Journal of Modern Applied Statistical Methods

Volume 18 | Issue 2 Article 1

7-17-2020

JMASM 52: Extremely Efficient Permutation and Bootstrap Hypothesis Tests Using R

Christina Chatzipantsiou
University of Crete, Greece, chatzipantsiou@gmail.com

Marios Dimitriadis
University of Crete, Greece, kmdimitriadis@gmail.com

Manos Papadakis
University of Crete, Greece, papadakm95@gmail.com

Michail Tsagris
University of Crete, Greece, mtsagris@yahoo.gr

Recommended Citation

Chatzipantsiou, C., Dimitriadis, M., Papadakis, M., & Tsagris, M. (2019). JMASM 52: Extremely efficient permutation and bootstrap hypothesis tests using R. Journal of Modern Applied Statistical Methods, 18(2), eP2898. doi: 10.22237/jmasm/1604189940

Journal of Modern Applied Statistical Methods
November 2019, Vol. 18, No. 2, eP2898.
doi: 10.22237/jmasm/1604189940

בס"ד

ISSN 1538 − 9472

JMASM 52: Extremely Efficient Permutation and Bootstrap Hypothesis Tests Using R

Christina Chatzipantsiou
University of Crete
Herakleion, Greece

Marios Dimitriadis
University of Crete
Herakleion, Greece

Manos Papadakis
University of Crete
Herakleion, Greece

Michail Tsagris
University of Crete
Rethymnon, Greece

Re-sampling based statistical tests are known to be computationally heavy, but reliable when small sample sizes are available. Despite their nice theoretical properties not much effort has been put to make them efficient. Computationally efficient method for calculating permutation-based *p*-values for the Pearson correlation coefficient and two independent samples *t*-test are proposed. The method is general and can be applied to other similar two sample mean or two mean vectors cases.

Keywords: Computational efficiency, permutation, bootstrap, hypothesis testing

Introduction

Computer intensive techniques, such as MCMC, bootstrap and permutation, are known to require computational power. There has been too much of research effort on trying to make MCMC more efficient, either by means of programming or statistical theory. Bootstrap and permutation are on the other hand are not that heavy. Both of them are very useful when performing simulation studies or even analyzing real datasets with small sample sizes and strange shapes of distributions (mixtures for example), or with statistics whose distribution is not known or is too complex to work with.

Two examples where the necessity for, efficient, computational statistics is apparent, are Bayesian networks (Neapolitan, 2003) and feature selection

doi: 10.22237/jmasm/1604189940 | Accepted: June 27, 2018; Published: July 17, 2020.
Correspondence: Michail Tsagris, mtsagris@yahoo.gr

algorithms (Tsamardinos et al., 2003). When applied to biological data, which usually have a small number of observations, re-sampling techniques is an advisable strategy. Pearson correlation is a cornerstone in Bayesian network learning (Neapolitan, 2003). Welch's *t*-test (Welch, 1938) on the other hand is very frequently utilized in gene expression data to discover which genes are differentially expressed, or to compare between (unmatched) case control samples. Other cases include multivariate statistics, which involve calculation of statistical functions whose distribution is not known. One such example is distance correlation (Szekely et al., 2007). Hypothesis testing procedures for multivariate data require large sample sizes in order to have a valid behavior of the type I error. (Tsagris et al., 2017).

With a focus on univariate statistics, Neto (2015a) proposed a vectorized function in R (R Core Team, 2017), as an alternative to the "for" function, to speed up the calculation of the bootstrap *p*-value of the hypothesis test of zero correlation. Comparing this permutation approach with other bootstrap functions available in other R packages indicated there were dramatic reduction in execution runtime. Neto (2015a) extended the bootstrap version of the two sample Welch's *t*-test, among others.

Similarly, a method is proposed here for a fast bootstrap or permutation-based *p*-value calculation. The method is efficient even for a large number of permutations or bootstrap samples (high CPU to data-size ratio tasks). It outperforms Neto's (2015a) vectorization method, showcasing a notable reduction in execution time even by an order of magnitude in some cases (Figure 3). The focus here is on Pearson correlation and two independent samples *t*-test. However, due to the method being simple, it is also extended to the James multivariate version of the *t*-test (James, 1954).

Bootstrap or Permutation Calibration

Pearson Correlation Coefficient

Given a sample n of paired observations (x_i, y_i), $i = 1,\ldots, n$, the sample Pearson correlation coefficient is calculated as

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}. \quad (1)$$

In order to test whether the true correlation, ρ, is equal to 0, the relevant test statistic is given by

$$Z = 0.5 \log\left(\frac{1+r}{1-r}\right)\sqrt{n-3}\,. \tag{2}$$

If $\rho = 0$, $Z \sim \mathrm{N}(0, 1)$ asymptotically, and for small n, N(0, 1) can be substituted by t_{n-3}.

Welch's *t*-Test

The two independent samples *t*-test is used to check whether the means of two populations, out of which the samples have been drawn are equal (null hypothesis). The main and rather strong assumption is that the variances are the same. Welch (1938) proposed an alternative to the *t*-test for the case of unequal variances, whose test statistic is given by

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}, \tag{3}$$

where $\bar{x}_1$ and $\bar{x}_2$ are the two sample means, s_1^2 and s_2^2 are the two sample variances, and n_1 and n_2 are the two sample sizes. The test statistic (3) is compared against a *t* distribution with some properly estimated degrees of freedom (Satterthwaite, 1946; Welch, 1947).

James Multivariate *t*-Test

Also consider the multivariate two independent samples *t*-test, namely the James test (James, 1954), which is the analogue of Welch's *t*-test. Similarly to Welch's *t*-test, James test makes no assumptions about the covariance matrices of the two populations from which the samples were sampled. The test statistic for two *d*-dimensional samples is

$$T_u^2 = \left(\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2\right)^{\mathrm{T}} \left(\frac{\mathbf{S}_1^2}{n_1} + \frac{\mathbf{S}_2^2}{n_2}\right)^{-1} \left(\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2\right), \tag{4}$$

where $\bar{\mathbf{X}}_1$ and $\bar{\mathbf{X}}_2$ are the two sample mean vectors, $\mathbf{S}_1^2$ and $\mathbf{S}_2^2$ are the two sample covariance matrices, and n_1 and n_2 are the two sample sizes. The test statistic (4) is compared against a corrected X^2 distribution (James, 1954).

Permutation- and Bootstrap-Based *p*-Values

As can be seen in (1), Pearson correlation coefficient is a function of two vectors of paired observations. The usual permutation-based *p*-value reorders the values of *x* or of *y*, thus changing the pairs (Legendre & Legendre, 2012; Berry et al., 2016). For every reordering of the values, (2) is calculated. This method is implemented in R by using a "for" loop. Vectorization is a faster alternative, if the number of permutations is the most used in practice, 999, otherwise if this number is 9999 a "for" loop might be a better option, for R, because of larger memory requirements and the need to handle matrices efficiently.

In the context of Bayesian network learning (Neapolitan, 2003), Tsamardinos and Borboudakis (2010) suggested a different approach with high computational savings with the G^2 test of independence for categorical data. Instead of permuting the data *B* (e.g. 999 times) times, they calculated the test statistic using 100 permutations. Their average is used as an estimator of the degrees of freedom of the x^2 distribution. They also suggest performing half of the permutations and estimate the probability of having a *p*-value less than the given significance level. Similarly, it is possible to stop the permutations once the proportion of times the permuted test statistic value exceeds the significance level.

Neto's Vectorized Bootstrap *p*-Value

Neto (2015a) proposed a computationally less heavy calculation of the *p*-value when using non-parametric bootstrap. Let n_i^* represent the number of times observation x_i is selected in the bootstrap sample X^* and $w_i^* = n_i^* / N$, where N is the sample size. Then, the category counts, $\mathbf{n}^* = \left(n_1^*, \ldots, n_N^*\right)$ of the bootstrap sample x^* are distributed according to the multinomial distribution

$$\mathbf{n}^* = N\mathbf{w}^* \sim \text{Multinomial}\left(N, N^{-1}\mathbf{J}_N\right), \tag{5}$$

where $\mathbf{J}_N = (1,\ldots, 1)^{\mathrm{T}}$ is the unit vector of size N.

As an example, Neto (2015a) described the case of the sample mean $\hat{\theta} = \sum_{i=1}^{N} x_i / N$.

1. Draw B bootstrap count vectors $\mathbf{n}^*$ from (5) using the command "rmultinom" in R.
2. Divide $\mathbf{n}^*$ by N to obtain the $N \times B$ bootstrap weights matrix $\mathbf{W}^*$.
3. Generate the vector of bootstrap means as $\hat{\boldsymbol{\theta}}_{\text{boot}} = \mathbf{x}^{\mathrm{T}}\mathbf{W}^*$.

For the case of the Pearson correlation coefficient, the vector with the bootstrap correlations is given by

$$\mathbf{r} = \frac{(\mathbf{x}\cdot\mathbf{y})^{\mathrm{T}}\,\mathbf{W}^* - \left(\mathbf{x}^{\mathrm{T}}\mathbf{W}^*\right)\cdot\left(\mathbf{y}^{\mathrm{T}}\mathbf{W}^*\right)}{\sqrt{\left(\mathbf{x}^2\right)^{\mathrm{T}}\,\mathbf{W}^* - \left(\mathbf{x}^{\mathrm{T}}\mathbf{W}^*\right)^2}\cdot\sqrt{\left(\mathbf{y}^2\right)^{\mathrm{T}}\,\mathbf{W}^* - \left(\mathbf{y}^{\mathrm{T}}\mathbf{W}^*\right)^2}} \tag{6}$$

where $\mathbf{x}\cdot\mathbf{y} = (x_1y_1,\ldots, x_ny_n)^{\mathrm{T}}$ is the element-wise multiplication of two vectors.

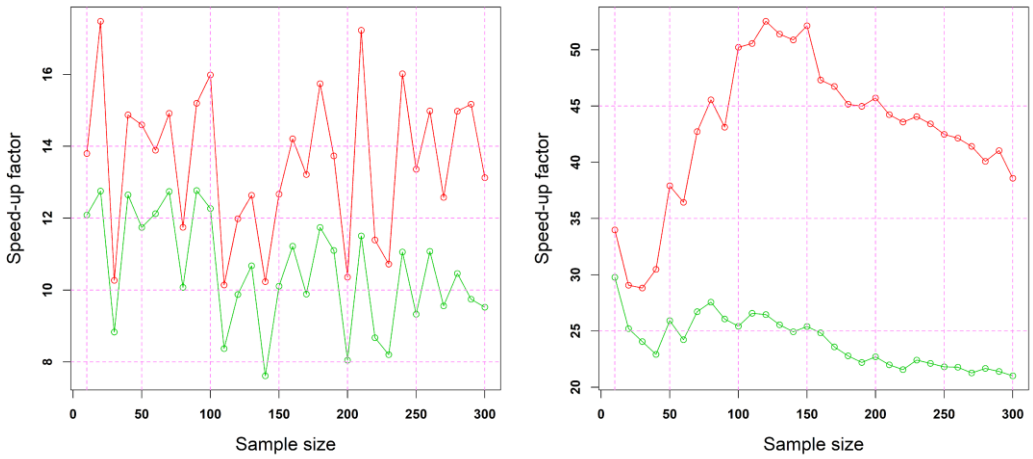
Efficient Calculation of the Permutation and Bootstrap *p*-Values

This efficient permutation and bootstrap based *p*-values, unlike the former strategies, performs nearly all B permutations. It is currently developed for the Pearson correlation and Welch's *t*-test. For the Pearson correlation (permutation case) it reorders the values of each vector $\left[\sqrt{B}\right]$ times, where [.] denotes the rounding operation. The same idea is used for Welch's *t*-test (bootstrap case), but with the difference that, for each vector, $\left[\sqrt{B}\right]$ bootstrap samples are taken. The process is similar for the James test. We create $\left[\sqrt{B}\right]$ samples from each sample and store their mean vectors and covariance matrices. The test statistic is then calculated for all $\left[\sqrt{B}\right]^2 \approx B$ combinations of the permuted or bootstrap samples. This results in substantial computational gains and the extra benefit is that similar ideas can be used in other settings, for example in calculating the permutation *p*-value of the distance correlation (Szekely et al., 2007).

Time Comparisons and Statistical Validation

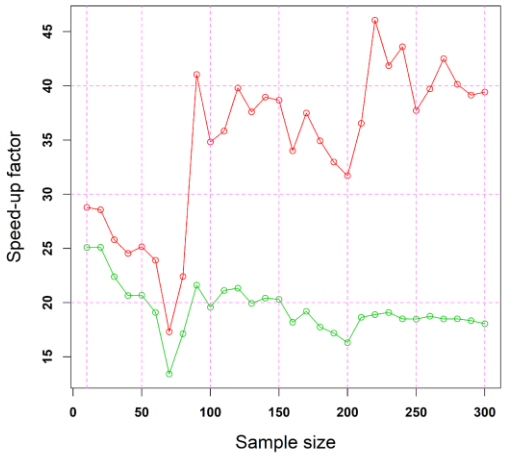
For the Pearson correlation and Welch's *t*-test scenarios, the sample sizes ranged from 10 to 300 in steps of 10 and 5 different numbers of permutations or bootstrap samples B = (999, 4999, 9999, 14999, 19999). For each combination (180 in total), vectors were sampled 10 times and the elapsed time was estimated using the package "microbenchmark" (Mersmann, 2015) with 100 repetitions; Neto's vectorized bootstrap R code is available via Neto (2015b). For the James test, the

sample sizes ranged from 10 up to 100 and the bootstrap samples were B = (999, 4999).

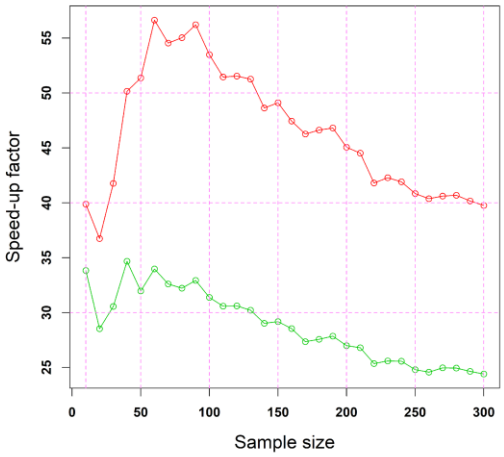


(a) B = 999

(b) B = 9,999



(c) B = 4,999



(d) B = 19,999

Figure 1. Correlation coefficient; speed-up factors of some standard methods against the new version; higher numbers indicate higher computational cost; the green line is with "replicate" while the red line is with "for"

Vectorized Bootstrap Versus Efficient Permutation of Pearson Correlation Coefficient

The computational times of the proposed method versus using two functions with a "for" loop was compared in R using the command "replicate." The relative performance improvements in terms of runtime are compiled in Figure 1.

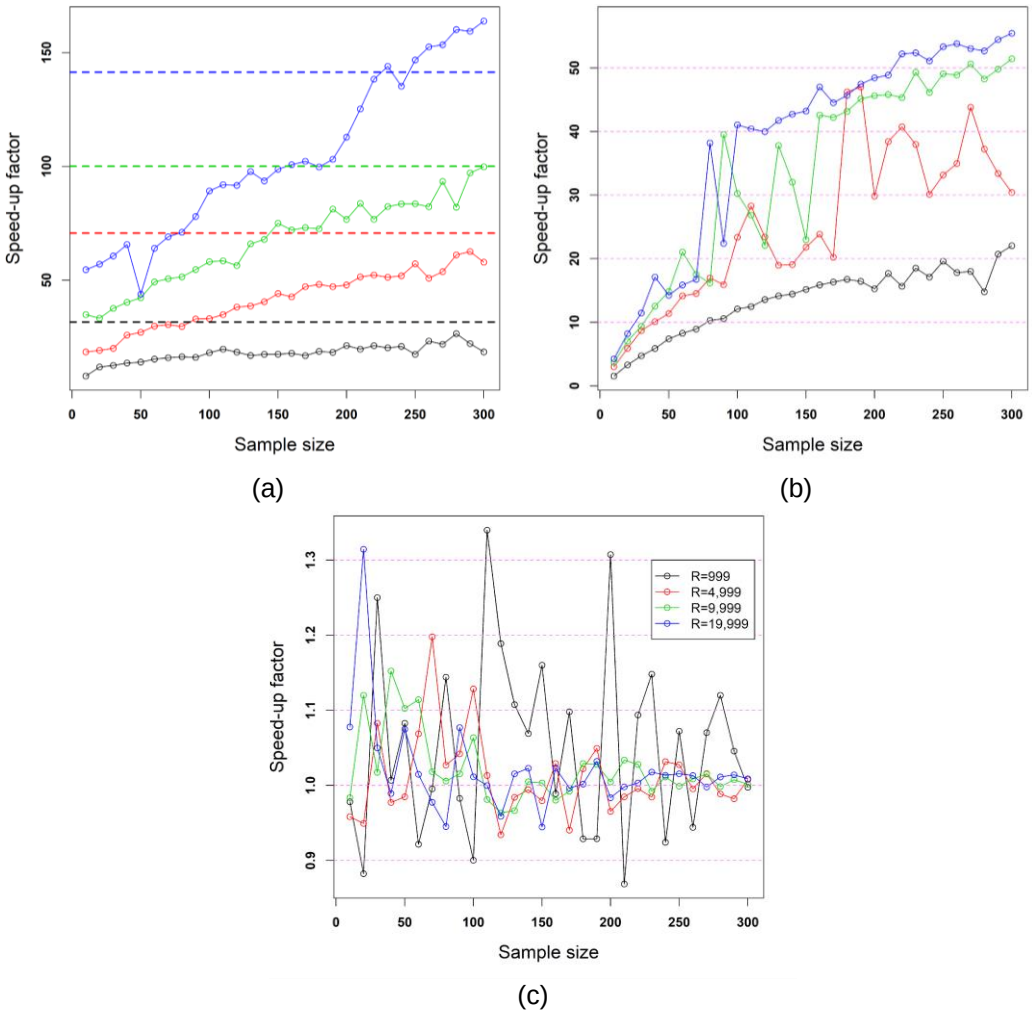


Figure 2. Correlation coefficient; (a) speed-up factors of the asymptotic *p*-value versus the permutation *p*-value; this is an estimate of how much slower the permutation *p*-value is, relative to the asymptotic *p*-value; (b) speed up factor of the vectorized bootstrap (Neto, 2015a) versus our permutation implementation; numbers greater than 1 indicate that our version is faster; (c) speed up factor of the C++ implementation versus the R implementation; this is an estimate of how slower R is in comparison to C++ for this method

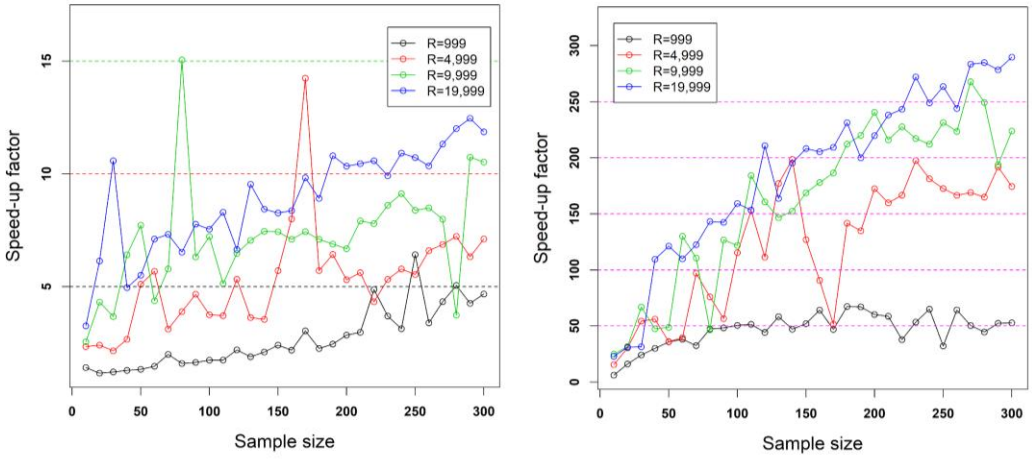


Figure 3. Welch's test; (a) speed-up factors of the asymptotic *p*-value versus the permutation *p*-value; this is an estimate of how much slower the permutation *p*-value is relative to the asymptotic *p*-value; the built-in command "t.test" was used; (b) speed up factor of the vectorized bootstrap (Neto, 2015a) versus our bootstrap implementation; this is an estimate of how much slower Neto's permutation method is, in comparison to our permutation method

The speed-up factor of the asymptotic *p*-value versus the permutation-based *p*-value was also estimated. As shown in Figure 2a, when $B = 999$ and $B = 4{,}999$ the permutation *p*-value was less than $\sqrt{B}$ times slower even for large sample sizes. The same was true for larger values of *B* but for sample sizes less than 150 or 200.

As noted in Figure 2b, the number of times Neto's bootstrap implementation (Neto, 2015a) is slower than our proposed permutation method. The execution speed differences were estimated between the R and C++ implementations. As shown in Figure 2c, for small sample sizes, C++ is two to three times faster, but as the sample size grows, the computational cost becomes the same.

As an addition to the simulation study described above, a comparison of the Welch's *t*-test on was conducted with a real gene-expression dataset containing 40 rows (observations) and 54,4675 columns (probesets). The dataset has the GSE15913 accession number in the GEO platform and can be downloaded from BioDataome (Lakiotaki et al., 2018). This dataset was chosen because it contains few samples. Bootstrap is mainly designed for small sample sized data for ensuring the correctness of tests, for example that the nominal selected significance level is close to the actual size of the test. We estimated the time required to perform bootstrap Welch's *t*-test for all columns. The average over 10 repetitions, using a

"for" loop to traverse the whole matrix, four our method was 6 seconds, whereas for Neto's (2015a) permutation method was 116 seconds (19 times slower).

Ordinary Bootstrap Versus Efficient Bootstrap for James Test

James multivariate version of the Welch's *t*-test was also examined. The rationale behind the application of our method for this test is similar to the ones previously described in this work. Presented in Figure 4 are the speed-up factors between the ordinary bootstrap procedure and our proposed efficient bootstrap procedure.

The R code for both cases appears in the Appendix. One can be seen, the number of "for" loops is more than the usual bootstrap implementation. The difference though lies in the number of calculations. In the ordinary bootstrap implementation, $2B$ covariance matrices are calculated, B for each sample. Using the efficient method, only $2\lceil\sqrt{B}\rceil$ covariance matrices are calculated. The computational cost does not reduce by $\lceil\sqrt{B}\rceil$, because more "for" loops were used and there were other heavy operations which cannot be avoided, such as inversion of covariance matrices. The number of inversions is $\lceil\sqrt{B}\rceil^2 > B$, and hence time is consumed there as well.

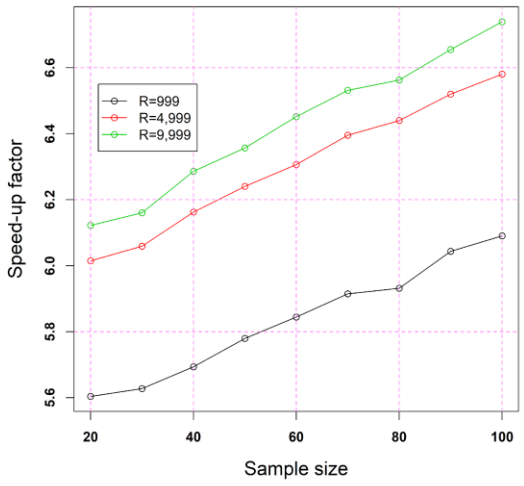


Table 4. James test; speed-up factors of the efficient versus the ordinary permutation bootstrap for the James test; this is an estimate of how much slower the ordinary bootstrap is, relative to our efficient bootstrap

Statistical Validation in Terms of Type I and Type II Errors

For both the correlation and Welch's *t*-test, data were generated under the H_0 hypotheses, of no correlation and equal means respectively. As shown in Figure 5, the quantiles of the true distribution were accurately estimated by the proposed method for the correlation and the Welch's *t*-test. Estimations for the James test at this point were not calculated; Tsagris et al. (2017) compared many hypothesis testing procedures with and without bootstrap calibration and have highlighted the importance of the bootstrap calibration in the multivariate case.

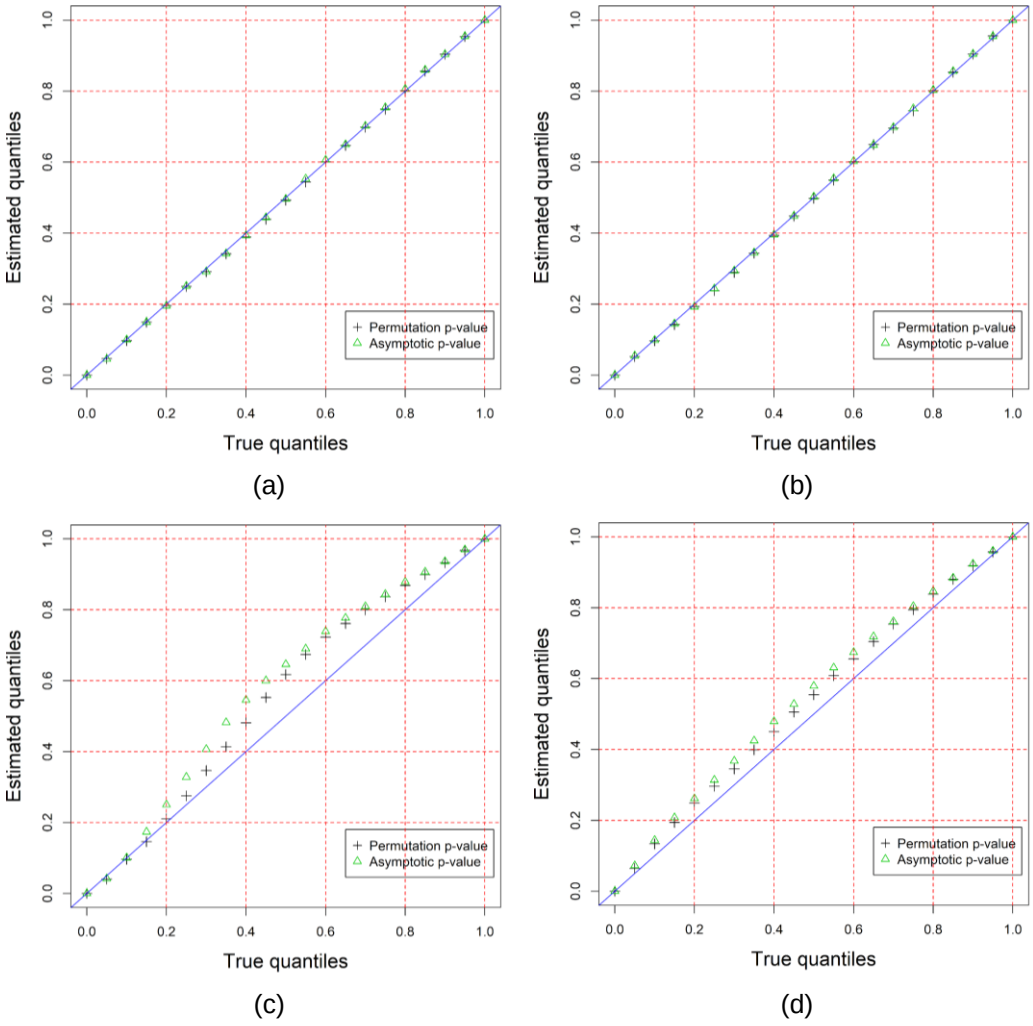


Figure 5. Estimated quantiles of the distribution of the test statistic under the null hypothesis; (a) and (b) refer to the correlation coefficient, while (c) and (d) refer to Welch's *t*-test

Conclusion

An efficient methodology was presented to obtain permutation-based *p*-values for the Pearson correlation coefficient, the two-sample *t*-test and extended it the two-sample multivariate *t*-test (James test). Another case where this method can be applied is for the distance correlation, where the *p*-value is estimated via permutations (Szekely et al., 2007). In (Bayesian) network construction, when partial correlation and partial distance correlations are employed, this method could help speed-up the process significantly. These methods are provided in the R package "Rfast" (Papadakis et al., 2018), as the functions 'permcor' and 'boot.ttest2' for the Pearson correlation and Welch *t*-test, respectively. The relative R code is also included in the Appendix.

References

Berry, K. J., Mielke, P. W., Jr., & Johnston, J. E. (2016). *Permutation statistical methods: An integrated approach*. Cham, Switzerland: Springer International. doi: 10.1007/978-3-319-28770-6

James, G. S. (1954). Tests of linear hypotheses in univariate and multivariate analysis when the ratios of the population variances are unknown. *Biometrika, 41*(1/2), 19-43. doi: 10.2307/2333003

Lakiotaki, K., Vorniotakis, N., Tsagris, M., Georgakopoulos, G., & Tsamardinos, I. (2018). Biodataome: A collection of uniformly preprocessed and automatically annotated datasets for data-driven biology. *Database, 2018*. doi: 10.1093/database/bay011

Legendre, P., & Legendre, L. F. (2012). *Numerical ecology* (3rd edition). Oxford, UK: Elsevier.

Mersmann, O. (2015). microbenchmark: Accurate timing functions [R package version 1.4-4]. https://cran.r-project.org/package=microbenchmark

Neapolitan, R. E. (2003). *Learning Bayesian networks*. Upper Saddle River, NJ: Pearson Prentice Hall.

Neto, E. C. (2015a). Speeding up non-parametric bootstrap computations for statistics based on sample moments in small/moderate sample size applications. *PloS One, 10*(6), e0131333. doi: 10.1371/journal.pone.0131333

Neto, E. C. [echaibub]. (2015b). *VectorizedNonParametricBootstrap* [GitHub repository]. https://github.com/echaibub/VectorizedNonParametricBootstrap

Papadakis, M., Tsagris, M., Dimitriadis, M., Fafalios, S., Tsamardinos, I., Fasiolo, M., Borboudakis, G., Burkardt, J., Zou, C., & Lakiotaki, K. (2018). Rfast: A collection of efficient and extremely fast R functions [R package version 1.8.8]. https://cran.r-project.org/package=Rfast

R Core Team. (2017). *R: A language and environment for statistical computing* [Software]. Vienna, Austria: R Foundation for Statistical Computing. https://www.r-project.org/

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin, 2*(6), 110-114. doi: 10.2307/3002019

Szekely, G. J., Rizzo, M. L., & Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *Annals of Statistics, 35*(6), 2769-2794. doi: 10.1214/009053607000000505

Tsagris, M., Preston, S., & Wood, A. T. A. (2017). Nonparametric hypothesis testing for equality of means on the simplex. *Journal of Statistical Computation and Simulation, 87*(2), 460-422. doi: 10.1080/00949655.2016.1216554

Tsamardinos, I., Aliferis, C. F., & Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In P. Domingos, C. Faloutsos, T. Senator, & L. Getoor (Eds.), *KDD-2003: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, Washington, D. C.* (pp. 673-678). New York: Association for Computing Machinery. doi: 10.1145/956750.956838

Tsamardinos, I., & Borboudakis, G. (2010). Permutation testing improves Bayesian network learning. In J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Eds.), *ECML PKDD 2010: Machine learning and knowledge discovery in databases, Barcelona, Spain* (Part III, pp. 322-337). Berlin: Springer. doi: 10.1007/978-3-642-15939-8_21

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. *Biometrika, 29*(3/4), 350-362. doi: 10.2307/2332010

Welch, B. L. (1947). The generalization of ‘Student's’ problem when several different population variances are involved. *Biometrika, 34*(1/2), 28-35. doi: 10.2307/2332510

Appendix

```
library(Rfast)  ## necessary package to load
### Permutations for correlation coefficient
permcor <- function(x, y, R = 999) {
  n <- length(x)    ;    m1 <- sum(x)    ;    m12 <- sum(x^2)
  m2 <- sum(y)    ;    m22 <- sum(y^2)    ;    up <-  m1 * m2 / n
  down <- sqrt( (m12 - m1^2 / n) * (m22 - m2^2 / n) )
  r <- ( sum(x * y) - up) / down
  test <- log( (1 + r) / (1 - r) )  ## the test statistic
  B <- round( sqrt(R) )    ;    xp <- matrix(0, n, B)    ;    yp
<- matrix(0, n, B)
  for (i in 1:B) {
     xp[, i] <- sample(x, n)      ;      yp[, i] <- sample(y, n)
  }
  sxy <- crossprod(xp, yp)     ;      rb <- (sxy - up) / down
  tb <- log( (1 + rb) / (1 - rb) )
  pvalue <- ( sum( abs(tb) > abs(test) ) + 1 ) / (B^2 + 1)  ## p-
value
  res <- c( r, pvalue )
  names(res) <- c('correlation', 'p-value')
  res
}

### Ordinary bootstrap for James test
james <- function(y1, y2, R = 999) {
  p <- dim(y1)[2]     ;   n1 <- dim(y1)[1]     ;    n2 <-
dim(y2)[1]
  ybar1 <- Rfast::colmeans(y1)    ;    ybar2 <-
Rfast::colmeans(y2)
  dbar <- ybar2 - ybar1    ;    A1 <- Rfast::cova(y1)/n1    ;
A2 <- Rfast::cova(y2)/n2
  test <- as.vector( dbar %*% solve(A1 + A2, dbar ) )
  a1inv <- Rfast::spdinv(A1)       ;        a2inv <-
Rfast::spdinv(A2)
  mc <- solve( a1inv + a2inv, a1inv %*% ybar1 + a2inv %*% ybar2 )
  mc1 <-  - ybar1 + mc           ;        mc2 <-  - ybar2 + mc
```

```
  x1 <- Rfast::eachrow(y1, mc1, oper = "+")
  x2 <- Rfast::eachrow(y2, mc2, oper = "+" )
  tb <- numeric(R)
  for (i in 1:R) {
    b1 <- sample(1:n1, n1, replace = TRUE)    ;    b2 <-
sample(1:n2, n2, replace = TRUE)
    xb1 <- x1[b1, ]     ;      xb2 <- x2[b2, ]
    db <- Rfast::colmeans(xb1) - Rfast::colmeans(xb2)
    Vb <- Rfast::cova(xb1) / n1 + Rfast::cova(xb2) / n2
    tb[i] <- sum( db %*% solve(Vb, db ) )
  }
  ( sum(tb > test) + 1 ) / (R + 1)
}

## Efficient bootstrap for James test
boot.james <- function(y1, y2, R = 999) {
  p <- dim(y1)[2]    ;    n1 <- dim(y1)[1]    ;    n2 <-
dim(y2)[1]
  ybar2 <- Rfast::colmeans(y2)      ;      ybar1 <-
Rfast::colmeans(y1)
  dbar <- ybar2 - ybar1    ;     A1 <- Rfast::cova(y1)/n1      ;
A2 <- Rfast::cova(y2)/n2
  test <- as.vector( dbar %*% solve(A1 + A2, dbar ) )
  a1inv <- Rfast::spdinv(A1)      ;      a2inv <-
Rfast::spdinv(A2)
  mc <- solve( a1inv + a2inv, a1inv %*% ybar1 + a2inv %*% ybar2 )
  mc1 <-  - ybar1 + mc       ;       mc2 <-  - ybar2 + mc
  x1 <- Rfast::eachrow(y1, mc1, oper = "+")     ;       x2 <-
Rfast::eachrow(y2, mc2, oper = "+" )
  B <- round( sqrt(R) )      ;       tb <- matrix(0, B, B)      ;
bm1 <- bm2 <- matrix(nrow = B, ncol = p)
  vb1 <- vector("list", B)     ;      vb2 <- vector("list", B)
  tb <- matrix(0, B, B)    ;     sqn1 <- sqrt(n1)     ;     sqn2
<- sqrt(n2)
  for (i in 1:B) {
    b1 <- sample(1:n1, n1, replace = TRUE)   ;     b2 <-
sample(1:n2, n2, replace = TRUE)
    yb1 <- x1[b1, ]    ;   yb2 <- x2[b2, ]
```

```
    bm1[i, ] <- Rfast::colmeans(yb1)      ;         bm2[i, ] <-
Rfast::colmeans(yb2)
    vb1[[ i ]] <- (crossprod(yb1) - tcrossprod( sqn1 *
bm1[i, ]) ) / n1
    vb2[[ i ]] <- (crossprod(yb2) - tcrossprod( sqn2 *
bm2[i, ]) ) / n2
  }
  for (i in 1:B) {
    for (j in 1:B) {
      vb <- vb1[[ i ]] + vb2[[ j ]]       ;      db <- bm1[i, ] -
bm2[j, ]
      tb[i, j] <- db %*% solve(vb, db)
    }
  }
  ( sum(tb > test) + 1 ) / (B^2 + 1)
}
```