Package: ICRanks (via r-universe)

September 13, 2024

Type Package

Title Simultaneous Confidence Intervals for Ranks

Version 3.1

Date 2019-06-21

Author Diaa Al Mohamad and Erik W. van Zwet and Jelle J. Goeman

Maintainer Diaa Al Mohamad <diaa.almohamad@gmail.com>

Description Algorithms to construct simultaneous confidence intervals for the ranks of means mu_1,...,mu_n based on an independent Gaussian sample using multiple testing techniques.

SystemRequirements C++11

License GPL (>= 2)

Imports Rcpp (>= 0.12.11), multcomp, gmp

LinkingTo Rcpp

RoxygenNote 6.0.1.9000

NeedsCompilation yes

Date/Publication 2019-06-21 22:00:58 UTC

Repository https://almohamad.r-universe.dev

RemoteUrl https://github.com/cran/ICRanks

RemoteRef HEAD

RemoteSha a76653c7cd9ed94008eea925f8b3f4de6d31629f

Contents

ic.ranks 2

Index **6**

`ic.ranks` *Confidence intervals for ranks*

Description

This function calculates simultaneous confidence (sets) intervals (CIs) at a pre-specified level (1-alpha) for the ranks of centers mu_1,...,mu_n which are observed through a sample y using multiple testing techniques. Several possibilities are presented through a "Method" variable. There are bascially two main choices; one which uses the partitioing principle and the likelihood ratio test and the the other is based on Tukey's pairwise comparison procedure. See choices below, and for more details see the references.

Usage

```
ic.ranks(y, sigma = rep(1, length(y)), Method = c("ExactLR", "BoundLR",
  "Tukey", "SeqTukey", "ApproximateLR", "TukeyNoTies", "RescaledExactLR",
  "RescaledTukey"), BoundChoice = c("Upper", "Lower"),
  ApproxAlgo = c("Exact", "Upper"), alpha = 0.05, control = list(crit
  = NULL, trace = TRUE, adjustL = FALSE, adjustU = FALSE, n_adjust =
  length(y) - 1, N = 10^4, MM = 10^3, gridSize = 5, RandPermut = 0,
  SwapPerm = TRUE))
```

Arguments

`y`	a real vector of observed data.
`sigma`	a vector of standard deviations. If sigma is a single value, then we consider that all centers have the same standard deviation.
`Method`	a character indicating the method used to produce the confidence intervals. The "ExactLR" produces confidence intervals using the partitioning principle and the likelihood ratio test. The "BoundLR" choice produces lower- or upper-bound confidence intervals (according to the "BoundChoice") for the ranks using a fast algorithm. The "Tukey" choice produces simultaneous confidence intervals for the ranks using Tukey's HSD. The "SeqTukey" produces simultaneous confidence intervals for the ranks using a sequential-rejective algorithm. The "Approximate" choice provides approximate confidence intervals which are shorter than the exact ones by considering a subset of the partitions (the correctly ordered ones, see refs and below for details). The "TukeyNoTies" choice calculates a readustement for Tukey's method under the assumption that there are no ties and then use Tukey's method again with adjusted level. The "RescaledExactLR" choice calculates a readustement for the "ExactLR" method by adjusting each and every local test. The "RescaledTukey" choice calculates a readustement for the "Tukey" method by pluging it into a partitioning procedure and then adjusting each and every local test.
`BoundChoice`	a character entry which is only relevant if the "Bound" choice is picked in the Method parameter. The default value is "Upper" which results in the upper-bound CIs for the ranks. If "Lower" is chosen, then the lower-bound CIs are generated.

`ApproxAlgo`	a character entry ("Upper" by default). This parameter controls which approximation is to be used.
`alpha`	the significance level of the internal tests we perform (which corresponds to the FWER control of the corresponding multiple testing procedure). CIs have simultaneous significance level of 1-alpha.
`control`	is a list of control parameters.
`crit`	is the critical value for Tukey's HSD. If it is kept NULL, then it is calculated internally. The use of this parameter becomes handful in case the user wishes to make several simulations. By providing it, we avoid repeating a Monte-Carlo estimation of the quantile and thus we gain in execution time. In some cases (espcially when all centers have the same standard deviation), the critical value for Tukey's HSD can be found in some statistical tables.
`trace`	a logical parameter which supresses the printing of the details of the method which was chosen. The default is TRUE (shows details).
`adjustL`	a logical variable (default to FALSE) indicating if an adjustment on the lower bound according to the data must be considered (if possible). This choice is only relevenat if Method is chosen as "BoundLR" and BoundChoice is chosen as "Lower".
`adjustU`	a logical variable (default to FALSE) which gives the user the choice to adjust the upper bound CIs through the parameter "n_adjust". This choice is only relevenat if Method is chosen as "BoundLR" and BoundChoice is chosen as "Upper".
`n_adjust`	an integer-valued entry for advanced control over the lower- or upper-bound algorithms. When the "adjustL" parameter is TRUE, the new value of n_adjust is chosen automatically as the best adjustment on the lower affine bound of the chi-square quantiles according to the data. If adjustU is TRUE, then n_adjust contains the point at which the upper affine bound is tangent on the chi-square quantiles. Possible values 1,...,n-1. If both adjustL and adjustU variables are left FALSE, then the default choice is that the lower affine bound passes between the chi-square quantiles at 1 and n-1 degrees of freedom, and the upper affine bound is tangent on n-1.
`N`	the number of iterations used in order to calculate the Studentized range quantile for Tukey's algorithms.
`MM`	the number of Monte-Carlo simultations required to estimate the (simultaneous) coverage. This is used in all rescaling methods.
`RandPermut`	is the number of additional permutations to perform when using either the "ExactLR" or the "BoundLR" algorithms and only when the standard deviations are different. When the standard deviations are the same, this has no influence on the result.
`SwapPerm`	corresponds to performing swap permutations (yes = TRUE, no = FALSE). This is used in all the methods except for "Tukey" and "ExactLR" (the latter when the standard deviations are not the same).

Details

The vector of observations needs to be sorted. Otherwise, it is done internally. The observations are supposed to be independent realizations of Guassian distributions with unknown centers mu_1,...,mu_n and known standard deviations sigma = (sigma_1,...,sigma_n).

The exact-partitioning confidence intervals (option "ExactLR") are calculated using an algorithm with exponential complexity. The hypotheses in each level of the partitioning are coded using the combinatorial number system.

The lower- and upper-bound CIs are calculated with a polynomial algorithm. The bracketing obtained from the lower and upper bounds is generally very narrow with a maximum gap of 1. Moreover, in regular situations, the lower and upper bounds coincide on at least 50 percent of the centers yielding the exact-partitioning result. Thus, the bracketing is an alternative for an exact-partitioning algorithm for medium- and large-size samples (n>50). When a calculus of the lower- and upper-bound CIs is required, the default choice is when no adjustment on neither the lower nor the upper bounds is taken into account. Thus, the lower affine bound of the chi-square is a line passing by the quantiles at 1 and n-1 degrees of freedom, whereas the upper affine bound is a line tangent on the chi-square quantiles at n-1 degrees of freedom. The adjustment on the lower bound CIs can in some contexts improve on the CIs and increase the number of centers where the lower and upper bounds coincide. The best option is to adjust for both the lower and upper bounds (separately).

Both "Tukey" and "SeqTukey" are based on multiple comparison testing and are superior to the LR-based CIs if the centers are far apart from each other and if the standard deviations are not significantly different from each other. The sequential rejective variant of Tukey's HSD rejects at least as much as Tukey's HSD and thus produces generally shorter confidence intervals for the ranks.

The "TukeyNoTies" method assumes that the true vector of parameters has no ties and therefore, instead of calculating a quantile q corresponding to mu=0 with set rank [1,n] for mu_i, we calculate a quantile corresponding to mu=0 with rank {i} for mu_i. The method provides shorter SCI for the ranks but is still conservative.

When the standard deviations are not the same for all the means, the methods based on the partitioning principle are not guaranteed to produce the same results. The "Block" algorithm, however, is always compatible with the lower and upper CIs provided by option "BoundLR". When the number of means exceeds 10, then performing any method based on the partitioning procedure requires a long execution time since the complexity of the algorithm is super exponential of order exp(exp(n)).

When the standard deviations are not the same the approximate methods based on the LRT are not guaranteed to cover and if the standard deviations are very different, the resulting SCIs are anticonservative. If the standard deviations are close to each other, then the result is still conservative.

In terms of execution time. The Tukey method is the fastest. It can be used always. The methods based on the partitioning principle have all exponential complexity. Therefore, when the standard deviations are the same, the "ExactLR" would produce results up to 40 means. When they are not the same, no method based on the partitioning principle can be used for more than 10 means unless we limit the number of random permutations that we use which in case of great differences in the standard deviations might lead to anticonservative results. More details can be found in the references.

Value

a list of two vectors containing the lower and upper bounds of the confidence intervals for the sorted observed centers.

Author(s)

Diaa Al Mohamad and Jelle J. Goeman and Erik W. van Zwet. Correspondence can be made to diaa.almohamad@gmail.com

References

Diaa Al Mohamad and Erik W. van Zwet and Jelle J. Goeman and Aldo Solari, Simultaneous confidence sets for ranks using the partitioning principle - Technical report (2017). https://arxiv.org/abs/1708.02729

Diaa Al Mohamad and Jelle J. Goeman and Erik W. van Zwet, An improvement of Tukey's HSD with application to ranking institutions (2017). https://arxiv.org/abs/1708.02428

Diaa Al Mohamad and Jelle J. Goeman and Erik W. van Zwet, Simultaneous Confidence Intervals for Ranks With Application to Ranking Institutions (2018). https://arxiv.org/abs/1812.05507

Examples

```
n = 5; TrueCenters = 1:n
alpha = 0.05; sigma = rep(0.5,n)
y = as.numeric(sapply(1:n, function(ll) rnorm(1,TrueCenters[ll],sd=sigma[ll])))
ind = sort.int(y, index.return = TRUE)$ix
y = y[ind]
sigma = sigma[ind] # The sigmas need to follow the order of the y's
res = ic.ranks(y, sigma, Method = "ExactLR",alpha = 0.05, control = list(trace = TRUE))
LowerExact = res$Lower; UpperExact = res$Upper
#res = ic.ranks(y, sigma, Method = "BoundLR", BoundChoice = "Lower",
#   control = list(adjustL = FALSE, adjustU = FALSE))
#LowerL = res$Lower; UpperL = res$Upper
#res = ic.ranks(y, sigma, Method = "BoundLR", BoundChoice = "Upper",
#   control = list(adjustL = FALSE, adjustU = FALSE, trace=FALSE))
#LowerU = res$Lower; UpperU = res$Upper
res = ic.ranks(y, sigma, Method = "Tukey")
LowerTuk = res$Lower; UpperTuk = res$Upper
res = ic.ranks(y, sigma, Method = "SeqTukey")
LowerTukSeq = res$Lower; UpperTukSeq = res$Upper
res = ic.ranks(y, sigma, Method = "TukeyNoTies")
LowerTukNoTies = res$Lower; UpperTukNoTies = res$Upper
resLR1 = ic.ranks(y, sigma, Method = "RescaledExactLR", alpha = alpha,
  control = list(trace = TRUE, gridSize = 4, MM = 100, RandPermut=factorial(n)))
LowerExact
#LowerL
#LowerU
LowerTuk
resLR1$Lower
resLR1$Upper
```

Index

ic.ranks, 2