on the arithmetic mean. `"GM"` is an approximation method based on the geometric mean. `"HM"` is an approximation method based on the harmonic mean. `"MHM"` is a modified approach based on the harmonic mean. `"V11"` is a method that utilizes a certain partitioning of a Wishart random matrix for deriving an approximate tolerance factor. `"HM.V11"` is a hybrid method of the `"HM"` and `"V11"` methods. `"MC"` is a simple Monte Carlo approach to estimating the tolerance factor, which is computationally expensive as the values of `B` and `M` increase.

Details

All of the methods are outlined in the references that we provided. In practice, we recommend using the Krishnamoorthy-Mondal approach. A basic sketch of how the Krishnamoorthy-Mondal algorithm works is as follows:

(1) Generate independent chi-square random variables and Wishart random matrices.

(2) Compute the eigenvalues of the randomly generated Wishart matrices.

(3) Iterate the above steps to generate a set of `B` sample values such that the `100(1-alpha)`-th percentile is an approximate tolerance factor.

Value

`mvtol.region` returns a matrix where the rows pertain to each confidence level `1-alpha` specified and the columns pertain to each proportion level `P` specified.

References

Krishnamoorthy, K. and Mathew, T. (1999), Comparison of Approximation Methods for Computing Tolerance Factors for a Multivariate Normal Population, *Technometrics*, **41**, 234–249.

Krishnamoorthy, K. and Mondal, S. (2006), Improved Tolerance Factors for Multivariate Normal Distributions, *Communications in Statistics - Simulation and Computation*, **35**, 461–478.

Examples

```
## 90%/90% bivariate normal tolerance region.

set.seed(100)
```

```
x1 <- rnorm(100, 0, 0.2)
x2 <- rnorm(100, 0, 0.5)
x <- cbind(x1, x2)

out1 <- mvtol.region(x = x, alpha = 0.10, P = 0.90, B = 1000,
                     method = "KM")
out1
plottol(out1, x)

## 90%/90% trivariate normal tolerance region.

set.seed(100)
x1 <- rnorm(100, 0, 0.2)
x2 <- rnorm(100, 0, 0.5)
x3 <- rnorm(100, 5, 1)
x <- cbind(x1, x2, x3)
mvtol.region(x = x, alpha = c(0.10, 0.05, 0.01),
             P = c(0.90, 0.95, 0.99), B = 1000, method = "KM")

out2 <- mvtol.region(x = x, alpha = 0.10, P = 0.90, B = 1000,
                     method = "KM")
out2
plottol(out2, x)
```

`negbintol.int` *Negative Binomial Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for negative binomial random variables. From a statistical quality control perspective, these limits use the number of failures that occur to reach `n` successes to bound the number of failures for a specified amount of future successes (`m`).

Usage

```
negbintol.int(x, n, m = NULL, alpha = 0.05, P = 0.99,
              side = 1, method = c("LS", "WU", "CB",
              "CS", "SC", "LR", "SP", "CC"))
```

Arguments

`x`	The total number of failures that occur from a sample of size `n`. Can be a vector of length `n`, in which case the sum of `x` is computed.
`n`	The target number of successes (sometimes called size) for each trial.
`m`	The target number of successes in a future lot for which the tolerance limits will be calculated. If `m = NULL`, then the tolerance limits will be constructed assuming `n` for the target number of future successes.
`alpha`	The level chosen such that `1-alpha` is the confidence level.

`P`	The proportion of the defective (or acceptable) units in future samples of size `m` to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the lower and upper confidence bounds, which are used in the calculation of the tolerance bounds. The default method is `"LS"`, which is the large-sample method based on the MLE. `"WU"` is a Wald-type interval based on the UMVUE of the negative binomial proportion. `"CB"` is the Casella-Berger exact method. `"CS"` is a method based on chi-square percentiles. `"SC"` is the score method. `"LR"` is a likelihood ratio-based method. `"SP"` is a method using a saddlepoint approximation for the confidence intervals. `"CC"` gives a continuity-corrected version of the large-sample method and is appropriate when `n` is large. More information on these methods can be found in the "References".

Details

This function takes the approach for Poisson and binomial random variables developed in Hahn and Chandra (1981) and applies it to the negative binomial case.

Value

`negbintol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of defective (or acceptable) units in future samples of size `m`.
`pi.hat`	The probability of success in each trial, calculated by `n/(n+x)`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

Note

Recall that the geometric distribution is the negative binomial distribution where the size is 1. Therefore, the case when `n = m = 1` will provide tolerance limits for a geometric distribution.

References

Casella, G. and Berger, R. L. (1990), *Statistical Inference*, Duxbury Press.

Hahn, G. J. and Chandra, R. (1981), Tolerance Intervals for Poisson and Binomial Variables, *Journal of Quality Technology*, **13**, 100–110.

Tian, M., Tang, M. L., Ng, H. K. T., and Chan, P. S. (2009), A Comparative Study of Confidence Intervals for Negative Binomial Proportions, *Journal of Statistical Computation and Simulation*, **79**, 241–249.

Young, D. S. (2014), A Procedure for Approximate Negative Binomial Tolerance Intervals, *Journal of Statistical Computation and Simulation*, **84**, 438–450.

See Also

NegBinomial, umatol.int

Examples

```
## Comparison of 95%/99% 1-sided tolerance limits with
## 50 failures before 10 successes are reached.

negbintol.int(x = 50, n = 10, side = 1, method = "LS")
negbintol.int(x = 50, n = 10, side = 1, method = "WU")
negbintol.int(x = 50, n = 10, side = 1, method = "CB")
negbintol.int(x = 50, n = 10, side = 1, method = "CS")
negbintol.int(x = 50, n = 10, side = 1, method = "SC")
negbintol.int(x = 50, n = 10, side = 1, method = "LR")
negbintol.int(x = 50, n = 10, side = 1, method = "SP")
negbintol.int(x = 50, n = 10, side = 1, method = "CC")

## 95%/99% 1-sided tolerance limits and 2-sided tolerance
## interval for the same setting above, but when we are
## interested in a future experiment that requires 20 successes
## be reached for each trial.

negbintol.int(x = 50, n = 10, m = 20, side = 1)
negbintol.int(x = 50, n = 10, m = 20, side = 2)
```

NegHypergeometric *The Negative Hypergeometric Distribution*

Description

Density, distribution function, quantile function, and random generation for the negative hypergeometric distribution.

Usage

```
dnhyper(x, m, n, k, log = FALSE)
pnhyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qnhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rnhyper(nn, m, n, k)
```

Arguments

x,q	Vector of quantiles representing the number of trials until k successes have occurred (e.g., until k white balls have been drawn from an urn without replacement).
m	The number of successes in the population (e.g., the number of white balls in the urn).

`n`	The population size (e.g., the total number of balls in the urn).
`k`	The number of successes (e.g., white balls) to achieve with the sample.
`p`	Vector of probabilities, which must be between 0 and 1.
`nn`	The number of observations. If `length>1`, then the length is taken to be the number required.
`log,log.p`	Logical vectors. If `TRUE`, then probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.

Details

A negative hypergeometric distribution (sometimes called the inverse hypergeometric distribution) models the total number of trials until `k` successes occur. Compare this to the negative binomial distribution, which models the number of failures that occur until a specified number of successes has been reached. The negative hypergeometric distribution has density

$$p(x) = \frac{\binom{x-1}{k-1}\binom{n-x}{m-k}}{\binom{n}{m}}$$

for $x = k, k+1, ..., n-m+k$.

Value

`dnhyper` gives the density, `pnhyper` gives the distribution function, `qnhyper` gives the quantile function, and `rnhyper` generates random deviates.

Invalid arguments will return value `NaN`, with a warning.

References

Wilks, S. S. (1963), *Mathematical Statistics*, Wiley.

See Also

`runif` and `.Random.seed` about random number generation.

Examples

```
## Randomly generated data from the negative hypergeometric
## distribution.

set.seed(100)
x <- rnhyper(nn = 1000, m = 15, n = 40, k = 10)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 = sort(x)
y <- dnhyper(x = x.1, m = 15, n = 40, k = 10)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pnhyper(q = x.1, m = 15, n = 40, k = 10),
     type = "l", xlab = "x", ylab = "Cumulative Probabilities")
```

```
qnhyper(p = 0.20, m = 15, n = 40, k = 10, lower.tail = FALSE)
qnhyper(p = 0.80, m = 15, n = 40, k = 10)
```

---

neghypertol.int *Negative Hypergeometric Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for negative hypergeometric random variables. When sampling without replacement, these limits are on the total number of expected draws in a future sample in order to achieve a certain number from group A (e.g., "black balls" in an urn).

Usage

```
neghypertol.int(x, n, N, m = NULL, alpha = 0.05, P = 0.99,
                side = 1, method = c("EX", "LS", "CC"))
```

Arguments

x	The number of units drawn in order to achieve n successes. Can be a vector, in which case the sum of x is used.
n	The target number of successes in the sample drawn (e.g., the number of "black balls" you are to draw in the sample).
N	The population size (e.g., the total number of balls in the urn).
m	The target number of successes to be sampled from the universe for a future study. If m = NULL, then the tolerance limits will be constructed assuming n for this quantity.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of units from group A in future samples of size m to be covered by this tolerance interval.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).
method	The method for calculating the lower and upper confidence bounds, which are used in the calculation of the tolerance bounds. The default method is "EX", which is an exact-based method. "LS" is the large-sample method. "CC" gives a continuity-corrected version of the large-sample method.

Value

neghypertol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of units from group A in future samples of size m.
rate	The sampling rate determined by x/N.

`p.hat`	The proportion of units in the sample from group A, calculated by `n/x`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

Note

As this methodology is built using large-sample theory, if the sampling rate is less than 0.05, then a warning is generated stating that the results are not reliable.

References

Khan, R. A. (1994), A Note on the Generating Function of a Negative Hypergeometric Distribution, *Sankhya: The Indian Journal of Statistics, Series B*, **56**, 309–313.

Young, D. S. (2014), Tolerance Intervals for Hypergeometric and Negative Hypergeometric Variables, *Sankhya: The Indian Journal of Statistics, Series B*, **77**(1), 114–140.

See Also

`acc.samp`, `NegHypergeometric`

Examples

```
## 90%/95% 2-sided negative hypergeometric tolerance
## intervals for a future number of 20 successes when
## the universe is of size 100.  The estimates are
## based on having drawn 50 in another sample to achieve
## 20 successes.

neghypertol.int(50, 20, 100, m = 20, alpha = 0.05,
                P = 0.95, side = 2, method = "LS")
```

`nlregtol.int` *Nonlinear Regression Tolerance Bounds, Version 2*

Description

Provides 1-sided or 2-sided nonlinear regression tolerance bounds.

Usage

```
nlregtol.int(formula, xy.data = data.frame(), x.new = NULL,
             side = 1, alpha = 0.05, P = 0.99, maxiter = 50,
             new = FALSE, ...)
```

Arguments

`formula`	A nonlinear model formula including variables and parameters.
`xy.data`	A data frame in which to evaluate the formulas in `formula`. The first column of `xy.data` must be the response variable.
`x.new`	Any new levels of the predictor(s) for which to report the tolerance bounds. The number of columns must be 1 less than the number of columns for `xy.data`.
`side`	Whether a 1-sided or 2-sided tolerance bound is required (determined by `side = 1` or `side = 2`, respectively).
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by the tolerance bound(s).
`maxiter`	A positive integer specifying the maximum number of iterations that the nonlinear least squares routine (`nls`) should run.
`new`	When `new = TRUE`, the function shows updated version of outcomes.
`...`	Optional arguments passed to `nls` when estimating the nonlinear regression equation.

Details

It is highly recommended that the user specify starting values for the `nls` routine.

Value

`npregtol.int2` returns a list with items:

`tol`	Data frame of original response varible `y`, fitted values `y.hat`, corresponding tolerance limits. The data frame is ordered by `X` values. If there are new data for prediction, predicted values are attached in the end.
`alpha.P.side`	Model specifications of critical level, content level and side.
`reg.type`	Type of regression model.
`model`	The linear regression model fitted.
`newdata`	X values of new data for prediction.
`xy.data.original`	Original data frame

References

Wallis, W. A. (1951), Tolerance Intervals for Linear Regression, in *Second Berkeley Symposium on Mathematical Statistics and Probability*, ed. J. Neyman, Berkeley: University of CA Press, 43–51.

Young, D. S. (2013), Regression Tolerance Intervals, *Communications in Statistics - Simulation and Computation*, **42**, 2040–2055.

See Also

`nls`, `nlregtol.int`

Examples

```
## 95%/95% 2-sided nonlinear regression tolerance bounds
## for a sample of size 50.
set.seed(100)
x <- runif(50, 5, 45)
f1 <- function(x, b1, b2) b1 + (0.49 - b1)*exp(-b2*(x - 8)) +
  rnorm(50, sd = 0.01)
y <- f1(x, 0.39, 0.11)
formula <- as.formula(y ~ b1 + (0.49 - b1)*exp(-b2*(x - 8)))
out1 <- nlregtol.int(formula = formula,
                     xy.data = data.frame(cbind(y, x)),
                     x.new=c(10,20), side = 2,
                     alpha = 0.05, P = 0.95 , new = TRUE)
out1
#########
set.seed(100)
x1 <- runif(50, 5, 45)
x2 <- rnorm(50, 0, 10)
f1 <- function(x1, x2, b1, b2) {(0.49 - b1)*exp(-b2*(x1 + x2 - 8)) +
    rnorm(50, sd = 0.01)}
y <- f1(x1 , x2 , 0.25 , 0.39)
formula <- as.formula(y ~ (0.49 - b1)*exp(-b2*(x1 + x2 - 8)))
out2 <- nlregtol.int(formula = formula,
                     xy.data = data.frame(cbind(y, x1 , x2)),
                     x.new=cbind(c(10,20) , c(47 , 53)), side = 2,
                     alpha = 0.05, P = 0.95 , new = TRUE)
out2
```

---

norm.OC — *Operating Characteristic (OC) Curves for K-Factors for Tolerance Intervals Based on Normality*

Description

Provides OC-type curves to illustrate how values of the k-factors for normal tolerance intervals, confidence levels, and content levels change as a function of the sample size.

Usage

```
norm.OC(k = NULL, alpha = NULL, P = NULL, n, side = 1,
        method = c("HE", "HE2", "WBE", "ELL", "KM", "EXACT",
        "OCT"), m = 50)
```

Arguments

k	If wanting OC curves where the confidence level or content level is on the y-axis, then a single positive value of k must be specified. This would be the target k-factor for the desired tolerance interval. If k = NULL, then OC curves will be

	constructed where the k-factor value is found for given levels of `alpha`, `P`, and `n`.
`alpha`	The set of levels chosen such that `1-alpha` are confidence levels. If wanting OC curves where the content level is being calculated, then each curve will correspond to a level in the set of `alpha`. If a set of `P` values is specified, then OC curves will be constructed where the k-factor is found and each curve will correspond to each combination of `alpha` and `P`. If `alpha = NULL`, then OC curves will be constructed to find the confidence level for given levels of `k`, `P`, and `n`.
`P`	The set of content levels to be considered. If wanting OC curves where the confidence level is being calculated, then each curve will correspond to a level in the set of `P`. If a set of `alpha` values is specified, then OC curves will be constructed where the k-factor is found and each curve will correspond to each combination of `alpha` and `P`. If `P = NULL`, then OC curves will be constructed to find the content level for given levels of `k`, `alpha`, and `n`.
`n`	A sequence of sample sizes to consider. This must be a vector of at least length 2 since all OC curves are constructed as functions of `n`.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. `"HE"` is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. `"HE2"` is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. `"WBE"` is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. `"ELL"` is the Ellison correction to the Weissberg-Beatty method when `f` is appreciably larger than `n^2`. A warning message is displayed if `f` is not larger than `n^2`. `"KM"` is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. `"EXACT"` computes the k-factor exactly by finding the integral solution to the problem via the `integrate` function. Note the computation time of this method is largely determined by `m`. `"OCT"` is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
`m`	The maximum number of subintervals to be used in the `integrate` function, which is used for the underlying exact method for calculating the normal tolerance intervals.

Value

`norm.OC` returns a figure with the OC curves constructed using the specifications in the arguments.

References

Young, D. S. (2016), Normal Tolerance Interval Procedures in the tolerance Package, *The R Journal*, **8**, 200–212.

See Also

K.factor, normtol.int

Examples

```
## The three types of OC-curves that can be constructed
## with the norm.OC function.

norm.OC(k = 4, alpha = NULL, P = c(0.90, 0.95, 0.99),
        n = 10:20, side = 1)

norm.OC(k = 4, alpha = c(0.01, 0.05, 0.10), P = NULL,
        n = 10:20, side = 1)

norm.OC(k = NULL, P = c(0.90, 0.95, 0.99),
        alpha=c(0.01,0.05,0.10), n = 10:20, side = 1)
```

---

norm.ss *Sample Size Determination for Normal Tolerance Intervals*

Description

Provides minimum sample sizes for a future sample size when constructing normal tolerance intervals. Various strategies are available for determining the sample size, including strategies that incorporate known specification limits.

Usage

```
norm.ss(x = NULL, alpha = 0.05, P = 0.99, delta = NULL,
        P.prime = NULL, side = 1, m = 50, spec = c(NA, NA),
        hyper.par = list(mu.0 = NULL, sig2.0 = NULL,
        m.0 = NULL, n.0 = NULL), method = c("DIR",
        "FW", "YGZO"))
```

Arguments

x	A vector of current data that is distributed according to a normal distribution. This is only required for method = "YGZO".
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of the population to be covered by this tolerance interval.
delta	The precision measure for the future tolerance interval as specified under the Faulkenberry-Weeks method.
P.prime	The proportion of the population (greater than P) such that the tolerance interval of interest will only exceed P.prime by the probability given by delta.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).

`m`	The maximum number of subintervals to be used in the `integrate` function, which is used for the underlying exact method for calculating the normal tolerance intervals.
`spec`	A vector of length 2 given known specification limits. These are required when `method = "DIR"` or `method = "YGZO"`. By default, the values are `NA`. The two elements of the vector are for the lower and upper specification limits, respectively. If `side = 1`, then only one of the specification limits must be specified. If `side = 2`, then both specification limits must be specified.
`hyper.par`	Necessary parameter values for the different methods. If `method = "DIR"` or `method = "YGZO"`, then `mu.0` and `sig2.0` must be specified, which correspond to the assumed population mean and variance of the underlying normal distribution, which further pertains to the historical data for `method = "YGZO"`. If `method = "YGZO"` and the sample size is to be determined using Bayesian normal tolerance intervals, then this is a required list consisting of the hyperparameters for the conjugate prior – the hyperparameters for the mean (`mu.0` and `n.0`) and the hyperparameters for the variance (`sig2.0` and `m.0`).
`method`	The method for performing the sample size determination. `"DIR"` is the direct method (intended as a simple calculation for planning purposes) where the mean and standard deviation are taken as truth and the sample size is determined with respect to the given specification limits. `"FW"` is for the traditional Faulkenberry-Weeks approach for sample size determination. `"YGZO"` is for the Young-Gordon-Zhu-Olin approach, which incorporates historical data and specification limits for determining the value of `delta` and/or `P.prime` in the Faulkenberry-Weeks approach. Note that for `"YGZO"`, at least one of `delta` and `P.prime` must be `NULL`.

Value

`norm.ss` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`delta`	The user-specified or calculated precision measure. Not returned if `method = "DIR"`.
`P.prime`	The user-specified or calculated closeness measure. Not returned if `method = "DIR"`.
`n`	The minimum sample size determined using the conditions specified for this function.

References

Faulkenberry, G. D. and Weeks, D. L. (1968), Sample Size Determination for Tolerance Limits, *Technometrics*, **10**, 343–348.

Young, D. S., Gordon, C. M., Zhu, S., and Olin, B. D. (2016), Sample Size Determination Strategies for Normal Tolerance Intervals Using Historical Data, *Quality Engineering*, **28**, 337–351.

See Also

bayesnormtol.int, Normal, normtol.int

Examples

```
## Sample size determination for 95%/95% 2-sided normal
## tolerance intervals using the direct method.

set.seed(100)
norm.ss(alpha = 0.05, P = 0.95, side = 2, spec = c(-3, 3),
        method = "DIR", hyper.par = list(mu.0 = 0,
        sig2.0 = 1))
```

---

normtol.int *Normal (or Log-Normal) Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to either a normal distribution or log-normal distribution.

Usage

```
normtol.int(x, alpha = 0.05, P = 0.99, side = 1,
            method = c("HE", "HE2", "WBE", "ELL", "KM",
            "EXACT", "OCT"), m = 50, log.norm = FALSE)
```

Arguments

x	A vector of data which is distributed according to either a normal distribution or a log-normal distribution.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of the population to be covered by this tolerance interval.
side	Whether a 1-sided or 2-sided tolerance interval is required (determined by side = 1 or side = 2, respectively).
method	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. "HE" is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. "HE2" is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. "WBE" is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. "ELL" is the Ellison correction to the Weissberg-Beatty method when f is appreciably larger than n^2. A warning message is displayed if f is not larger

	than `n^2`. `"KM"` is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. `"EXACT"` computes the k-factor exactly by finding the integral solution to the problem via the `integrate` function. Note the computation time of this method is largely determined by `m`. `"OCT"` is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
`m`	The maximum number of subintervals to be used in the `integrate` function. This is necessary only for `method = "EXACT"` and `method = "OCT"`. The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for `method = "EXACT"`.
`log.norm`	If `TRUE`, then the data is considered to be from a log-normal distribution, in which case the output gives tolerance intervals for the log-normal distribution. The default is `FALSE`.

Details

Recall that if the random variable X is distributed according to a log-normal distribution, then the random variable $Y = ln(X)$ is distributed according to a normal distribution.

Value

`normtol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`x.bar`	The sample mean.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Howe, W. G. (1969), Two-Sided Tolerance Limits for Normal Populations - Some Improvements, *Journal of the American Statistical Association*, **64**, 610–620.

Wald, A. and Wolfowitz, J. (1946), Tolerance Limits for a Normal Distribution, *Annals of Mathematical Statistics*, **17**, 208–215.

Weissberg, A. and Beatty, G. (1969), Tables of Tolerance Limit Factors for Normal Distributions, *Technometrics*, **2**, 483–500.

See Also

`Normal`, `K.factor`

Examples

```
## 95%/95% 2-sided normal tolerance intervals for a sample
## of size 100.

set.seed(100)
x <- rnorm(100, 0, 0.2)
out <- normtol.int(x = x, alpha = 0.05, P = 0.95, side = 2,
                   method = "HE", log.norm = FALSE)
out

plottol(out, x, plot.type = "both", side = "two",
        x.lab = "Normal Data")
```

`np.order` *Sample Size Determination for Tolerance Limits Based on Order Statistics*

Description

For given values of `m`, `alpha`, and `P`, this function solves the necessary sample size such that the `r`-th (or `(n-s+1)`-th) order statistic is the `[100(1-alpha)%, 100(P)%]` lower (or upper) tolerance limit (see the Details section below for further explanation). This function can also report all combinations of order statistics for 2-sided intervals.

Usage

```
np.order(m, alpha = 0.05, P = 0.99, indices = FALSE)
```

Arguments

`m`	See the Details section below for how `m` is defined.
`alpha`	1 minus the confidence level attained when it is desired to cover a proportion `P` of the population with the order statistics.
`P`	The proportion of the population to be covered with confidence `1-alpha` with the order statistics.
`indices`	An optional argument to report all combinations of order statistics indices for the upper and lower limits of the 2-sided intervals. Note that this can only be calculated when `m>1`.

Details

For the 1-sided tolerance limits, `m=s+r` such that the probability is at least `1-alpha` that at least the proportion `P` of the population is below the `(n-s+1)`-th order statistic for the upper limit or above the `r`-th order statistic for the lower limit. This means for the 1-sided upper limit that `r=1`, while for the 1-sided lower limit it means that `s=1`. For the 2-sided tolerance intervals, `m=s+r` such that the probability is at least `1-alpha` that at least the proportion `P` of the population is between the `r`-th and `(n-s+1)`-th order statistics. Thus, all combinations of r>0 and s>0 such that `m=s+r` are considered.

Value

If `indices = FALSE`, then a single number is returned for the necessary sample size such that the `r`-th (or `(n-s+1)`-th) order statistic is the `[100(1-alpha)%, 100(P)%]` lower (or upper) tolerance limit. If `indices = TRUE`, then a list is returned with a single number for the necessary sample size and a matrix with 2 columns where each row gives the pairs of indices for the order statistics for all permissible `[100(1-alpha)%, 100(P)%]` 2-sided tolerance intervals.

References

Hanson, D. L. and Owen, D. B. (1963), Distribution-Free Tolerance Limits Elimination of the Requirement That Cumulative Distribution Functions Be Continuous, *Technometrics*, **5**, 518–522.

Scheffe, H. and Tukey, J. W. (1945), Non-Parametric Estimation I. Validation of Order Statistics, *Annals of Mathematical Statistics*, **16**, 187–192.

See Also

`nptol.int`

Examples

```
## Only requesting the sample size.

np.order(m = 5, alpha = 0.05, P = 0.95)

## Requesting the order statistics indices as well.

np.order(m = 5, alpha = 0.05, P = 0.95, indices = TRUE)
```

`npbetol.int` *Nonparametric Beta-Expectation Tolerance Intervals*

Description

Provides 1-sided or 2-sided nonparametric (i.e., distribution-free) beta-expectation tolerance intervals for any continuous data set. These are equivalent to nonparametric prediction intervals based on order statistics.

Usage

```
npbetol.int(x, Beta = 0.95, side = 1, upper = NULL, lower = NULL)
```

Arguments

`x`	A vector of data which no distributional assumptions are made. The data is only assumed to come from a continuous distribution.
`Beta`	The confidence level.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`upper`	The upper bound of the data. When `NULL`, then the maximum of `x` is used.
`lower`	The lower bound of the data. When `NULL`, then the minimum of `x` is used.

Value

`nptol.int` returns a data frame with items:

`Beta`	The specified confidence level.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Beran, R. and Hall, P. (1993), Interpolated Nonparametric Prediction Intervals and Confidence Intervals, *Journal of the Royal Statistical Society, Series B*, **55**, 643–652.

See Also

`distfree.est`, `npregtol.int`, `nptol.int`

Examples

```
## Nonparametric 90%-expectation tolerance intervals
## for a sample of size 100.

set.seed(100)
x <- rexp(100, 5)
out <- npbetol.int(x = x, Beta = 0.90, side = 2,
                   upper = NULL, lower = NULL)
out
```

npmvtol.region *Nonparametric Multivariate Hyperrectangular Tolerance Regions*

Description

Provides depth-based multivariate central or semi-space nonparametric tolerance regions. These can be calculated for any continuous multivariate data set. Either (P, 1-alpha) tolerance regions or beta-expectation tolerance regions can be specified.

Usage

```
npmvtol.region(x, alpha = NULL, P = NULL, Beta = NULL, depth.fn,
               adjust = c("no", "floor", "ceiling"),
               type = c("central", "semispace"),
               semi.order = list(lower = NULL, center = NULL, upper = NULL),
               L = -Inf, U = Inf, ...)
```

Arguments

`x`	An `nxp` matrix of data assumed to be drawn from a `p`-dimensional multivariate distribution. `n` pertains to the sample size.
`alpha`	The level chosen such that `1-alpha` is the confidence level. Note that if a (P, 1-alpha) tolerance region is required, then both `alpha` and `P` must be specified, but `Beta` must be set to `NULL`.
`P`	The proportion of the population to be covered by this tolerance interval. Note that if a (P, 1-alpha) tolerance region is required, then both `alpha` and `P` must be specified, but `Beta` must be set to `NULL`.
`Beta`	The confidence level for a beta-expectation tolerance region. Note that if a beta-expectation tolerance region is required, then `Beta` must be specified, but both `alpha` and `P` must be set to `NULL`.
`depth.fn`	The data depth function used to perform the ordering of the multivariate data. Thus function must be coded in such a way that the first argument is multivariate data for which to calculate the depth values and the second argument is the original multivariate sample, `x`. For the purpose of this tolerance region calculation, these two arguments should both be the original multivariate sample.
`adjust`	Whether an adjustment should be made during an intermediate calculation for determining the number of points that need to be included in the multivariate region. If `adjust = "no"`, the default, then no adjustment is made during the intermediate calculation. If `adjust = "floor"`, then the intermediate calculation is rounded down to the next nearest integer. If `adjust = "ceiling"`, then the intermediate calculation is rounded up to the next nearest integer.
`type`	The type of multivariate hyperrectangular region to calculate. If `type = "central"`, then two-sided intervals are reported for each dimension of the data `x`. If `type = "semispace"`, then a combination of one-sided intervals and two-sided intervals are reported for the dimensions of `x`. Which interval is calculated for each dimension in this latter setting is dictated by the `semi.order` argument.

semi.order	If type = "semispace", then this argument must be specified. This argument is a list of length 3, such that each element gives the indices of the dimensions of x for which the type of interval should be calculated. Indices specified for the element of lower will return one-sided lower limits for those dimensions, indices specified for the element of center will return two-sided intervals for those dimensions, and indices specified for the element of upper will return one-sided upper limits for those dimensions.
L	If type = "semispace", these are the lower limits for any dimensions for which one requests one-sided upper limits.
U	If type = "semispace", these are the upper limits for any dimensions for which one requests one-sided lower limits.
...	Additional arguments passed to the depth.fn function.

Value

npmvtol.region returns a px2 matrix where the columns give the lower and upper limits, respectively, of the multivariate hyperrectangular tolerance region.

References

Young, D. S. and Mathew, T. (2020), Nonparametric Hyperrectangular Tolerance and Prediction Regions for Setting Multivariate Reference Regions in Laboratory Medicine, *Statistical Methods in Medical Research*, **29**, 3569–3585.

See Also

distfree.est, mvtol.region, npregtol.int

Examples

```
## 90%/95% semi-space tolerance region for a sample
## of size 20 generated from a multivariate normal
## distribution. The mdepth function below is not
## a true depth function, but used only for
## illustrative purposes.

mdepth <- function(pts, x){
          mahalanobis(pts, center = rep(0, 3),
                      cov = diag(1, 3))
          }

set.seed(100)
x <- cbind(rnorm(100), rnorm(100), rnorm(100))
out <- npmvtol.region(x = x, alpha = 0.10, P = 0.95, depth.fn = mdepth,
                      type = "semispace", semi.order = list(lower = 2,
                      center = 3, upper = 1))
out
```

npregtol.int *Nonparametric Regression Tolerance Bounds*

Description

Provides 1-sided or 2-sided nonparametric regression tolerance bounds.

Usage

```
npregtol.int(x, y, y.hat, side = 1, alpha = 0.05, P = 0.99,
             method = c("WILKS", "WALD", "HM"), upper = NULL,
             lower = NULL, new = FALSE)
```

Arguments

`x`	A vector of values for the predictor variable. Currently, this function is only capable of handling a single predictor.
`y`	A vector of values for the response variable.
`y.hat`	A vector of fitted values extracted from a nonparametric smoothing routine.
`side`	Whether a 1-sided or 2-sided tolerance bound is required (determined by `side = 1` or `side = 2`, respectively).
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by the tolerance bound(s).
`method`	The method for determining which indices of the ordered residuals will be used for the tolerance bounds. `"WILKS"`, `"WALD"`, and `"HM"` are each described in `nptol.int`. However, since only one tolerance bound can actually be reported for this procedure, only the first tolerance bound will be returned. Note that this is not an issue when `method = "WILKS"` is used as it only produces one set of tolerance bounds.
`upper`	The upper bound of the data. When `NULL`, then the maximum of `x` is used.
`lower`	The lower bound of the data. When `NULL`, then the minimum of `x` is used.
`new`	When `new = TRUE`, the function shows updated version of outcomes.

Value

`npregtol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by the tolerance bound(s).
`x`	The values of the predictor variable.
`y`	The values of the response variable.
`y.hat`	The predicted value of the response for the fitted nonparametric smoothing routine.

`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side` = 1.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side` = 1.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side` = 2.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side` = 2.

References

Young, D. S. (2013), Regression Tolerance Intervals, *Communications in Statistics - Simulation and Computation*, **42**, 2040–2055.

See Also

`loess`, `nptol.int`, `spline`

Examples

```
## 95%/95% 2-sided nonparametric regression tolerance bounds
## for a sample of size 50.

set.seed(100)
x <- runif(50, 5, 45)
f1 <- function(x, b1, b2) b1 + (0.49 - b1)*exp(-b2*(x - 8)) +
              rnorm(50, sd = 0.01)
y <- f1(x, 0.39, 0.11)
y.hat <- loess(y~x)$fit
out <- npregtol.int(x = x, y = y, y.hat = y.hat, side = 2,
                    alpha = 0.05, P = 0.95, method = "WILKS",
                    new = TRUE)
out

library(plotly)
plotly_regtol(tol.out = out , x = x , y = y)
```

`nptol.int` *Nonparametric Tolerance Intervals*

Description

Provides 1-sided or 2-sided nonparametric (i.e., distribution-free) tolerance intervals for any continuous data set.

Usage

```
nptol.int(x, alpha = 0.05, P = 0.99, side = 1,
          method = c("WILKS", "WALD", "HM", "YM"),
          upper = NULL, lower = NULL)
```

Arguments

`x`	A vector of data which no distributional assumptions are made. The data is only assumed to come from a continuous distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for determining which indices of the ordered observations will be used for the tolerance intervals. `"WILKS"` is the Wilks method, which produces tolerance bounds symmetric about the observed center of the data by using the beta distribution. `"WALD"` is the Wald method, which produces (possibly) multiple tolerance bounds for `side = 2` (each having at least the specified confidence level), but is the same as `method = "WILKS"` for `side = 1`. `"HM"` is the Hahn-Meeker method, which is based on the binomial distribution, but the upper and lower bounds may exceed the minimum and maximum of the sample data. For `side = 2`, this method will yield two intervals if an odd number of observations are to be trimmed from each side. `"YM"` is the Young-Mathew method for performing interpolation or extrapolation based on the order statistics. See below for more information on this method.
`upper`	The upper bound of the data. When `NULL`, then the maximum of `x` is used. If `method = "YM"` and extrapolation is performed, then `upper` will be greater than the maximum.
`lower`	The lower bound of the data. When `NULL`, then the minimum of `x` is used. If `method = "YM"` and extrapolation is performed, then `lower` will be less than the minimum.

Details

For the Young-Mathew method, interpolation or extrapolation is performed. When `side = 1`, two intervals are given: one based on linear interpolation/extrapolation of order statistics (`OS-Based`) and one based on fractional order statistics (`FOS-Based`). When `side = 2`, only an interval based on linear interpolation/extrapolation of order statistics is given.

Value

`nptol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Bury, K. (1999), *Statistical Distributions in Engineering*, Cambridge University Press.

Hahn, G. J. and Meeker, W. Q. (1991), *Statistical Intervals: A Guide for Practitioners*, Wiley-Interscience.

Wald, A. (1943), An Extension of Wilks' Method for Setting Tolerance Limits, *The Annals of Mathematical Statistics*, **14**, 45–55.

Wilks, S. S. (1941), Determination of Sample Sizes for Setting Tolerance Limits, *The Annals of Mathematical Statistics*, **12**, 91–96.

Young, D. S. and Mathew, T. (2014), Improved Nonparametric Tolerance Intervals Based on Interpolated and Extrapolated Order Statistics, *Journal of Nonparametric Statistics*, **26**, 415–432.

See Also

`distfree.est`, `npregtol.int`

Examples

```
## 90%/95% 2-sided nonparametric tolerance intervals for a
## sample of size 200.

set.seed(100)
x <- rlogis(200, 5, 1)
out <- nptol.int(x = x, alpha = 0.10, P = 0.95, side = 1,
                 method = "WILKS", upper = NULL, lower = NULL)
out

plottol(out, x, plot.type = "both", side = "two", x.lab = "X")
```

`paretotol.int` *Pareto (or Power Distribution) Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to either a Pareto distribution or a power distribution (i.e., the inverse Pareto distribution).

Usage

```
paretotol.int(x, alpha = 0.05, P = 0.99, side = 1,
              method = c("GPU", "DUN"), power.dist = FALSE)
```

Arguments

`x`	A vector of data which is distributed according to either a Pareto distribution or a power distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for how the upper tolerance bound is approximated when transforming to utilize the relationship with the 2-parameter exponential distribution. `"GPU"` is the Guenther-Patil-Upppuluri method. `"DUN"` is the Dunsmore method, which was empirically shown to be an improvement for samples greater than or equal to 8. More information on these methods can be found in the "References".
`power.dist`	If `TRUE`, then the data is considered to be from a power distribution, in which case the output gives tolerance intervals for the power distribution. The default is `FALSE`.

Details

Recall that if the random variable X is distributed according to a Pareto distribution, then the random variable $Y = ln(X)$ is distributed according to a 2-parameter exponential distribution. Moreover, if the random variable W is distributed according to a power distribution, then the random variable $X = 1/W$ is distributed according to a Pareto distribution, which in turn means that the random variable $Y = ln(1/W)$ is distributed according to a 2-parameter exponential distribution.

Value

`paretotol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Dunsmore, I. R. (1978), Some Approximations for Tolerance Factors for the Two Parameter Exponential Distribution, *Technometrics*, **20**, 317–318.

Engelhardt, M. and Bain, L. J. (1978), Tolerance Limits and Confidence Limits on Reliability for the Two-Parameter Exponential Distribution, *Technometrics*, **20**, 37–39.

Guenther, W. C., Patil, S. A., and Uppuluri, V. R. R. (1976), One-Sided β-Content Tolerance Factors for the Two Parameter Exponential Distribution, *Technometrics*, **18**, 333–340.

Krishnamoorthy, K., Mathew, T., and Mukherjee, S. (2008), Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability, *Technometrics*, **50**, 69–78.

See Also

TwoParExponential, exp2tol.int

Examples

```
## 95%/99% 2-sided Pareto tolerance intervals
## for a sample of size 500.

set.seed(100)
x <- exp(r2exp(500, rate = 0.15, shift = 2))
out <- paretotol.int(x = x, alpha = 0.05, P = 0.99, side = 2,
                     method = "DUN", power.dist = FALSE)
out

plottol(out, x, plot.type = "both", side = "two",
        x.lab = "Pareto Data")
```

---

plotly_anovatol *Plotting Tolerance Intervals for ANOVA*

Description

Plot tolerance intervals for each factor level in a balanced (or nearly-balanced) ANOVA

Usage

```
plotly_anovatol(tol.out,
                x,
                factors = NULL,
                side = c("two","upper", "lower"),
                range.min = NULL,
                range.max = NULL,
                x.lab = NULL,
                x.lab.size = NULL,
                y.lab = NULL,
                y.lab.size = NULL,
                x.tick.size = NULL,
                y.tick.size = NULL,
                x.col = NULL,
                x.cex = NULL,
                tol.col = NULL,
                tol.lwd = NULL,
                tol.line.type = c("dash","dot","dashdot","solid"),
                tol.lower.col = NULL,
                tol.lower.lwd = NULL,
                tol.lower.line.type = c("dash","dot","dashdot","solid"),
                tol.upper.col = NULL,
                tol.upper.lwd = NULL,
                tol.upper.line.type = c("dash","dot","dashdot","solid"),
                title = NULL,
                title.position.x = NULL,
                title.position.y = NULL,
                title.size = NULL)
```

Arguments

`tol.out`	Output from any ANOVA tolerance interval procedure.
`x`	A data frame consisting of the data fitted in lm.out. Note that data must have one column for each main effect (i.e., factor) that is analyzed in lm.out and that these columns must be of class factor.
`factors`	Specify certain factor(s) to present. The name(s) of the factor(s) needs to be consistent with the name(s) in the original dataset.
`side`	`side = "two"` produces plots for either the two-sided tolerance intervals or both one-sided tolerance intervals. This will be determined by the output in `tol.out`. `side = "upper"` produces plots showing the upper tolerance bounds. `side = "lower"` produces plots showing the lower tolerance bounds. Note that if the output of in `tol.out` shows 2-sided tolerance intervals, `side = "upper"` and `side = "lower"` still shows both upper AND lower tolerance intervals.
`range.min`	Minimum value on the y-axis. If actual lower limit is greater than `range.min`, then the lower limit will be presented.
`range.max`	Maximum value on the y-axis. If actual upper limit is smaller than `range.max`, then the upper limit will be presented.
`x.lab`	Label of the x-axis.
`x.lab.size`	Size of label of the x-axis.
`y.lab`	Label of the y-axis.
`y.lab.size`	Size of label of the y-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.tick.size`	Size of tick marks on the y-axis.
`x.col`	Color of original data points.
`x.cex`	Size of original data points.
`tol.col`	Color of the tolerance intervals when `tol.out` shows 2-sided tolerance intervals.
`tol.lwd`	Width of the tolerance intervals when `tol.out` shows 2-sided tolerance intervals.
`tol.line.type`	Line type of the tolerance intervals when `tol.out` shows 2-sided tolerance intervals.
`tol.lower.col`	Color of the lower tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different colors for upper and lower tolerance intervals.
`tol.lower.lwd`	Width of the lower tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different widths for upper and lower tolerance intervals.

`tol.lower.line.type`	Line type of lower tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different widths for upper and lower tolerance intervals.
`tol.upper.col`	Color of the upper tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different colors for upper and lower tolerance intervals.
`tol.upper.lwd`	Width of the upper tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different widths for upper and lower tolerance intervals.
`tol.upper.line.type`	Line type of upper tolerance interval when `tol.out` shows 1-sided tolerance intervals. When `side = "two"`, users still have options to choose different widths for upper and lower tolerance intervals.
`title`	The main title on top of the plot
`title.position.x`	Horizontal position of the title.
`title.position.y`	Vertical position of the title.
`title.size`	Size of the title.

Value

`plotly_anovatol` returns box plots as well as corresponding tolerance intervals for each main effect of an ANOVA.

References

Howe, W. G. (1969), Two-Sided Tolerance Limits for Normal Populations - Some Improvements, *Journal of the American Statistical Association*, **64**, 610–620.

Weissberg, A. and Beatty, G. (1969), Tables of Tolerance Limit Factors for Normal Distributions, *Technometrics*, **2**, 483–500.

See Also

`anovatol.int`, `plottol`, `K.factor`, `normtol.int`, `lm`, `anova`

Examples

```
## 90%/95% 1-sided tolerance intervals for a 2-way ANOVA
## using the "warpbreaks" data.
attach(warpbreaks)
lm.out <- lm(breaks ~ wool + tension)
out.1 <- anovatol.int(lm.out, data = warpbreaks, alpha = 0.10,
                     P = 0.95, side = 1, method = "HE")
out.1
plotly_anovatol(out.1, x = warpbreaks , factors = 'wool' , x.lab = "Wool" , side="two")
```

```
## 90%/95% 2-sided tolerance intervals for a 2-way ANOVA
## using the "warpbreaks" data.
out.2 <- anovatol.int(lm.out, data = warpbreaks, alpha = 0.10,
                      P = 0.95, side = 2, method = "HE")
out.2
plotly_anovatol(out.2, x = warpbreaks , range.min = 20 , range.max = 60)
```

---

plotly_controltol *Plotting Tolerance Intervals for Control Charts*

Description

Provides interactive control charts for tolerance bounds on continuous data.

Usage

```
plotly_controltol(tol.out ,
                  x ,
                  side = c("two","upper", "lower"),
                  x.lab = NULL,
                  x.lab.size = NULL,
                  y.lab = NULL,
                  y.lab.size = NULL,
                  x.tick.size = NULL,
                  y.tick.size = NULL,
                  x.col = NULL,
                  x.cex = NULL,
                  fit.col = NULL,
                  fit.lwd = NULL,
                  fit.line.type = c("solid","dash","dot","dashdot"),
                  tol.col = NULL,
                  tol.lwd = NULL,
                  tol.line.type = c("dash","dot","dashdot","solid"),
                  title.position.x = NULL,
                  title.position.y = NULL,
                  title.size = NULL,
                  title = NULL)
```

Arguments

`tol.out`	Output from any continuous tolerance interval procedure.
`x`	Data from a continuous distribution.
`side`	`side = "two"` produces plots for either the two-sided tolerance intervals or both one-sided tolerance intervals. This will be determined by the output in `tol.out`. `side = "upper"` produces plots showing the upper tolerance bounds. `side = "lower"` produces plots showing the lower tolerance bounds. Note that if the output of in `tol.out` shows 2-sided tolerance intervals, `side = "upper"` and `side = "lower"` still shows both upper AND lower tolerance intervals.

`x.lab`	Label of the x-axis.
`x.lab.size`	Size of label of the x-axis.
`y.lab`	Label of the y-axis.
`y.lab.size`	Size of label of the y-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.tick.size`	Size of tick marks on the y-axis.
`x.col`	Color of original data points.
`x.cex`	Size of original data points.
`fit.col`	Color of fitted line.
`fit.lwd`	Width of fitted line.
`fit.line.type`	Type of the fitted line.
`tol.col`	Color of the tolerance intervals when `tol.out` shows 2-sided tolerance intervals.
`tol.lwd`	Width of the tolerance intervals when `tol.out` shows 2-sided tolerance intervals.
`tol.line.type`	Line type of tolerance intervals.
`title`	The main title on top of the plot.
`title.size`	Size of the title.
`title.position.x`	Horizontal position of the title.
`title.position.y`	Vertical position of the title.

Value

`plotly_controltol` can return boxplots as well as corresponding tolerance intervals for any continuous data.

References

Montgomery, D. C. (2005), *Introduction to Statistical Quality Control*, Fifth Edition, John Wiley & Sons, Inc.

See Also

`plottol`

Examples

```
## 95%/85% 2-sided Bayesian normal tolerance limits for
## a sample of size 100.
set.seed(100)
x <- rnorm(100)
out <- bayesnormtol.int(x = x, alpha = 0.05, P = 0.85,
                        side = 2, method = "EXACT",
                        hyper.par = list(mu.0 = 0,
                                            sig2.0 = 1, n.0 = 10, m.0 = 10))
out
plotly_controltol(out, x, x.lab = "Normal Data")
```

`plotly_histtol` *Plotting Histograms and Corresponding Tolerance Intervals for Continuous Data*

Description

Provides interactive tolerance intervals for continous data based on its histogram.

Usage

```
plotly_histtol(tol.out,
               x,
               side = c("two","upper", "lower"),
               x.lab = NULL,
               x.lab.size = NULL,
               x.tick.size = NULL,
               y.lab.size = NULL,
               y.tick.size = NULL,
               title = NULL,
               title.size = NULL,
               title.position.x = NULL,
               title.position.y = NULL,
               bin.col = NULL,
               tol.col = NULL,
               tol.lwd = NULL,
               tol.line.type = c("dash","dot","dashdot","solid"))
```

Arguments

`tol.out`	Output from any continuous tolerance interval procedure.
`x`	Data from a continuous distribution.
`side`	`side = "two"` produces plots for either the two-sided tolerance intervals or both one-sided tolerance intervals. This will be determined by the output in `tol.out`. `side = "upper"` produces plots showing the upper tolerance bounds. `side = "lower"` produces plots showing the lower tolerance bounds. Note that if the output of in `tol.out` shows 2-sided tolerance intervals, `side = "upper"` and `side = "lower"` still shows both upper AND lower tolerance intervals.
`x.lab`	Label of the x-axis.
`x.lab.size`	Size of label of the x-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.lab.size`	Size of label of the y-axis.

`y.tick.size`	Size of tick marks on the y-axis.
`title`	The main title on top of the histogram.
`title.size`	Size of the title.
`title.position.x`	Horizontal position of the title.
`title.position.y`	Vertical position of the title.
`bin.col`	Color of the bins.
`tol.col`	Color of the tolerance interval(s).
`tol.lwd`	Width of the tolerance interval(s).
`tol.line.type`	Line type of the tolerance interval(s).

Value

`plotly_histtol` can return histograms as well as corresponding tolerance intervals for any continuous data.

References

Montgomery, D. C. (2005), *Introduction to Statistical Quality Control*, Fifth Edition, John Wiley & Sons, Inc.

See Also

`plottol`

Examples

```
## 90%/90% 1-sided Weibull tolerance intervals for a sample
## of size 150.
set.seed(100)
x <- rweibull(150, 3, 75)
out <- exttol.int(x = x, alpha = 0.15, P = 0.90, dist = "Weibull" , side = 1)
out
plotly_histtol(out, x, side = "lower", x.lab = "Weibull Data" , tol.lwd = 3)
```

`plotly_multitol` *Plotting Tolerance Region for Multivariate Distributions*

Description

Provides interactive tolerance region on multivariate continuous data.

Usage

```
plotly_multitol(tol.out,
               x,
               x.lab = NULL,
               x.lab.size = NULL,
               y.lab = NULL,
               y.lab.size = NULL,
               z.lab = NULL,
               z.lab.size = NULL,
               x.tick.size = NULL,
               y.tick.size = NULL,
               z.tick.size = NULL,
               x.col = NULL,
               x.cex = NULL,
               tol.col = NULL,
               tol.lwd = NULL,
               tol.line.type = c("dash","dot","dashdot","solid"),
               title = NULL,
               title.position.x = NULL,
               title.position.y = NULL,
               title.size = NULL)
```

Arguments

`tol.out`	Output from `mvtol.region` for multivariate data.
`x`	Multivariate data from continuous distributions.
`x.lab`	Label of the x-axis.
`x.lab.size`	Size of label of the x-axis.
`y.lab`	Label of the y-axis.
`y.lab.size`	Size of label of the y-axis.
`z.lab`	Label of the z-axis.
`z.lab.size`	Size of label of the z-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.tick.size`	Size of tick marks on the y-axis.
`z.tick.size`	Size of tick marks on the z-axis.
`x.col`	Color of original data points.
`x.cex`	Size of original data points.
`tol.col`	Color of the tolerance region.
`tol.lwd`	Width of boundary of the tolerance region when data is bivariate.
`tol.line.type`	Line type of the tolerance region for bivariate data.
`title`	The main title on top of the plot.
`title.size`	Size of the title.

`title.position.x`
: Horizontal position of the title.

`title.position.y`
: Vertical position of the title.

Value

`plotly_multitol` returns tolerance regions for both bivariate and trivariate continuous data.

References

Krishnamoorthy, K. and Mathew, T. (1999), Comparison of Approximation Methods for Computing Tolerance Factors for a Multivariate Normal Population, *Technometrics*, **41**, 234–249.

Krishnamoorthy, K. and Mondal, S. (2006), Improved Tolerance Factors for Multivariate Normal Distributions, *Communications in Statistics - Simulation and Computation*, **35**, 461–478.

See Also

`plottol`, `mvtol.region`

Examples

```
## 90%/90% bivariate normal tolerance region.
set.seed(100)
x1 <- rnorm(100, 0, 0.2)
x2 <- rnorm(100, 0, 0.5)
x <- cbind(x1, x2)
out1 <- mvtol.region(x = x, alpha = 0.10, P = 0.90, B = 1000,
                     method = "KM")
out1
plotly_multitol(out1, x , x.lab = "X1" , y.lab = "X2")

## 90%/90% trivariate normal tolerance region.
set.seed(100)
x1 <- rnorm(100, 0, 0.2)
x2 <- rnorm(100, 0, 0.5)
x3 <- rnorm(100, 5, 1)
x <- cbind(x1, x2, x3)
mvtol.region(x = x, alpha = c(0.10, 0.05, 0.01),
             P = c(0.90, 0.95, 0.99), B = 1000, method = "KM")
out2 <- mvtol.region(x = x, alpha = 0.10, P = 0.90, B = 1000,
                     method = "KM")
out2
plotly_multitol(out2, x , x.lab = "X1" , y.lab = "X2" , z.lab = "X3",
                title.position.x = 0.57)
```

`plotly_normOC` *Operating Characteristic (OC) Curves for K-Factors for Tolerance Intervals Based on Normality (a `plotly` version of `norm.OC`)*

Description

`plotly_normOC` is an updated function rooted in `norm.OC`.

Usage

```
plotly_normOC(k = NULL,
              alpha = NULL,
              P = NULL,
              n,
              side = 1,
              method = c("HE", "HE2", "WBE", "ELL", "KM", "EXACT", "OCT"),
              m = 50,
              range.min = NULL,
              range.max = NULL,
              x.lab.size = NULL,
              y.lab.size = NULL,
              x.tick.size = NULL,
              y.tick.size = NULL,
              title = NULL,
              title.size = NULL,
              title.position.x = NULL,
              title.position.y = NULL,
              legend.size = NULL,
              x.cex = NULL,
              line.width = NULL,
              line.type = c("solid","dash","dot","dashdot"))
```

Arguments

`k`	If wanting OC curves where the confidence level or content level is on the y-axis, then a single positive value of `k` must be specified. This would be the target k-factor for the desired tolerance interval. If `k = NULL`, then OC curves will be constructed where the k-factor value is found for given levels of `alpha`, `P`, and `n`.
`alpha`	The set of levels chosen such that `1-alpha` are confidence levels. If wanting OC curves where the content level is being calculated, then each curve will correspond to a level in the set of `alpha`. If a set of `P` values is specified, then OC curves will be constructed where the k-factor is found and each curve will correspond to each combination of `alpha` and `P`. If `alpha = NULL`, then OC curves will be constructed to find the confidence level for given levels of `k`, `P`, and `n`.

`P`	The set of content levels to be considered. If wanting OC curves where the confidence level is being calculated, then each curve will correspond to a level in the set of `P`. If a set of `alpha` values is specified, then OC curves will be constructed where the k-factor is found and each curve will correspond to each combination of `alpha` and `P`. If `P = NULL`, then OC curves will be constructed to find the content level for given levels of `k`, `alpha`, and `n`.
`n`	A sequence of sample sizes to consider. This must be a vector of at least length 2 since all OC curves are constructed as functions of `n`.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the k-factors. The k-factor for the 1-sided tolerance intervals is performed exactly and thus is the same for the chosen method. `"HE"` is the Howe method and is often viewed as being extremely accurate, even for small sample sizes. `"HE2"` is a second method due to Howe, which performs similarly to the Weissberg-Beatty method, but is computationally simpler. `"WBE"` is the Weissberg-Beatty method (also called the Wald-Wolfowitz method), which performs similarly to the first Howe method for larger sample sizes. `"ELL"` is the Ellison correction to the Weissberg-Beatty method when `f` is appreciably larger than `n^2`. A warning message is displayed if `f` is not larger than `n^2`. `"KM"` is the Krishnamoorthy-Mathew approximation to the exact solution, which works well for larger sample sizes. `"EXACT"` computes the k-factor exactly by finding the integral solution to the problem via the `integrate` function. Note the computation time of this method is largely determined by `m`. `"OCT"` is the Owen approach to compute the k-factor when controlling the tails so that there is not more than (1-P)/2 of the data in each tail of the distribution.
`m`	The maximum number of subintervals to be used in the `integrate` function, which is used for the underlying exact method for calculating the normal tolerance intervals.
`range.min`	The minimum value of the y-axis.
`range.max`	The maximum value of the y-axis.
`x.lab.size`	Size of label of the x-axis.
`y.lab.size`	Size of label of the y-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.tick.size`	Sze of tick marks on the y-axis.
`title`	The main title on top of the plot.
`title.size`	Size of the title.
`title.position.x`	Horizontal position of the title.
`title.position.y`	Vertical position of the title.
`legend.size`	Size of the legend.
`x.cex`	Size of data points.
`line.width`	Width of lines connecting data points.
`line.type`	The type of lines connection data points.

Value

norm.OC returns a figure with the OC curves constructed using the specifications in the arguments.

References

Young, D. S. (2016), Normal Tolerance Interval Procedures in the tolerance Package, *The R Journal*, **8**, 200–212.

See Also

K.factor, normtol.int, norm.OC

Examples

```
## The three types of OC-curves that can be constructed
## with the ggnorm.OC function.

plotly_normOC(k = 4, alpha = NULL, P = c(0.90, 0.95, 0.99),
          n = 10:20, side = 1)

plotly_normOC(k = 4, alpha = c(0.01, 0.05, 0.10), P = NULL,
          n = 10:20, side = 1)

plotly_normOC(k = NULL, P = c(0.90, 0.95, 0.99),
          alpha=c(0.01,0.05,0.10), n = 10:20, side = 1)
```

---

plotly_npmvtol *plotting Nonparametric Multivaraite Hyperrectangular Tolerance Region*

Description

plotly_npmvtol is plotting function for nonparametric multivaraite hyperrectangular tolerance region. The function takes the outcome of npmvtol.region as an input and provides visualzation for hypperrectangular tolerance regions between two variables.

Usage

```
plotly_npmvtol(tol.out,
                x,
                var.names = NULL,
                title = NULL,
                x.col = "#4298B5",
                x.cex = 6,
                x.shape = "dot",
                outlier.col = "#A6192E",
                outlier.cex = 8,
                outlier.shape = "triangle-up",
                tol.col = "#D1DDE6",
                tol.opacity = 0.4,
                x.lab.size = 12,
                x.tick.size = 12,
                y.lab.size = 12,
                y.tick.size = 12,
                title.position.x = 0.5,
                title.position.y = 0.98,
                title.size = 12,
                show.bound = TRUE,
                bound.type = c("dash", "dot", "solid", "longdash",
                               "dashdot", "longdashdot"),
                bound.col = "#000000",
                bound.lwd = 1
                )
```

Arguments

`tol.out`	Output from `npmvtol.region` for multivariate data.
`x`	Data frame for different variables. Columns of `x` represent for different variables.
`var.names`	Labels of variable names. The dimension of `var.names` needs to be consistent with column dimension of `x`.
`title`	The main title on top of the plot. The length of `title` can be either 1 or multiple. If only 1 title is specified, all plots share the same title. If multiple titles are specified, number of titles needs to be consistent with the number of combinations of variables. For example, if the data has 4 variables, either 1 or 6 (choose 2 out of 4) titles need to be specified.
`x.col`	Color of original data points, excluding outliers.
`x.cex`	Size of original data points, excluding outliers.
`x.shape`	Shape of original data points, excluding outliers.
`outlier.col`	Color of outliers.
`outlier.cex`	Size of outliers.
`outlier.shape`	Shape of outliers.
`tol.col`	Color of tolerance region.
`tol.opacity`	Opacity of tolerance region.
`x.lab.size`	Size of label of the x-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.lab.size`	Size of label of the y-axis.
`y.tick.size`	Size of tick marks on the y-axis.
`title.position.x`	Horizontal position of the title.

`title.position.y`	Vertical position of the title.
`title.size`	Size of the title.
`show.bound`	Logical indicating to show rectanglular boundaries. Default is `TRUE`.
`bound.type`	Line type of the rectangle boundaries.
`bound.col`	Color of the rectangle boundaries.
`bound.lwd`	Width of the rectangle boundaries.

Value

`plotly_npmvtol` returns figures of hypperectangular tolerance regions between two random variable generated by `npmvtol.region`.

References

Young, D. S., & Mathew, T. (2020), Nonparametric Hyperrectangular Tolerance and Prediction Regions for Setting Multivariate Reference Regions in Laboratory Medicine. *Statistical Methods in Medical Research*, **29**, 3569–3585.

See Also

`npmvtol.region`

Examples

```
library(plotly)

mdepth <- function(pts, x){
  mahalanobis(pts, center = rep(0, 3),
              cov = diag(1, 3))
}

set.seed(100)
x <- cbind(X1=rnorm(300), X2=rnorm(300), X3=rnorm(300))
out <-npmvtol.region(x = x, alpha = 0.10, P = 0.90, depth.fn = mdepth,
                     type = "semispace", semi.order = list(lower = 2,
                                                            center = 3, upper = 1))

gg.out <- plotly_npmvtol(tol.out = out , x = x)
```

plotly_regtol *Plotting Tolerance Intervals for Regressions*

Description

Provides interactive tolerance intervals for regression data. More specifically, `plotly_regtol` presents tolerance bounds for linear regression, nonlinear regression, and nonparametric regression models. In addtion, this updated function is capable of showing tolerance plane for trivariate regression models.

Usage

```
plotly_regtol(tol.out,
              x,
              new.x = NULL,
              y,
              side = c("two","upper", "lower"),
              rect = FALSE,
              smooth = 4,
              x.lab = NULL,
              x.lab.size = NULL,
              y.lab = NULL,
              y.lab.size = NULL,
              z.lab = NULL,
              z.lab.size = NULL,
              x.tick.size = NULL,
              y.tick.size = NULL,
              z.tick.size = NULL,
              x.col = NULL,
              x.cex = NULL,
              fit.col = NULL,
              fit.lwd = NULL,
              fit.line.type = c("dash","dot","dashdot","solid"),
              fit.opacity = NULL,
              tol.col = NULL,
              tol.lwd = NULL,
              tol.line.type = c("dash","dot","dashdot","solid"),
              tol.opacity = NULL,
              title.position.x = NULL,
              title.position.y = NULL,
              title = NULL,
              title.size = NULL)
```

Arguments

`tol.out`	Output from `regtol.int`, `nlregtol.int`, `npregtol.int` or `mvregtol.region` for regressional data.

`x`	Data frame for explanatory variables. If there are more than one explanatory variables, columns of `x` represents regressors.
`new.x`	An optional data frame in which to look for variables with which to predict. `new.x` can be used to plot linear regression, nonlinear regression, and multivariate linear regression. `new.x` has to be a subset of new data in the original output.
`y`	Data frame for response variable. `y` is in the formate of a vector.
`side`	`side = "two"` produces plots for either the two-sided tolerance intervals or both one-sided tolerance intervals. This will be determined by the output in `tol.out`. `side = "upper"` produces plots showing the upper tolerance bounds. `side = "lower"` produces plots showing the lower tolerance bounds. Note that if the output of in `tol.out` shows 2-sided tolerance intervals, `side = "upper"` and `side = "lower"` still shows both upper AND lower tolerance intervals.
`rect`	This argument is used for plotting tolerance plane(s) of multivariate regression region. When `rect=TRUE` the x1-x2 plane is a rectangle.
`smooth`	The smooth parameter for the x1-x2 plane when `rect=TRUE`.
`x.lab`	Label of the x-axis.
`x.lab.size`	Size of label of the x-axis.
`y.lab`	Label of the y-axis.
`y.lab.size`	Size of label of the y-axis.
`z.lab`	Label of the z-axis.
`z.lab.size`	Size of label of the z-axis.
`x.tick.size`	Size of tick marks on the x-axis.
`y.tick.size`	Size of tick marks on the y-axis.
`z.tick.size`	Size of tick marks on the z-axis.
`x.col`	Color of original data points.
`x.cex`	Size of original data points.
`fit.col`	Color of fitted line or fitted plane.
`fit.lwd`	Width of fitted line or fitted plane.
`fit.line.type`	Type of fitted line or fitted plane.
`fit.opacity`	Opacity of fitted line or fitted plane.
`tol.col`	Color of tolerance intervals or tolerance plane.
`tol.lwd`	Width of tolerance intervals.
`tol.line.type`	Line type of tolerance intervals
`tol.opacity`	Opacity of tolerance region.
`title.position.x`	Horizontal position of the title.
`title.position.y`	Vertical position of the title.
`title`	The main title on top of the plot.
`title.size`	Size of the title.

Value

`plotly_regtol` returns tolerance intervals for linear regression, nonlinear regression, nonparametric regression, as well as tolerance planes for multivariate (multiple) linear regression models.

References

Montgomery, D. C. (2005), Introduction to Statistical Quality Control, Fifth Edition, *John Wiley & Sons, Inc*.

See Also

`plottol`, `regtol.int`, `regtol.int`, `nlregtol.int`, `npregtol.int`, `npregtol.int`,`mvregtol.region`

Examples

```
## 95%/95% 1-sided linear regression tolerance bounds
## for a sample of size 100.

library(plotly)

set.seed(100)
x <- runif(100, 0, 10)
y <- 20 + 5*x + rnorm(100, 0, 3)
out1 <- regtol.int(reg = lm(y ~ x), new.x = c(3,6,20), new=TRUE ,
                    side = 1, alpha = 0.05, P = 0.95)
plotly_regtol(tol.out = out1 , x=x , y=y , new.x = c(6,20), side = "two" ,
                   fit.line.type = "dash" , tol.line.type = "solid")
########################
set.seed(100)
x1 <- runif(100, 0, 10)
x2 <- rpois(100 , 5)
y <- 20 + 5*x1 + 3*x2 + rnorm(100, 0, 3)
x1.new <- runif(10 , 0 , 10)
x2.new <- rpois(10 , 5)
out2 <- regtol.int(reg = lm(y ~ x1 + x2), new.x = cbind(x1.new , x2.new), new=TRUE,
                    side = 1, alpha = 0.05, P = 0.95)
plotly_regtol(tol.out = out2 , y=y , x=cbind(x1,x2) , new.x = cbind(x1.new , x2.new) ,
                   rect = TRUE , side = "two")
###########################
## 95%/95% 2-sided nonlinear regression tolerance bounds
## for a sample of size 50.
set.seed(100)
x <- runif(50, 5, 45)
f1 <- function(x, b1, b2) b1 + (0.49 - b1)*exp(-b2*(x - 8)) +
  rnorm(50, sd = 0.01)
y <- f1(x, 0.39, 0.11)
formula <- as.formula(y ~ b1 + (0.49 - b1)*exp(-b2*(x - 8)))
out1 <- nlregtol.int(formula = formula,
                      xy.data = data.frame(cbind(y, x)),
                      x.new=c(10,20,50), side = 2,
                      alpha = 0.05, P = 0.95 , new = TRUE)
plotly_regtol(tol.out = out1 , x=x , y=y , new.x = c(20,50) , side = "two",
                   fit.line.type = "dot")
###############

## 95%/95% 1-sided nonparametric regression tolerance bounds
## for a sample of size 50.
set.seed(100)
x <- runif(50, 5, 45)
f1 <- function(x, b1, b2) b1 + (0.49 - b1)*exp(-b2*(x - 8)) + rnorm(50, sd = 0.01)
y <- f1(x, 0.39, 0.11)
y.hat <- loess(y~x)$fit
out1 <- npregtol.int(x = x, y = y, y.hat = y.hat, side = 1,
                      alpha = 0.05, P = 0.95, method = "WILKS" , new = TRUE)
plotly_regtol(tol.out = out1 , x=x , y=y , side = "two" , fit.line.type = "dash")
############
set.seed(100)
x1 <- runif(50, 5, 45)
x2 <- rnorm(50 , 0 , 1)
f1 <- function(x1 , x2 , b1, b2) {b1 + (0.49 - b1)*exp(-b2*(x1 + x2 - 8)) + rnorm(50, sd = 0.01)}
y <- f1(x1 , x2 , 0.39, 0.11)
y.hat <- loess(y~ x1 + x2)$fit
out2 <- npregtol.int(x = cbind(x1 , x2), y = y, y.hat = y.hat, side = 1,
                      alpha = 0.05, P = 0.95, method = "WILKS" , new = TRUE)
plotly_regtol(tol.out = out2 , y=y , x=cbind(x1,x2) ,
                   rect = TRUE , smooth = 100 ,  side = "two")
```

plottol — *Plotting Capabilities for Tolerance Intervals*

Description

Provides control charts and/or histograms for tolerance bounds on continuous data as well as tolerance ellipses for data distributed according to bivariate and trivariate normal distributions. Scatterplots with regression tolerance bounds and interval plots for ANOVA tolerance intervals may also be produced.

Usage

```
plottol(tol.out, x, y = NULL, y.hat = NULL,
         side = c("two", "upper", "lower"),
         plot.type = c("control", "hist", "both"),
         x.lab = NULL, y.lab = NULL, z.lab = NULL, ...)
```

Arguments

`tol.out`	Output from any continuous (including ANOVA) tolerance interval procedure or from a regression tolerance bound procedure.
`x`	Either data from a continuous distribution or the predictors for a regression model. If this is a design matrix for a linear regression model, then it must be in matrix form AND include a column of 1's if there is to be an intercept. Note

	that multiple predictors are only allowed if considering polynomial regression. If the output for `tol.out` concerns ANOVA tolerance intervals, then `x` must be a data frame.
`y`	The response vector for a regression setting. Leave as `NULL` if not doing regression tolerance bounds.
`y.hat`	The fitted values from a nonparametric smoothing routine if plotting nonparametric regression tolerance bounds. Otherwise, leave as `NULL`.
`side`	`side = "two"` produces plots for either the two-sided tolerance intervals or both one-sided tolerance intervals. This will be determined by the output in `tol.out`. `side = "upper"` produces plots showing the upper tolerance bounds. `side = "lower"` produces plots showing the lower tolerance bounds.
`plot.type`	`plot.type = "control"` produces a control chart of the data along with the tolerance bounds specified by `side`. `plot.type = "hist"` produces a histogram of the data along with the tolerance bounds specified by `side`. `plot.type = "both"` produces both the control chart and histogram. This argument is ignored when plotting regression data.
`x.lab`	Specify the label for the x-axis.
`y.lab`	Specify the label for the y-axis.
`z.lab`	Specify the label for the z-axis.
`...`	Additional arguments passed to the plotting function used for the control charts or regression scatterplots.

Value

`plottol` can return a control chart, histogram, or both for continuous data along with the calculated tolerance intervals. For regression data, `plottol` returns a scatterplot along with the regression tolerance bounds. For ANOVA output, `plottol` returns an interval plot for each factor.

References

Montgomery, D. C. (2005), *Introduction to Statistical Quality Control*, Fifth Edition, John Wiley & Sons, Inc.

Examples

```
## 90%/90% 1-sided Weibull tolerance intervals for a sample
## of size 150.

set.seed(100)
x <- rweibull(150, 3, 75)
out <- exttol.int(x = x, alpha = 0.15, P = 0.90,
                  dist = "Weibull")
out

plottol(out, x, plot.type = "both", side = "lower",
        x.lab = "Weibull Data")
```

```
## 90%/90% trivariate normal tolerance region.

set.seed(100)
x1 <- rnorm(100, 0, 0.2)
x2 <- rnorm(100, 0, 0.5)
x3 <- rnorm(100, 5, 1)
x <- cbind(x1, x2, x3)
mvtol.region(x = x, alpha = c(0.10, 0.05, 0.01),
             P = c(0.90, 0.95, 0.99), B = 1000)

out2 <- mvtol.region(x = x, alpha = 0.10, P = 0.90, B = 1000)
out2
plottol(out2, x)

## 95%/95% 2-sided linear regression tolerance bounds
## for a sample of size 100.

set.seed(100)
x <- runif(100, 0, 10)
y <- 20 + 5*x + rnorm(100, 0, 3)
out3 <- regtol.int(reg = lm(y ~ x), new.x = data.frame(x = c(3, 6, 9)),
                   side = 2, alpha = 0.05, P = 0.95)
plottol(out3, x = cbind(1, x), y = y, side = "two", x.lab = "X",
        y.lab = "Y")
```

poislind.ll *Maximum Likelihood Estimation for the Discrete Poisson-Lindley Distribution*

Description

Performs maximum likelihood estimation for the parameter of the Poisson-Lindley distribution.

Usage

```
poislind.ll(x, theta = NULL, ...)
```

Arguments

`x`	A vector of raw data which is distributed according to a Poisson-Lindley distribution.
`theta`	Optional starting value for the parameter. If `NULL`, then the method of moments estimator is used.
`...`	Additional arguments passed to the `mle` function.

Details

The discrete Poisson-Lindley distribution is a compound distribution that, potentially, provides a better fit for count data relative to the traditional Poisson and negative binomial distributions.

Value

See the help file for `mle` to see how the output is structured.

References

Ghitany, M. E. and Al-Mutairi, D. K. (2009), Estimation Methods for the Discrete Poisson-Lindley Distribution, *Journal of Statistical Computation and Simulation*, **79**, 1–9.

Sankaran, M. (1970), The Discrete Poisson-Lindley Distribution, *Biometrics*, **26**, 145–149.

See Also

`mle`, `PoissonLindley`

Examples

```
## Maximum likelihood estimation for randomly generated data
## from the Poisson-Lindley distribution.

set.seed(100)

pl.data <- rpoislind(n = 500, theta = 0.5)
out.pl <- poislind.ll(pl.data)
stats4::coef(out.pl)
stats4::vcov(out.pl)
```

`poislindtol.int` *Poisson-Lindley Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to the Poisson-Lindley distribution.

Usage

```
poislindtol.int(x, m = NULL, alpha = 0.05, P = 0.99, side = 1,
                ...)
```

Arguments

`x`	A vector of raw data which is distributed according to a Poisson-Lindley distribution.
`m`	The number of observations in a future sample for which the tolerance limits will be calculated. By default, `m = NULL` and, thus, `m` will be set equal to the original sample size.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.

`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`...`	Additional arguments passed to the `poislind.ll` function, which is used for maximum likelihood estimation.

Details

The discrete Poisson-Lindley distribution is a compound distribution that, potentially, provides a better fit for count data relative to the traditional Poisson and negative binomial distributions. Poisson-Lindley distributions are heavily right-skewed distributions. For most practical applications, one will typically be interested in 1-sided upper bounds.

Value

`poislindtol.int` returns a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`theta`	MLE for the shape parameter `theta`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Naghizadeh Qomi, M., Kiapour, A., and Young, D. S. (2015), Approximate Tolerance Intervals for the Discrete Poisson-Lindley Distribution, *Journal of Statistical Computation and Simulation*, **86**, 841–854.

See Also

`PoissonLindley`, `poislind.ll`

Examples

```
## 90%/90% 1-sided tolerance intervals for data assuming
## the Poisson-Lindley distribution.

x <- c(rep(0, 447), rep(1, 132), rep(2, 42), rep(3, 21),
       rep(4, 3), rep(5, 2))
out <- poislindtol.int(x, alpha = 0.10, P = 0.90, side = 1)
out
```

`PoissonLindley` *Discrete Poisson-Lindley Distribution*

Description

Density (mass), distribution function, quantile function, and random generation for the Poisson-Lindley distribution.

Usage

```
dpoislind(x, theta, log = FALSE)
ppoislind(q, theta, lower.tail = TRUE, log.p = FALSE)
qpoislind(p, theta, lower.tail = TRUE, log.p = FALSE)
rpoislind(n, theta)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	The number of observations. If `length>1`, then the length is taken to be the number required.
`theta`	The shape parameter, which must be greater than 0.
`log, log.p`	Logical vectors. If `TRUE`, then the probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.

Details

The Poisson-Lindley distribution has mass

$$p(x) = \frac{\theta^2(x + \theta + 2)}{(\theta + 1)^{x+3}},$$

where $x = 0, 1, \ldots$ and $\theta > 0$ is the shape parameter.

Value

`dpoislind` gives the density (mass), `ppoislind` gives the distribution function, `qpoislind` gives the quantile function, and `rpoislind` generates random deviates for the specified distribution.

References

Ghitany, M. E. and Al-Mutairi, D. K. (2009), Estimation Methods for the Discrete Poisson-Lindley Distribution, *Journal of Statistical Computation and Simulation*, **79**, 1–9.

Sankaran, M. (1970), The Discrete Poisson-Lindley Distribution, *Biometrics*, **26**, 145–149.

See Also

runif and .Random.seed about random number generation.

Examples

```
## Randomly generated data from the Poisson-Lindley
## distribution.

set.seed(100)
x <- rpoislind(n = 150, theta = 0.5)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- dpoislind(x = x.1, theta = 0.5)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, ppoislind(q = x.1, theta = 0.5), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")

qpoislind(p = 0.20, theta = 0.5, lower.tail = FALSE)
qpoislind(p = 0.80, theta = 0.5)
```

---

poistol.int *Poisson Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for Poisson random variables. From a statistical quality control perspective, these limits bound the number of occurrences (which follow a Poisson distribution) in a specified future time period.

Usage

```
poistol.int(x, n, m = NULL, alpha = 0.05, P = 0.99, side = 1,
            method = c("TAB", "LS", "SC", "CC", "VS", "RVS",
            "FT", "CSC"))
```

Arguments

x	The number of occurrences of the event in time period n. Can be a vector of length n, in which case the sum of x is used.
n	The time period of the original measurements.
m	The specified future length of time. If m = NULL, then the tolerance limits will be constructed assuming n for the future length of time.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of occurrences in future time lengths of size m to be covered by this tolerance interval.

`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`method`	The method for calculating the lower and upper confidence bounds, which are used in the calculation of the tolerance bounds. The default method is `"TAB"`, which is the tabular method and is usually preferred for a smaller number of occurrences. `"LS"` gives the large-sample (Wald) method, which is usually preferred when the number of occurrences is `x>20`. `"SC"` gives the score method, which again is usually used when the number of occurrences is relatively large. `"CC"` gives a continuity-corrected version of the large-sample method. `"VS"` gives a variance-stabilized version of the large-sample method. `"RVS"` is a recentered version of the variance-stabilization method. `"FT"` is the Freeman-Tukey method. `"CSC"` is the continuity-corrected version of the score method. More information on these methods can be found in the "References".

Value

`poistol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of occurrences in future time periods of length `m`.
`lambda.hat`	The mean occurrence rate per unit time, calculated by `x/n`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Barker, L. (2002), A Comparison of Nine Confidence Intervals for a Poisson Parameter When the Expected Number of Events Is ≤ 5, *The American Statistician*, **56**, 85–89.

Freeman, M. F. and Tukey, J. W. (1950), Transformations Related to the Angular and the Square Root, *Annals of Mathematical Statistics*, **21**, 607–611.

Hahn, G. J. and Chandra, R. (1981), Tolerance Intervals for Poisson and Binomial Variables, *Journal of Quality Technology*, **13**, 100–110.

See Also

`Poisson`, `umatol.int`

Examples

```
## 95%/90% 1-sided Poisson tolerance limits for future
## occurrences in a period of length 3.  All seven methods
## are presented for comparison.

poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "TAB")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "LS")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "SC")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "CC")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "VS")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "RVS")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "FT")
poistol.int(x = 45, n = 9, m = 3, alpha = 0.05, P = 0.90,
            side = 1, method = "CSC")

## 95%/90% 2-sided Poisson tolerance intervals for future
## occurrences in a period of length 15.  All seven methods
## are presented for comparison.

poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "TAB")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "LS")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "SC")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "CC")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "VS")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "RVS")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "FT")
poistol.int(x = 45, n = 9, m = 15, alpha = 0.05, P = 0.90,
            side = 2, method = "CSC")
```

`regtol.int` *(Multiple) Linear Regression Tolerance Bounds*

Description

Provides 1-sided or 2-sided (multiple) linear regression tolerance bounds. It is also possible to fit a regression through the origin model.

Usage

```
regtol.int(reg, new.x = NULL, side = 1, alpha = 0.05, P = 0.99, new = FALSE)
```

Arguments

reg	An object of class lm (i.e., the results from a linear regression routine).
new.x	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
side	Whether a 1-sided or 2-sided tolerance bound is required (determined by side = 1 or side = 2, respectively).
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of the population to be covered by the tolerance bound(s).
new	When new = TRUE, the function shows updated version of outcomes.

Value

regtol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of the population covered by the tolerance bound(s).
y	The value of the response given on the left-hand side of the model in reg.
y.hat	The predicted value of the response for the fitted linear regression model. This data frame is sorted by this value.
1-sided.lower	The 1-sided lower tolerance bound. This is given only if side = 1.
1-sided.upper	The 1-sided upper tolerance bound. This is given only if side = 1.
2-sided.lower	The 2-sided lower tolerance bound. This is given only if side = 2.
2-sided.upper	The 2-sided upper tolerance bound. This is given only if side = 2.

References

Wallis, W. A. (1951), Tolerance Intervals for Linear Regression, in *Second Berkeley Symposium on Mathematical Statistics and Probability*, ed. J. Neyman, Berkeley: University of CA Press, 43–51.

Young, D. S. (2013), Regression Tolerance Intervals, *Communications in Statistics - Simulation and Computation*, **42**, 2040–2055.

See Also

lm

Examples

```
## 95%/95% 2-sided linear regression tolerance bounds
## for a sample of size 100.

set.seed(100)
x <- runif(100, 0, 10)
y <- 20 + 5*x + rnorm(100, 0, 3)
out <- regtol.int(reg = lm(y ~ x), new.x = data.frame(x = c(3, 6, 9)),
                  side = 2, alpha = 0.05, P = 0.95)
```

```
out

plottol(out, x = cbind(1, x), y = y, side = "two", x.lab = "X",
        y.lab = "Y")
```

`semiconttol.int` *Generalized Intervals for Semicontinuous Data*

Description

Provides confidence intervals, one-sided prediction limits, and one-sided tolerance limits for semi-continuous data — either zero-inflated gamma (ZIG) or zero-inflated lognormal (ZILN) distribution — using a generalized fiducial framework.

Usage

```
semiconttol.int(x, alpha = 0.05, P = 0.99, N = 1000)
```

Arguments

`x`	A vector of semicontinuous data.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`N`	The number of fiducial samples to generate.

Value

`semiconttol.int` returns a list with items:

`ZIG.CI`	The generalized confidence interval under a ZIG distribution.
`ZIG.PI`	The generalized (upper) prediction limit under a ZIG distribution.
`ZIG.TI`	The generalized (upper) tolerance limit under a ZIG distribution.
`ZIG.TI.appx`	The generalized (upper) tolerance limit under a ZIG distribution based on the Wilson-Hilferty approximation.
`ZILN.CI`	The generalized confidence interval under a ZILN distribution.
`ZILN.PI`	The generalized (upper) prediction limit under a ZILN distribution.
`ZILN.TI`	The generalized (upper) tolerance limit under a ZILN distribution.
`ZILN.TI.appx`	The generalized (upper) tolerance limit under a ZILN distribution based on an approximation used in Hasan and Krishnamoorthy (2018).
`‘NA‘`	The number of times generalized fiducial quantities could not be calculated due to unlucky samples being drawn; e.g., a sample with all 0s. This will happen rarely and usually only when there is a very large proportion of zeros.

References

Hasan, M. S. and Krishnamoorthy, K. (2018), Confidence Intervals for the Mean and a Percentile Based on Zero-Inflated Lognormal Data, *Journal of Statistical Computation and Simulation*, **88**, 1499–1514.

Zou, Y. and Young, D. S. (2024), Fiducial-Based Statistical Intervals for Zero-Inflated Gamma Data, *Journal of Statistical Theory and Practice*, **18**, 1–20.

See Also

`fidbintol.int`, `fidnegbintol.int`, `fidpoistol.int`

Examples

```
## Generalized intervals assuming 95% confidence and
## 95% content for a dataset analyzed in Hasan and
## Krishnamoorthy (2018).

x <- c(6, 0, 6, 9, 6.5, 0, 0, 0, 1, 0.5, 2, 2, 0, 0, 1)
set.seed(1)
out <- semiconttol.int(x, P = 0.95, alpha = 0.05, N = 500)
out
```

`simnormtol.int` *Simultaneous Normal (or Log-Normal) Tolerance Intervals*

Description

Provides simultaneous 1-sided or 2-sided tolerance intervals for data distributed according to either a normal distribution or log-normal distribution.

Usage

```
simnormtol.int(x, alpha = 0.05, P = 0.99, side = 1,
               method = c("EXACT", "BONF"), m = 50, log.norm = FALSE)
```

Arguments

`x`	Either a matrix or list of vectors of the data. If a matrix, then the columns are the samples from the different normal (or log-normal) populations. If `method = "EXACT"`, then `x` must be a matrix.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether simultaneous 1-sided or 2-sided tolerance intervals are required (determined by `side = 1` or `side = 2`, respectively).

`method`	The method for calculating the k-factors. `"EXACT"` is an exact method that can be used when all `l` groups have the same sample size. `"BONF"` is an approximate method using the Bonferroni inequality, which can be used when the `l` groups have different sample sizes.
`m`	The maximum number of subintervals to be used in the `integrate` function. This is necessary only for `method = "EXACT"`. The larger the number, the more accurate the solution. Too low of a value can result in an error. A large value can also cause the function to be slow for `method = "EXACT"`.
`log.norm`	If `TRUE`, then the data is considered to be from a log-normal distribution, in which case the output gives tolerance intervals for the log-normal distribution. The default is `FALSE`.

Details

Recall that if the random variable X is distributed according to a log-normal distribution, then the random variable $Y = ln(X)$ is distributed according to a normal distribution.

Value

`normtol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`x.bar`	The sample means.
`1-sided.lower`	The simultaneous 1-sided lower tolerance bounds. This is given only if `side = 1`.
`1-sided.upper`	The simultaneous 1-sided upper tolerance bounds. This is given only if `side = 1`.
`2-sided.lower`	The simultaneous 2-sided lower tolerance bounds. This is given only if `side = 2`.
`2-sided.upper`	The simultaneous 2-sided upper tolerance bounds. This is given only if `side = 2`.

Note

The code for this functions is built upon code provided by Andrew Landgraf.

References

Krishnamoorthy, K. and Mathew, T. (2009), *Statistical Tolerance Regions: Theory, Applications, and Computation*, Wiley.

Mee, R. W. (1990), Simultaneous Tolerance Intervals for Normal Populations with Common Variance, *Technometrics*, **32**, 83-92.

See Also

`Normal`, `K.factor.sim`

Examples

```
## 95%/95% simultaneous 1-sided normal tolerance
## intervals for two samples of unequal size.

set.seed(100)
x <- list(rnorm(5,1),rnorm(7,1,2))
out <- simnormtol.int(x = x, alpha = 0.05, P = 0.95,
                      side = 1, method = "BONF")
out
```

`TwoParExponential` *The 2-Parameter Exponential Distribution*

Description

Density, distribution function, quantile function, and random generation for the 2-parameter exponential distribution with rate equal to `rate` and shift equal to `shift`.

Usage

```
d2exp(x, rate = 1, shift = 0, log = FALSE)
p2exp(q, rate = 1, shift = 0, lower.tail = TRUE, log.p = FALSE)
q2exp(p, rate = 1, shift = 0, lower.tail = TRUE, log.p = FALSE)
r2exp(n, rate = 1, shift = 0)
```

Arguments

`x,q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	The number of observations. If `length>1`, then the length is taken to be the number required.
`rate`	Vector of rates.
`shift`	Vector of shifts.
`log,log.p`	Logical vectors. If `TRUE`, then probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.

Details

If `rate` or `shift` are not specified, then they assume the default values of 1 and 0, respectively.

The 2-parameter exponential distribution has density

$$f(x) = \frac{1}{\beta}e^{(x-\mu)/\beta}$$

where $x \geq \mu$, μ is the shift parameter, and $\beta > 0$ is the scale parameter.

Value

d2exp gives the density, p2exp gives the distribution function, q2exp gives the quantile function, and r2exp generates random deviates.

See Also

runif and .Random.seed about random number generation.

Examples

```
## Randomly generated data from the 2-parameter exponential
## distribution.

set.seed(100)
x <- r2exp(n = 500, rate = 3, shift = -10)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 = sort(x)
y <- d2exp(x = x.1, rate = 3, shift = -10)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, p2exp(q = x.1, rate = 3, shift = -10), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")

q2exp(p = 0.20, rate = 3, shift = -10, lower.tail = FALSE)
q2exp(p = 0.80, rate = 3, shift = -10)
```

---

umatol.int *Uniformly Most Accurate Upper Tolerance Limits for Certain Discrete Distributions*

Description

Provides uniformly most accurate upper tolerance limits for the binomial, negative binomial, and Poisson distributions.

Usage

```
umatol.int(x, n = NULL, dist = c("Bin", "NegBin", "Pois"), N,
           alpha = 0.05, P = 0.99)
```

Arguments

x	A vector of data which is distributed according to one of the binomial, negative binomial, or Poisson distributions. If the length of x is 1, then it is assumed that this number is the sum of iid values from the assumed distribution.
n	The sample size of the data. If null, then n is calculated as the length of x.

dist	The distribution for the data given by x. The options are "Bin" for the binomial distribution, "NegBin" for the negative binomial distribution, and "Pois" for the Poisson distribution.
N	Must be specified for the binomial and negative binomial distributions. If dist = "Bin", then N is the number of Bernoulli trials and must be a positive integer. If dist = "NegBin", then N is the total number of successful trials (or dispersion parameter) and must be strictly positive.
alpha	The level chosen such that 1-alpha is the confidence level.
P	The proportion of the population to be covered by this tolerance interval.

Value

umatol.int returns a data frame with items:

alpha	The specified significance level.
P	The proportion of the population covered by this tolerance interval.
p.hat	The maximum likelihood estimate for the probability of success in each trial; reported if dist = "Bin".
nu.hat	The maximum likelihood estimate for the probability of success in each trial; reported if dist = "NegBin".
lambda.hat	The maximum likelihood estimate for the rate of success; reported if dist = "Pois".
1-sided.upper	The 1-sided upper tolerance limit.

References

Zacks, S. (1970), Uniformly Most Accurate Tolerance Limits for Monotone Likelihood Ratio Families of Discrete Distributions, *Journal of the American Statistical Association*, **65**, 307–316.

See Also

Binomial, NegBinomial, Poisson

Examples

```
## Examples from Zacks (1970).

umatol.int(25, n = 4, dist = "Bin", N = 10, alpha = 0.10,
           P = 0.95)
umatol.int(13, n = 10, dist = "NegBin", N = 2, alpha = 0.10,
           P = 0.95)
umatol.int(37, n = 10, dist = "Pois", alpha = 0.10, P = 0.95)
```

uniftol.int *Uniform Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to a uniform distribution.

Usage

```
uniftol.int(x, alpha = 0.05, P = 0.99, upper = NULL,
            lower = NULL, side = 1)
```

Arguments

`x`	A vector of data which is distributed according to a uniform distribution.
`alpha`	The level chosen such that `1-alpha` is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`upper`	The upper bound of the data. When `NULL`, then the maximum of `x` is used.
`lower`	The lower bound of the data. When `NULL`, then the minimum of `x` is used.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).

Value

`uniftol.int` returns a data frame with items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

References

Faulkenberry, G. D. and Weeks, D. L. (1968), Sample Size Determination for Tolerance Limits, *Technometrics*, **10**, 343–348.

Examples

```
## 90%/90% 1-sided uniform tolerance intervals for a sample
## of size 50 with a known lower bound of 0.

set.seed(100)
x <- runif(50, 0, 50)
out <- uniftol.int(x = x, alpha = 0.10, P = 0.90, lower = 0,
                   side = 1)
out

plottol(out, x, plot.type = "hist", side = "two",
        x.lab = "Uniform Data")
```

`ZipfMandelbrot` *Zipf-Mandelbrot Distributions*

Description

Density (mass), distribution function, quantile function, and random generation for the Zipf, Zipf-Mandelbrot, and zeta distributions.

Usage

```
dzipfman(x, s, b = NULL, N = NULL, log = FALSE)
pzipfman(q, s, b = NULL, N = NULL, lower.tail = TRUE,
         log.p = FALSE)
qzipfman(p, s, b = NULL, N = NULL, lower.tail = TRUE,
         log.p = FALSE)
rzipfman(n, s, b = NULL, N = NULL)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	The number of observations. If `length>1`, then the length is taken to be the number required.
`s, b`	The shape parameters, both of which must be greater than 0. `b` must be specified for Zipf-Mandelbrot distributions.
`N`	The number of categories, which must be integer-valued for Zipf and Zipf-Mandelbrot distributions. For a zeta distribution, `N = Inf` must be used.
`log, log.p`	Logical vectors. If `TRUE`, then the probabilities are given as `log(p)`.
`lower.tail`	Logical vector. If `TRUE`, then probabilities are $P[X \leq x]$, else $P[X > x]$.

Details

The Zipf-Mandelbrot distribution has mass

$$p(x) = \frac{(x+b)^{-s}}{\sum_{i=1}^{N}(i+b)^{-s}},$$

where $x = 1, \ldots, N$, `s,b>0` are shape parameters, and `N` is the number of distinct categories. The Zipf distribution is just a special case of the Zipf-Mandelbrot distribution where the second shape parameter `b=0`. The zeta distribution has mass

$$p(x) = \frac{x^{-s}}{\zeta(s)},$$

where $x = 1, 2, \ldots$, `s>1` is the shape parameter, and $\zeta()$ is the Riemann zeta function given by:

$$\zeta(t) = \sum_{i=1}^{\infty} \frac{1}{i^t} < \infty.$$

Note that the zeta distribution is just a special case of the Zipf distribution where `s>1` and `N` goes to infinity.

Value

`dzipfman` gives the density (mass), `pzipfman` gives the distribution function, `qzipfman` gives the quantile function, and `rzipfman` generates random deviates for the specified distribution.

Note

These functions may be updated in a future version of the package so as to allow greater flexibility with the inputs.

References

Mandelbrot, B. B. (1965), Information Theory and Psycholinguistics. In B. B. Wolman and E. Nagel, editors. *Scientific Psychology*, Basic Books.

Young, D. S. (2013), Approximate Tolerance Limits for Zipf-Mandelbrot Distributions, *Physica A: Statistical Mechanics and its Applications*, **392**, 1702–1711.

Zipf, G. K. (1949), *Human Behavior and the Principle of Least Effort*, Hafner.

Zornig, P. and Altmann, G. (1995), Unified Representation of Zipf Distributions, *Computational Statistics and Data Analysis*, **19**, 461–473.

See Also

`runif` and `.Random.seed` about random number generation.

Examples

```
## Randomly generated data from the Zipf distribution.

set.seed(100)
x <- rzipfman(n = 150, s = 2, N = 100)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- dzipfman(x = x.1, s = 2, N = 100)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pzipfman(q = x.1, s = 2, N = 100), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")

qzipfman(p = 0.20, s = 2, N = 100, lower.tail = FALSE)
qzipfman(p = 0.80, s = 2, N = 100)

## Randomly generated data from the Zipf-Mandelbrot distribution.

set.seed(100)
x <- rzipfman(n = 150, s = 2, b = 3, N = 100)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- dzipfman(x = x.1, s = 2, b = 3, N = 100)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pzipfman(q = x.1, s = 2, b = 3, N = 100), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")

qzipfman(p = 0.20, s = 2, b = 3, N = 100, lower.tail = FALSE)
qzipfman(p = 0.80, s = 2, b = 3, N = 100)

## Randomly generated data from the zeta distribution.

set.seed(100)
x <- rzipfman(n = 100, s = 1.3, N = Inf)
hist(x, main = "Randomly Generated Data", prob = TRUE)

x.1 <- sort(x)
y <- dzipfman(x = x.1, s = 1.3, N = Inf)
lines(x.1, y, col = 2, lwd = 2)

plot(x.1, pzipfman(q = x.1, s = 1.3, N = Inf), type = "l",
     xlab = "x", ylab = "Cumulative Probabilities")

qzipfman(p = 0.20, s = 1.3, lower.tail = FALSE, N = Inf)
qzipfman(p = 0.80, s = 1.3, N = Inf)
```

`zipftol.int` *Zipf-Mandelbrot Tolerance Intervals*

Description

Provides 1-sided or 2-sided tolerance intervals for data distributed according to Zipf, Zipf-Mandelbrot, and zeta distributions.

Usage

```
zipftol.int(x, m = NULL, N = NULL, alpha = 0.05, P = 0.99,
            side = 1, s = 1, b = 1, dist = c("Zipf",
            "Zipf-Man", "Zeta"), ties = FALSE, ...)
```

Arguments

`x`	A vector of raw data or a table of counts which is distributed according to a Zipf, Zipf-Mandelbrot, or zeta distribution. Do not supply a vector of counts!
`m`	The number of observations in a future sample for which the tolerance limits will be calculated. By default, `m = NULL` and, thus, `m` will be set equal to the original sample size.
`N`	The number of categories when `dist = "Zipf"` or `dist = "Zipf-Man"`. This is not used when `dist = "Zeta"`. If `N = NULL`, then `N` is estimated based on the number of categories observed in the data.
`alpha`	The level chosen such that 1-alpha is the confidence level.
`P`	The proportion of the population to be covered by this tolerance interval.
`side`	Whether a 1-sided or 2-sided tolerance interval is required (determined by `side = 1` or `side = 2`, respectively).
`s`	The initial value to estimate the shape parameter in the `zm.ll` function.
`b`	The initial value to estimate the second shape parameter in the `zm.ll` function when `dist = "Zipf-Man"`.
`dist`	Options are `dist = "Zipf"`, `dist = "Zipf-Man"`, or `dist = "Zeta"` if the data is distributed according to the Zipf, Zipf-Mandelbrot, or zeta distribution, respectively.
`ties`	How to handle if there are other categories with the same frequency as the category at the estimated tolerance limit. The default is `FALSE`, which does no correction. If `TRUE`, then the highest ranked (i.e., lowest number) of the tied categories is selected for the lower limit and the lowest ranked (i.e., highest number) of the tied categories is selected for the upper limit.
`...`	Additional arguments passed to the `zm.ll` function, which is used for maximum likelihood estimation.

Details

Zipf-Mandelbrot models are commonly used to model phenomena where the frequencies of categorical data are approximately inversely proportional to its rank in the frequency table. Zipf-Mandelbrot distributions are heavily right-skewed distributions with a (relatively) large mass placed on the first category. For most practical applications, one will typically be interested in 1-sided upper bounds.

Value

`zipftol.int` returns a data frame with the following items:

`alpha`	The specified significance level.
`P`	The proportion of the population covered by this tolerance interval.
`s.hat`	MLE for the shape parameter `s`.
`b.hat`	MLE for the shape parameter `b` when `dist = "Zipf-Man"`.
`1-sided.lower`	The 1-sided lower tolerance bound. This is given only if `side = 1`.
`1-sided.upper`	The 1-sided upper tolerance bound. This is given only if `side = 1`.
`2-sided.lower`	The 2-sided lower tolerance bound. This is given only if `side = 2`.
`2-sided.upper`	The 2-sided upper tolerance bound. This is given only if `side = 2`.

Note

This function may be updated in a future version of the package so as to allow greater flexibility with the inputs.

References

Mandelbrot, B. B. (1965), Information Theory and Psycholinguistics. In B. B. Wolman and E. Nagel, editors. *Scientific Psychology*, Basic Books.

Young, D. S. (2013), Approximate Tolerance Limits for Zipf-Mandelbrot Distributions, *Physica A: Statistical Mechanics and its Applications*, **392**, 1702–1711.

Zipf, G. K. (1949), *Human Behavior and the Principle of Least Effort*, Hafner.

Zornig, P. and Altmann, G. (1995), Unified Representation of Zipf Distributions, *Computational Statistics and Data Analysis*, **19**, 461–473.

See Also

`ZipfMandelbrot`, `zm.ll`

Examples

```
## 95%/99% 1-sided tolerance intervals for the Zipf,
## Zipf-Mandelbrot, and zeta distributions.

set.seed(100)

s <- 2
b <- 5
N <- 50

zipf.data <- rzipfman(n = 150, s = s, N = N)
zipfman.data <- rzipfman(n = 150, s = s, b = b, N = N)
zeta.data <- rzipfman(n = 150, s = s, N = Inf)

out.zipf <- zipftol.int(zipf.data, dist = "Zipf")
out.zipfman <- zipftol.int(zipfman.data, dist = "Zipf-Man")
out.zeta <- zipftol.int(zeta.data, N = Inf, dist = "Zeta")

out.zipf
out.zipfman
out.zeta
```

---

zm.ll *Maximum Likelihood Estimation for Zipf-Mandelbrot Models*

Description

Performs maximum likelihood estimation for the parameters of the Zipf, Zipf-Mandelbrot, and zeta distributions.

Usage

```
zm.ll(x, N = NULL, s = 1, b = 1, dist = c("Zipf", "Zipf-Man",
      "Zeta"), ...)
```

Arguments

x	A vector of raw data or a table of counts which is distributed according to a Zipf, Zipf-Mandelbrot, or zeta distribution. Do not supply a vector of counts!
N	The number of categories when `dist = "Zipf"` or `dist = "Zipf-Man"`. This is not used when `dist = "Zeta"`. If `N = NULL`, then `N` is estimated based on the number of categories observed in the data.
s	The initial value to estimate the shape parameter, which is set to 1 by default. If a poor initial value is specified, then a `WARNING` message is returned.
b	The initial value to estimate the second shape parameter when `dist = "Zipf-Man"`, which is set to 1 by default. If a poor initial value is specified, then a `WARNING` message is returned.
dist	Options are `dist = "Zipf"`, `dist = "Zipf-Man"`, or `dist = "Zeta"` if the data is distributed according to the Zipf, Zipf-Mandelbrot, or zeta distribution, respectively.
...	Additional arguments passed to the `mle` function.

Details

Zipf-Mandelbrot models are commonly used to model phenomena where the frequencies of categorical data are approximately inversely proportional to its rank in the frequency table.

Value

See the help file for `mle` to see how the output is structured.

Note

This function may be updated in a future version of the package so as to allow greater flexibility with the inputs.

References

Mandelbrot, B. B. (1965), Information Theory and Psycholinguistics. In B. B. Wolman and E. Nagel, editors. *Scientific Psychology*, Basic Books.

Zipf, G. K. (1949), *Human Behavior and the Principle of Least Effort*, Hafner.

Zornig, P. and Altmann, G. (1995), Unified Representation of Zipf Distributions, *Computational Statistics and Data Analysis*, **19**, 461–473.

See Also

`mle`, `ZipfMandelbrot`

Examples

```
## Maximum likelihood estimation for randomly generated data
## from the Zipf, Zipf-Mandelbrot, and zeta distributions.

set.seed(100)

s <- 2
b <- 5
N <- 50

zipf.data <- rzipfman(n = 500, s = s, N = N)
out.zipf <- zm.ll(zipf.data, N = N, dist = "Zipf")
stats4::coef(out.zipf)
stats4::vcov(out.zipf)

zipfman.data <- rzipfman(n = 500, s = s, b = b, N = N)
out.zipfman <- zm.ll(zipfman.data, N = N, dist = "Zipf-Man")
stats4::coef(out.zipfman)
diag(stats4::vcov(out.zipfman))

zeta.data <- rzipfman(n = 200, s = s, N = Inf)
out.zeta <- zm.ll(zeta.data, N = Inf, dist = "Zeta")
stats4::coef(out.zeta)
stats4::vcov(out.zeta)
```

Index