Journal of Statistical Software

March 2021, Volume 97, Issue 6. *doi: 10.18637/jss.v097.i06*

Ball: An R Package for Detecting Distribution Difference and Association in Metric Spaces

Jin Zhu
Sun Yat-Sen University

Wenliang Pan
Sun Yat-Sen University

Wei Zheng
University of Tennessee

Xueqin Wang
University of Science and Technology of China

Abstract

The rapid development of modern technology has created many complex datasets in non-linear spaces, while most of the statistical hypothesis tests are only available in Euclidean or Hilbert spaces. To properly analyze the data with more complicated structures, efforts have been made to solve the fundamental test problems in more general spaces (Lyons 2013; Pan, Tian, Wang, and Zhang 2018; Pan, Wang, Zhang, Zhu, and Zhu 2020). In this paper, we introduce a publicly available R package **Ball** for the comparison of multiple distributions and the test of mutual independence in metric spaces, which extends the test procedures for the equality of two distributions (Pan *et al.* 2018) and the independence of two random objects (Pan *et al.* 2020). The **Ball** package is computationally efficient since several novel algorithms as well as engineering techniques are employed in speeding up the ball test procedures. Two real data analyses and diverse numerical studies have been performed, and the results certify that the **Ball** package can detect various distribution differences and complicated dependencies in complex datasets, e.g., directional data and symmetric positive definite matrix data.

Keywords: K-sample test problem, test of mutual independence problem, ball divergence, ball covariance, metric space.

1. Introduction

With the advanced modern instruments such as the Doppler shift acoustic radar, functional magnetic resonance imaging (fMRI) apparatus, and Heidelberg retina tomograph device, a large number of complex datasets are being collected for contemporary scientific research. For example, to investigate whether the wind directions of two places are distinct, meteorologists measure the wind directions by colatitude and longitude coordinates on the earth. Another typical example arises in biology. By using fMRI data, biologists are able to study the association between the brain connectivity and the age. Although these complex datasets

are potentially useful for the progress of scientific research, their various and complicated structures challenge testing the equality of distributions and testing the mutual independence of random objects, two fundamental problems of statistical inference. The two problems are generally named as the K-sample test problem and the test of mutual independence problem, which we reconsider in general metric spaces here.

In the literature, a large number of methods have been developed to address these two problems. Correspondingly, there are many functions currently available in R (R Core Team 2021), including `oneway.test`, `kruskal.test` and `cor.test` from package **stats**, `ad.test` from package **kSamples** (Scholz and Zhu 2019), `tauStarTest` from package **TauStar** (Weihs 2019), `HellCor` from package **HellCor** (Geenens and Lafaye de Micheaux 2021), `hotelling.test` from package **Hotelling** (Curran 2018), `coeffRV` from package **FactoMineR** (Lê, Josse, and Husson 2008), `kmmd` from package **kernlab** (Karatzoglou, Smola, Hornik, and Zeileis 2004), `hsic.test` from package **kpcalg** (Verbyla, Desgranges, and Wernisch 2017), `dhsic.test` from package **dHSIC** (Pfister and Peters 2019), `eqdist.test` and `dcov.test` from package **energy** (Rizzo and Székely 2019), `mdm_test` from package **EDMeasure** (Jin, Yao, Matteson, and Shao 2018), `multivariance.test` from package **multivariance** (Böttcher 2020), `independence_test` from package **coin** (Hothorn, Hornik, van de Wiel, and Zeileis 2008), `MINTperm` from package **IndepTest** (Berrett, Grose, and Samworth 2018), `hoeffD`, `hoeffR`, `pTStar` and `jTStar` from package **SymRC** (`https://github.com/Lucaweihs/SymRC`), `hhg.test.k.sample` and `hhg.test` from package **HHG** (Brill, Heller, and Heller 2018), and so on. Among them, the functions in **stats** and **kSamples** implement the classical parametric and non-parametric hypothesis tests for univariate distributions and univariate random variables. The **TauStar** and **HellCor** packages implement two novel univariate dependence measures, Bergsma-Dassios sign covariance (Bergsma and Dassios 2014) and Hellinger correlation (Geenens and Lafaye de Micheaux 2021), which possess admirable theoretical advantages. In short, they are designed for univariate data, and hence are restricted. The **Hotelling** and **FactoMineR** packages provide the multivariate extension of the Student's t test and the Pearson correlation test, but the normality assumption for multivariate data is usually difficult to validate. The functions in **kernlab**, **kpcalg**, **dHSIC**, **energy**, **EDMeasure**, and **multivariance** are capable of distinguishing distributions and examining (mutual) independence assumptions for univariate/multivariate continuous/discrete data. Unfortunately, since these packages rely on energy distance (Székely and Rizzo 2004) and distance covariance (Székely, Rizzo, and Bakirov 2007) or maximum mean discrepancy (Gretton, Borgwardt, Rasch, Schölkopf, and Smola 2012) and Hilbert-Schmidt independence criterion (Gretton, Bousquet, Smola, and Schölkopf 2005), they will totally lose power when the metric is not of strong negative type (Lyons 2013) or the kernel is not of positive definiteness (Sejdinovic, Sriperumbudur, Gretton, and Fukumizu 2013). As for the **coin** package, it offers user-friendly and highly flexible interfaces for performing the K-sample and independence permutation tests on Euclidean geometry datasets under the framework proposed by Strasser and Weber (1999). The functions in **IndepTest** use mutual information, a well-known dependence measure, to perform the independence test between two Euclidean vectors (Berrett and Samworth 2019). The **SymRC** package provides a competitive dependence measure, symmetric rank covariances, to test the multivariate independence in Euclidean space (Drton, Weihs, and Meinshausen 2018). The functions in **HHG**, based on the work of Heller, Heller, and Gorfine (2013), are known to be useful in detecting distinctions among multivariate distributions and associations between multivariate random variables in Euclidean space.

Recently, two novel concepts, ball divergence (Pan *et al.* 2018) and ball covariance (Pan *et al.* 2020), are proposed to measure the discrepancy between two distributions and the dependence between two random objects in metric spaces, respectively. Ball divergence (BD) enjoys a remarkable property, homogeneity-zero equivalence, and ball covariance (BCOV) holds another brilliant property, independence-zero equivalence. The BD and BCOV statistics, as the empirical versions of BD and BCOV, can tackle the two-sample test and test of independence problems, which are special cases of the K-sample test and test of mutual independence problems. The BD and BCOV statistics are both robust rank statistics, and the test procedures based on them are consistent against any general alternative hypothesis without distribution or moment assumptions (Pan *et al.* 2018, 2020). Besides, the BD statistic is proved to cope well with imbalanced data, and the BCOV statistic can be standardized to the ball correlation statistic to extract important features from ultra-high dimensional data (Pan, Wang, Xiao, and Zhu 2019).

In this paper, we introduce a user-friendly R package **Ball** (Wang *et al.* 2021). The **Ball** package contributes to the open-source statistical software community in the following aspects: (i) It provides the BD two-sample test and the BCOV independence test to R users; (ii) It implements several dedicated design algorithms to accelerate the BD two-sample test and the BCOV independence test; (iii) It provides three powerful K-sample BD test statistics and an efficient K-sample permutation test procedure to distinguish distributions in metric spaces; (iv) It extends the BCOV test statistic to detect the mutual dependence among complex random objects in metric spaces; (v) It supports several generic sure independence screening procedures which are capable of extracting important features associated with complex objects in metric spaces. At present, the **Ball** package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=Ball`.

The remaining sections are organized as follows. In Section 2, we propose our ball test statistics and ball test procedures to tackle the K-sample test and test of mutual independence problems in metric spaces. In Section 3, we introduce several novel and efficient algorithms for the ball test statistics and ball test procedures. Section 4 gives a detailed account for the main functions in the **Ball** package and provides two real data examples to demonstrate their usages for complex dataset. Section 5 discusses the numerical performance of the ball test statistics in the K-sample test and the test of mutual independence problems. Finally, the paper concludes with a short summary in Section 6.

2. Ball test statistics

In this section, we define and illustrate the ball divergence (BD) and ball covariance (BCOV) statistics in Section 2.1 and 2.2, respectively. We describe the details of the ball test procedures in Section 2.3.

2.1. K-sample test and ball divergence statistic

In a metric space (V, d), given K independent observed samples $\mathcal{X}_k = \{X_{ki} \mid i = 1, \ldots, n_k\}$, $k = 1, \ldots, K$, assuming that in the k-th sample, $X_{k1}, \ldots, X_{kn_k}$ are i.i.d. and associated with the Borel probability measure μ_k. The null hypothesis of the K-sample test problem is formulated as

$$H_0 : \mu_1 = \cdots = \mu_K.$$

We first revisit the BD statistic designed for the two-sample test problem (Pan *et al.* 2018), a special case of the K-sample test problem, where $K = 2$ and $N = n_1 + n_2$. Let $I(\cdot)$ be an indicator function. For any $x, y, z \in V$, denote $B(x, y)$ as a closed ball with center x and radius $d(x, y)$, and $\delta(x, y, z) = I(z \in B(x, y))$. Therefore, $\delta(x, y, z)$ takes the value of 1 when z is inside the closed ball $B(x, y)$, or 0 otherwise. Let

$$P_{ij}^{\mu_1\mu_1} = \frac{1}{n_1}\sum_{t=1}^{n_1}\delta(X_{1i}, X_{1j}, X_{1t}),\ P_{ij}^{\mu_1\mu_2} = \frac{1}{n_2}\sum_{t=1}^{n_2}\delta(X_{1i}, X_{1j}, X_{2t}),$$
$$P_{kl}^{\mu_2\mu_1} = \frac{1}{n_1}\sum_{t=1}^{n_1}\delta(X_{2k}, X_{2l}, X_{1t}),\ P_{kl}^{\mu_2\mu_2} = \frac{1}{n_2}\sum_{t=1}^{n_2}\delta(X_{2k}, X_{2l}, X_{2t}),$$

then the two-sample BD statistic is defined as

$$\mathrm{BD}_N(\mu_1, \mu_2) = \frac{1}{n_1^2}\sum_{i,j=1}^{n_1}(P_{ij}^{\mu_1\mu_1} - P_{ij}^{\mu_1\mu_2})^2 + \frac{1}{n_2^2}\sum_{k,l=1}^{n_2}(P_{kl}^{\mu_2\mu_1} - P_{kl}^{\mu_2\mu_2})^2.$$

Intuitively, if $\mathcal{X}_1$ and $\mathcal{X}_2$ come from the same distribution, the proportions of the elements of $\mathcal{X}_1$ and $\mathcal{X}_2$ in the closed balls $B(X_{1i}, X_{1j})$ and $B(X_{2k}, X_{2l})$ are almost the same, in other words, $P_{ij}^{\mu_1\mu_1} \approx P_{ij}^{\mu_1\mu_2}, P_{kl}^{\mu_2\mu_1} \approx P_{kl}^{\mu_2\mu_2}$. Consequently, BD_N approaches to zero in this scenario. Otherwise, if $\mathcal{X}_1$ and $\mathcal{X}_2$ come from two distinct distributions, then BD_N is, relatively, far away from zero.

Generally, for $K > 2$ and $N = \sum_{k=1}^{K} n_k$, the definition of the K-sample BD statistic could be to directly sum up all of the two-sample BD statistics

$$\mathrm{BD}_N^{\mathcal{S}_K}(\mu_1, \ldots, \mu_K) = \sum_{1\le s<t\le K}\mathrm{BD}_{n_s+n_t}(\mu_s, \mu_t),$$

or to find one group with the largest difference to other groups

$$\mathrm{BD}_N^{\mathcal{MS}_K}(\mu_1, \ldots, \mu_K) = \max_t \sum_{s=1, s\neq t}^{K}\mathrm{BD}_{n_s+n_t}(\mu_s, \mu_t),$$

or to aggregate the $K-1$ most significant two-sample BD statistics

$$\mathrm{BD}_N^{\mathcal{M}_K}(\mu_1, \ldots, \mu_K) = \sum_{k=1}^{K-1}\mathrm{BD}_{(k)},$$

where $\mathrm{BD}_{(1)}, \ldots, \mathrm{BD}_{(K-1)}$ are the largest $K-1$ two-sample BD statistics among the set $\{\mathrm{BD}_{n_s+n_t}(\mu_s, \mu_t) \mid 1 \le s < t \le K\}$. When $K = 2$, $\mathrm{BD}_N^{\mathcal{S}_K}, \mathrm{BD}_N^{\mathcal{M}_K}$, and $\mathrm{BD}_N^{\mathcal{MS}_K}$ degenerate into BD_N.

2.2. Test of mutual independence and ball covariance statistic

Assume that $(V_1, d_1), \ldots, (V_K, d_K)$ are metric spaces, D is the Euclidean norm on R^K, and (V, d) is the product metric space of $(V_1, d_1), \ldots, (V_K, d_K)$, where the product metric is defined by $d((x_1, \ldots, x_K), (y_1, \ldots, y_K)) = D(d_1(x_1, y_1), \ldots, d_K(x_K, y_K))$. Suppose $\mathcal{X} = \{(X_{i1}, \ldots, X_{iK}) \mid i = 1, \ldots, N\}$ is a sample with N i.i.d. observations in the product metric space V, whose associated joint Borel probability measure and marginal Borel probability

measures are μ and $\mu_1, \ldots, \mu_K$. The null hypothesis of the test of mutual independence problem is formulated as

$$H_0 : \mu = \mu_1 \otimes \cdots \otimes \mu_K.$$

For $x, y, z \in V_i$, denote $B_i(x, y)$ as a closed ball with center x and radius $d_i(x, y)$, and $\delta_i(x, y, z) = I(z \in B_i(x, y))$. Thus, $\delta_i(x, y, z)$ is an indicator taking the value of 1 if z is within the closed ball $B_i(x, y)$, or 0 otherwise. Let

$$P_{ij}^{\mu} = \frac{1}{N} \sum_{t=1}^{N} \prod_{k=1}^{K} \delta_k(X_{ik}, X_{jk}, X_{tk}),$$

and

$$P_{ij}^{\mu_k} = \frac{1}{N} \sum_{t=1}^{N} \delta_k(X_{ik}, X_{jk}, X_{tk}), \quad k = 1, \ldots, K,$$

then our BCOV statistic can be presented as

$$\mathrm{BCov}_N^K(\mu_1, \ldots, \mu_K) = \frac{1}{N^2} \sum_{i,j=1}^{N} (P_{ij}^{\mu} - \prod_{k=1}^{K} P_{ij}^{\mu_k})^2.$$

We provide a heuristic explanation for BCov_N^K. If $\mu = \mu_1 \otimes \cdots \otimes \mu_K$, the proportion of the elements of $\mathcal{X}$ in $\otimes_{k=1}^K B_k(X_{ik}, X_{jk})$ should be close to the product of the proportions of $X_{1k}, \ldots, X_{Nk}$ in $B_k(X_{ik}, X_{jk})$, i.e., $P_{ij}^{\mu} \approx \prod_{k=1}^K P_{ij}^{\mu_k}$. Since BCov_N^K is the average of $\{(P_{ij}^{\mu} - \prod_{k=1}^K P_{ij}^{\mu_k})^2 \mid i, j = 1, \ldots, N\}$, a significantly large BCov_N^K implies that the null hypothesis $\mu = \mu_1 \otimes \cdots \otimes \mu_K$ is implausible.

The BCOV statistic can be extended with positive weights $\hat{\omega}_k(X_{ik}, X_{jk}), k = 1, \ldots, K$,

$$\mathrm{BCov}_{\omega,N}^K(\mu_1, \ldots, \mu_K) = \frac{1}{N^2} \sum_{i,j=1}^{N} (P_{ij}^{\mu} - \prod_{k=1}^{K} P_{ij}^{\mu_k})^2 \prod_{k=1}^{K} \hat{\omega}_k(X_{ik}, X_{jk}).$$

As a more general dependence measure framework based on the BCOV statistic, such a weighted extension not only allows flexible test statistics but also connects the BCOV statistic with HHG. Several choices for the weights are feasible. For example, we could choose the probability weight $\hat{\omega}_k(X_{ik}, X_{jk}) = [P_{ij}^{\mu_k}]^{-1}$ and the Chi-square weight $\hat{\omega}_k(X_{ik}, X_{jk}) = [P_{ij}^{\mu_k}(1 - P_{ij}^{\mu_k})]^{-1}$, and denote their corresponding statistics as $\mathrm{BCov}_{\Delta,N}^K$ and $\mathrm{BCov}_{\chi^2,N}^K$, respectively. $\mathrm{BCov}_{\Delta,N}^K$ focuses on smaller balls, while $\mathrm{BCov}_{\chi^2,N}^K$ standardizes $(P_{ij}^{\mu} - \prod_{k=1}^K P_{ij}^{\mu_k})^2$ by the variance of $\delta_k(X_{ik}, X_{jk}, X_{tk})$ (i, j are fixed). Furthermore, $\mathrm{BCov}_{\chi^2,N}^K(K = 2)$ is asymptotically equivalent to HHG (Pan *et al.* 2020). From the definitions of BCov_N^K, $\mathrm{BCov}_{\Delta,N}^K$, and $\mathrm{BCov}_{\chi^2,N}^K$, we may expect: (i) BCov_N^K is good at detecting linear relationships, especially when noises are influential, because treating each ball indiscriminately is a reasonable strategy which keeps weights away from potential instabilities; (ii) $\mathrm{BCov}_{\Delta,N}^K$ has more power in detecting strong nonlinear relationships since paying more attention to smaller balls makes $\mathrm{BCov}_{\Delta,N}^K$ tending to detect the locally linear relationship; (iii) $\mathrm{BCov}_{\chi^2,N}^K$ (or HHG) is an intermediate between BCov_N^K and $\mathrm{BCov}_{\Delta,N}^K$.

The BCOV statistic can be normalized. Let

$$\mathrm{BCov}^K_{\omega,N}(\mu_k) = \frac{1}{N^2}\sum_{i,j=1}^{N}[P^{\mu_k}_{ij} - (P^{\mu_k}_{ij})^K]^2(\hat{\omega}_k(X_{ik}, X_{jk}))^K, \quad k = 1, \ldots, K,$$

then the normalized version of the BCOV statistic is defined as the square root of

$$\mathrm{BCor}^K_{\omega,N}(\mu_1, \ldots, \mu_K) = \mathrm{BCov}^K_{\omega,N}(\mu_1, \ldots, \mu_K)/\sqrt{\prod_{k=1}^{K}\mathrm{BCov}^K_{\omega,N}(\mu_k)},$$

if $\prod_{k=1}^{K}\mathrm{BCov}^K_{\omega,N}(\mu_k) > 0$, or 0 otherwise. $\mathrm{BCor}^K_{\omega,N}(K = 2)$ is the ball correlation statistic (Pan *et al.* 2020) which ranges from 0 to 1.

2.3. Ball permutation test procedure

After computing an observed ball test statistic, say B, the permutation methodology described in Efron and Tibshirani (1994) and Davison and Hinkley (1997) is employed to derive the p value of the ball test procedures in a distribution-free manner. We specify the permutation procedures for the K-sample test and test of mutual independence problems below.

For the K-sample test problem, let $\mathbf{k}_{n_k}$ be an n_k-dimensional vector whose entries are all k, and let $L = (\mathbf{1}^\top_{n_1}, \mathbf{2}^\top_{n_2}, \ldots, \mathbf{K}^\top_{n_K})^\top$ be the group label vector of the pooled sample $\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_K$. Denote L^* as a permutation of L and $\{Y_i\}_{i=1}^N$ as $\mathcal{X}$, the shuffled pooled sample associated with L^* is $\mathcal{X}^* = \mathcal{X}^*_1 \cup \cdots \cup \mathcal{X}^*_K$, where $\mathcal{X}^*_k = \{Y_i \in \mathcal{X} \mid L^*_i = k, i = 1, \ldots, N\}$. With the shuffled pooled sample $\mathcal{X}^*$, we can compute the K-sample BD statistic. The permutation is replicated M times to derive the K-sample BD statistics under the null hypothesis: $B'_1, \ldots, B'_M$. Finally, the p value is computed according to:

$$p \text{ value} = \frac{1 + \sum_{m=1}^{M} I(B'_m \geq B)}{1 + M}. \tag{1}$$

With respect to the test of mutual independence problem, each permutation is performed following this manner: for each $k \in \{1, \ldots, K\}$, $X_{1k}, \ldots, X_{Nk}$ are randomly shuffled while $X_{1k'}, \ldots, X_{Nk'}$ $(k' \neq k)$ are fixed. The permutation is replicated M times and the p value is estimated by Equation 1.

3. Algorithm

The computational complexities of the ball test statistics are $O(N^3)$ if we compute them according to their definitions; however, in many cases, their computational complexities can reduce to a more moderate level (see Table 1) by efficient algorithms utilizing the rank nature of the ball test statistics. The rank nature motivates the acceleration in three aspects. First, the computation procedure of the ball test statistics can avoid repeatedly evaluating whether a point is in a closed ball because $n_t P^{\mu_s\mu_t}_{ij}(s, t \in \{1, \ldots, K\})$ and $NP^{\mu_k}_{ij}(k = 1, \ldots, K)$ are related to some ranks. Second, for the K-sample permutation test, performing ranking procedures could be more efficient by preserving information for the first time when we compute the K-sample BD statistics. Third, for univariate datasets, we can optimize the ranking procedure for computing $n_t P^{\mu_s\mu_t}_{ij}(s, t \in \{1, \ldots, K\})$ and $NP^{\mu_k}_{ij}(k = 1, \ldots, K)$ without preserving

Statistics	Univariate	Other
$\mathrm{BD}_N^{\mathcal{S}_K}/\mathrm{BD}_N^{\mathcal{MS}_K}/\mathrm{BD}_N^{\mathcal{M}_K}$	$O(N^2)$	$O(N^2)$
$\mathrm{BCov}_{\omega,N}^K(K=2)$	$O(N^2)$	$O(N^2 \log N)$

Table 1: The optimized computational complexities of several ball test statistics.

auxiliary information. The three aspects will be illustrated in Section 3.1, 3.2, and 3.3, respectively. For simplicity, we assume there is no ties among datasets. For our **Ball** package, it can properly handle tied data and compute the exact ball test statistics.

Unfortunately, in the case of measuring mutual dependence among at least three random objects, it is not easy to optimize the computation procedure of the BCOV statistic. Nevertheless, with engineering optimizations such as multi-threading, the time consumption of computing the BCOV statistic can be cut down.

3.1. Rank-based algorithm

We first recap the $O(N^2 \log N)$ algorithm for BD_N proposed by Pan *et al.* (2018). Assume the pairwise distance matrix of $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ is:

$$D^{\mathcal{X}\mathcal{X}} = \begin{pmatrix} D^{\mathcal{X}_1\mathcal{X}_1} & D^{\mathcal{X}_1\mathcal{X}_2} \\ D^{\mathcal{X}_2\mathcal{X}_1} & D^{\mathcal{X}_2\mathcal{X}_2} \end{pmatrix}_{N\times N}.$$

Notice that, $n_1 P_{ij}^{\mu_1\mu_1}$ is the rank of $d(X_{1i}, X_{1j})$ among the i-th row of $D^{\mathcal{X}_1\mathcal{X}_1}$, and $n_1 P_{ij}^{\mu_1\mu_1} + n_2 P_{ij}^{\mu_1\mu_2}$ is the rank of $d(X_{1i}, X_{1j})$ among the i-th row of $D^{\mathcal{X}\mathcal{X}}$. Consequently, after spending $O(N^2 \log N)$ time on ranking $D^{\mathcal{X}_1\mathcal{X}_1}$ and $D^{\mathcal{X}\mathcal{X}}$ row by row to obtain their corresponding rank matrices $R^{\mathcal{X}_1\mathcal{X}_1}$ and $R^{\mathcal{X}\mathcal{X}}$, we only need $O(1)$ time to compute $n_1 P_{ij}^{\mu_1\mu_1}$ and $n_1 P_{ij}^{\mu_1\mu_1} + n_2 P_{ij}^{\mu_1\mu_2}$ by directly extracting the (i,j)-elements of $R^{\mathcal{X}_1\mathcal{X}_1}$ and $R^{\mathcal{X}\mathcal{X}}$. Therefore, computing $\{(P_{ij}^{\mu_1\mu_1}, P_{ij}^{\mu_1\mu_2}) \mid i,j = 1,\ldots,n_1\}$ is of $O(N^2 \log N)$ time complexity, and similarly for computing $\{(P_{kl}^{\mu_2\mu_1}, P_{kl}^{\mu_2\mu_2}) \mid k,l = 1,\ldots,n_2)\}$. In summary, for BD_N and the K-sample BD statistics, their time complexities are $O(N^2 \log N)$.

With respect to $\mathrm{BCov}_{\omega,N}^K(K=2)$, the aforementioned algorithm can be directly applied to calculate $P_{ij}^{\mu_1}$ and $P_{ij}^{\mu_2}$ within $O(N^2 \log N)$ time. To compute $\{NP_{ij}^{\mu} \mid i,j = 1,\ldots,N\}$ within $O(N^2 \log N)$ time, we first rearrange $\{(d_1(X_{i1}, X_{j1}), d_2(X_{i2}, X_{j2})) \mid j = 1,\ldots,N\}$ to $\{(d_1(X_{i1}, X_{i_j 1}), d_2(X_{i2}, X_{i_j 2})) \mid j = 1,\ldots,N)\}$, where i_j is the location of the j-th smallest value among $\{d_1(X_{i1}, X_{j1}) \mid j = 1,\ldots,N\}$. Further, for $j = 1,\ldots,N$, we have

$$\begin{aligned} NP_{ii_j}^{\mu} &= \sum_{t=1}^{N} I(d_1(X_{i1}, X_{i_t 1}) \le d_1(X_{i1}, X_{i_j 1}))I(d_2(X_{i2}, X_{i_t 2}) \le d_2(X_{i2}, X_{i_j 2})) \\ &= \sum_{t=1}^{j} I(d_2(X_{i2}, X_{i_t 2}) \le d_2(X_{i2}, X_{i_j 2})) \\ &= \sum_{t=1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \le d_2(X_{i2}, X_{i_j 2})) - \sum_{t=j+1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \le d_2(X_{i2}, X_{i_j 2})) \\ &= NP_{ii_j}^{\mu_2} - \sum_{t=j+1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \le d_2(X_{i2}, X_{i_j 2})). \end{aligned} \tag{2}$$

Owing to Equation 2, the computation of $\{NP_{ii_j}^{\mu} \mid j = 1, \ldots, N\}$ could be turned into computing $\{\sum_{t=j+1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \leq d_2(X_{i2}, X_{i_j 2})) \mid j = 1, \ldots, N\}$. Notice that, given the array $A = \{d_2(X_{i2}, X_{i_j 2}) \mid j = 1, \ldots, N\}$, $\sum_{t=j+1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \leq d_2(X_{i2}, X_{i_j 2}))$ is the number of the elements of A which are not only behind $d_2(X_{i2}, X_{i_j 2})$ but also no larger than $d_2(X_{i2}, X_{i_j 2})$. Therefore, computing $\{\sum_{t=j+1}^{N} I(d_2(X_{i2}, X_{i_t 2}) \leq d_2(X_{i2}, X_{i_j 2})) \mid j = 1, \ldots, N\}$ is the typical "count of smaller numbers after self" problem (https://leetcode.com/problems/count-of-smaller-numbers-after-self/), which can be solved within $O(N \log N)$ time via well designed algorithms such as binary indexed tree, binary search tree, and merge sort. Our implementation (see Appendix A) utilizes merge sort to resolve the problem due to its efficiency. Thus, the time complexity of $\{NP_{ij}^{\mu} \mid i, j = 1, \ldots, N\}$ reduces to $O(N^2 \log N)$, and finally, the computation of $\mathrm{BCov}_{\omega,N}^{K}(K = 2)$ costs $O(N^2 \log N)$ time.

3.2. Optimized K-sample permutation test procedure

In this section, an efficient procedure of $O(N^2)$ time complexity is introduced to compute the K-sample BD statistics on a shuffled dataset $\mathcal{X}^*$. According to the definition of the K-sample BD statistics, to reduce their time complexity to $O(N^2)$, we should reduce the time complexity of any $\mathrm{BD}_{n_s+n_t}$ to $O(N^2)$. From Section 3.1, to reduce the time complexity of $\mathrm{BD}_{n_s+n_t}$ to $O(N^2)$, we need to compute the row-wise ranks of pairwise distance matrices $D^{\mathcal{X}_s^* \mathcal{X}_s^*}$, $D^{\mathcal{X}_t^* \mathcal{X}_t^*}$, and $D^{\mathcal{X}_{s,t}^* \mathcal{X}_{s,t}^*}(\mathcal{X}_{s,t}^* = \mathcal{X}_s^* \cup \mathcal{X}_t^*)$ within $O(N^2)$ time. We propose Algorithm 1 to achieve this goal. The core of Algorithm 1 is utilizing an $N \times N$ order information matrix $I^{\mathcal{X}\mathcal{X}}$ as well as an N-dimensional vector G. For the order information matrix $I^{\mathcal{X}\mathcal{X}}$, $I_{ij}^{\mathcal{X}\mathcal{X}} = q$ means that the j-th smallest element of the i-th row of pairwise distance matrix $D^{\mathcal{X}\mathcal{X}}$ is $D_{iq}^{\mathcal{X}\mathcal{X}}$. $I^{\mathcal{X}\mathcal{X}}$ can be obtained for the first time when we rank each row of the pairwise distance matrix $D^{\mathcal{X}\mathcal{X}}$. Concerning the vector G, it is related to the shuffled group label vector L^* and the cumulative sample size vector $C = (0, n_1, \ldots, \sum_{k=1}^{K-1} n_k)$. Specifically, when $L_i^* = k$, $G_i = C_k + \sum_{j=1}^{i-1} I(L_j^* = k) + 1$ if $i > 1$, or $G_i = C_k + 1$ if $i = 1$. By scanning $I^{\mathcal{X}\mathcal{X}}$ in a row-wise manner, Algorithm 1 can assign a proper rank value to the element of the row-wise rank matrices of $D^{\mathcal{X}_s^* \mathcal{X}_s^*}$, $D^{\mathcal{X}_t^* \mathcal{X}_t^*}$, and $D^{\mathcal{X}_{s,t}^* \mathcal{X}_{s,t}^*}$ with the help of the vector G.

3.3. Fast algorithm for univariate data

We first consider BD_N. For convenience, assume $X_{11}, \ldots, X_{1n_1}$ have been sorted in ascending order. For univariate data, we have

$$\begin{aligned} n_1 P_{ij}^{\mu_1\mu_1} &= \sum_{t=1}^{n_1} I(|X_{1t} - X_{1i}| \leq |X_{1i} - X_{1j}|) \\ &= \sum_{t=1}^{n_1} I(X_{1i} - |X_{1i} - X_{1j}| \leq X_{1t} \leq X_{1i} + |X_{1i} - X_{1j}|). \end{aligned} \tag{3}$$

Let $X_{1l_{ij}}$ be the smallest value satisfying $X_{1l_{ij}} \geq X_{1i} - |X_{1i} - X_{1j}|$ and $X_{1r_{ij}}$ be the largest element satisfying $X_{1r_{ij}} \leq X_{1i} + |X_{1i} - X_{1j}|$. Thanks to Equation 3, it is easy to verify that $n_1 P_{ij}^{\mu_1\mu_1} = r_{ij} - l_{ij} + 1$, and consequently, an alternative way to compute $n_1 P_{ij}^{\mu_1\mu_1}$ is to find out $X_{1l_{ij}}$ and $X_{1r_{ij}}$. Inspired by this, we develop Algorithm 2 to accomplish the computation of $\{n_1 P_{ij}^{\mu_1\mu_1} \mid j = 1, \ldots, n_1\}$ in linear time. Through slightly modifying Algorithm 2, the computational complexity of $\{n_1 P_{ij}^{\mu_1\mu_1} + n_2 P_{ij}^{\mu_1\mu_2} \mid j = 1, \ldots, n_1\}$ also reduces to $O(N)$, and

Algorithm 1 Optimized algorithm for computing the row-wise rank matrices of $D^{\mathcal{X}_s^*\mathcal{X}_s^*}$, $D^{\mathcal{X}_t^*\mathcal{X}_t^*}$, and $D^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*}$

Require: The sample size of the s-th group n_s, the cumulative sample size vector C, the order information matrix $I^{\mathcal{X}\mathcal{X}}$ as well as the N-dimensional vector G.

Initialize all element of the row-wise rank matrices $R^{\mathcal{X}_s^*\mathcal{X}_s^*}$/ $R^{\mathcal{X}_t^*\mathcal{X}_t^*}$/ $R^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*}$ of $D^{\mathcal{X}_s^*\mathcal{X}_s^*}$/ $D^{\mathcal{X}_t^*\mathcal{X}_t^*}$/ $D^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*}$ with 0.
for $i = 1, \ldots, N$ **do**
  $g \leftarrow L_i^*$.
  if $g = s$ **or** $g = t$ **then**
    $rank_1 \leftarrow 1, rank_2 \leftarrow 1, rank_3 \leftarrow 1$.
    for $j = 1, \ldots, N$ **do**
      $o \leftarrow I_{i,j}^{\mathcal{X}\mathcal{X}}$, $g' \leftarrow L_o^*$.
      if $g = g'$ **then**
        if $g = s$ **then**
          $r \leftarrow G_i - C_s, c \leftarrow G_o - C_s$,
          $R_{rc}^{\mathcal{X}_s^*\mathcal{X}_s^*} \leftarrow rank_1, R_{rc}^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*} \leftarrow rank_3$,
          $rank_1 \leftarrow rank_1 + 1$.
        else if $g = t$ **then**
          $r \leftarrow G_i - C_t, c \leftarrow G_o - C_t$,
          $r' \leftarrow G_i - C_t + n_s, c' \leftarrow G_o - C_t + n_s$,
          $R_{rc}^{\mathcal{X}_t^*\mathcal{X}_t^*} \leftarrow rank_2, R_{r'c'}^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*} \leftarrow rank_3$,
          $rank_2 \leftarrow rank_2 + 1$.
        end if
      else
        if $g = s$ **and** $g' = t$ **then**
          $r \leftarrow G_i - C_s, c \leftarrow G_o - C_t + n_s$.
        else if $g = t$ **and** $g' = s$ **then**
          $r \leftarrow G_i - C_t + n_s, c \leftarrow G_o - C_s$.
        end if
        $R_{rc}^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*} \leftarrow rank_3$.
      end if
      $rank_3 \leftarrow rank_3 + 1$.
    end for
  end if
end for
return $R^{\mathcal{X}_s^*\mathcal{X}_s^*}, R^{\mathcal{X}_t^*\mathcal{X}_t^*}$, and $R^{\mathcal{X}_{s,t}^*\mathcal{X}_{s,t}^*}$.

hence, the computational complexity of $\{(n_1 P_{ij}^{\mu_1\mu_1}, n_1 P_{ij}^{\mu_1\mu_2}) \mid i, j = 1, \ldots, n_1\}$ reduces to $O(N^2)$. Similarly, the time complexity of $\{(n_1 P_{kl}^{\mu_2\mu_1}, n_2 P_{kl}^{\mu_2\mu_2}) \mid i, j = 1, \ldots, n_2\}$ is $O(N^2)$. In summary, the computational complexity of BD_N is $O(N^2)$ for univariate distributions, so are $\mathrm{BD}_N^{\mathcal{S}_K}$, $\mathrm{BD}_N^{\mathcal{MS}_K}$, and $\mathrm{BD}_N^{\mathcal{M}_K}$. As for $\mathrm{BCov}_{\omega,N}^K(K = 2)$, by slightly modifying Algorithm 2, the time complexity of computing $\{(NP_{ij}^{\mu_1}, NP_{ij}^{\mu_2}) \mid i, j = 1, \ldots, N\}$ reduces to $O(N^2)$ when X_{i1}, X_{i2} are univariate. Further, retaining (l_{ij}^1, r_{ij}^1) and (l_{ij}^2, r_{ij}^2) which satisfy $NP_{ij}^{\mu_1} = r_{ij}^1 - l_{ij}^1 + 1$ and $NP_{ij}^{\mu_2} = r_{ij}^2 - l_{ij}^2 + 1$, we can accomplish the computation of $\{P_{ij}^{\mu} \mid i, j = 1, \ldots, N\}$ within quadratic time by Algorithm 3. The key of Algorithm 3 is employing the inclusion-

Algorithm 2 Fast algorithm for $\{n_1 P_{ij}^{\mu_1\mu_1} \mid j = 1, \ldots, n_1\}$ in the univariate case

Require: A sorted array $\{X_{11}, \ldots, X_{1n_1}\}$ with ascending order.

Initialize $l = 1, r = n_1$.
while $l \leq r$ **do**
  if $X_{1r} - X_{1i} \geq X_{1i} - X_{1l}$ **then**
    $n_1 P_{ir}^{\mu_1\mu_1} \leftarrow r - l + 1$,
    $r \leftarrow r - 1$.
  else
    $n_1 P_{il}^{\mu_1\mu_1} \leftarrow r - l + 1$,
    $l \leftarrow l + 1$.
  end if
end while
return $\{n_1 P_{ij}^{\mu_1\mu_1} \mid j = 1, \ldots, n_1\}$

Algorithm 3 Fast algorithm for $\{P_{ij}^{\mu} \mid i, j = 1, \ldots, N\}$ in the univariate case

Require: A set including N bivariate observations $\{(X_{i1}, X_{i2}) \mid i = 1, \ldots, N\}$; a set containing lower and upper bound pairs $\{(l_{ij}^1, r_{ij}^1), (l_{ij}^2, r_{ij}^2) \mid i, j = 1, \ldots, N\}$.

Compute the rank of $\{X_{11}, \ldots, X_{N1}\}$ and $\{X_{12}, \ldots, X_{N2}\}$, respectively, and denote them as $\{r_1^1, \ldots, r_N^1\}$ and $\{r_1^2, \ldots, r_N^2\}$.
Pay $O(N^2)$ time to compute the values of bivariate empirical cumulative distribution $F_N(x, y) = \frac{1}{N}\sum_{t=1}^{N} I(r_t^1 \leq x, r_t^2 \leq y)$ on $\{(i, j) \mid i, j = 1, \ldots, N\}$, and set $F_N(0, y) = F_N(x, 0) = 0$.
for $i = 1, \ldots, N$ **do**
  for $j = 1, \ldots, N$ **do**
    $P_{ij}^{\mu} \leftarrow F_N(r_{ij}^1, r_{ij}^2) - F_N(l_{ij}^1 - 1, r_{ij}^2) - F_N(r_{ij}^1, l_{ij}^2 - 1) + F_N(l_{ij}^1 - 1, l_{ij}^2 - 1)$.
  end for
end for
return $\{P_{ij}^{\mu} \mid i, j = 1, \ldots, N\}$

exclusion principle which is also used in Heller, Heller, Kaufman, Brill, and Gorfine (2016). In summary, with Algorithms 2 and 3, the time complexity of $\mathrm{BCov}_{\omega,N}^{K}(K = 2)$ is $O(N^2)$.

4. The Ball package

In this section, we introduce the **Ball** package, an R package which implements the ball test statistics and procedures introduced in Section 2 as well as the algorithms illustrated in Section 3. The core of **Ball** is programmed in C to improve computational efficiency. Moreover, we employ the parallel computing technique in the ball test procedures to speed up the computation. To be specific, during the permutation procedure, multiple ball test statistics are concurrently calculated with OpenMP which supports the multi-platform shared memory multiprocessing programming in C level. Aside from speed, the **Ball** package is concise and user-friendly. An R user can conduct the K-sample and independence tests via `bd.test` and `bcov.test` functions in the **Ball** package, respectively.

We supply instructions and primary usages for `bd.test` and `bcov.test` functions in Sec-

tion 4.1. Additionally, in Section 4.2, two real data examples are provided to demonstrate how to use these functions to tackle data drawn from manifold spaces.

4.1. Functions `bd.test` and `bcov.test`

The functions `bd.test` and `bcov.test` are programmed for the K-sample test and test of mutual independence problems, respectively. The default usages of two functions are:

```
bd.test.default(x, y = NULL, num.permutations = 99, distance = FALSE,
  size = NULL, seed = 1, num.threads = 0, kbd.type = "sum", ...)
bcov.test.default(x, y = NULL, num.permutations = 99, distance = FALSE,
  weight = FALSE, seed = 1, num.threads = 0, ...)
```

The arguments of the two functions are described as follows.

- `x`: A numeric vector, matrix, or data frame, or a list containing at least two numeric vectors, matrices, or data frames.
- `y`: A numeric vector, matrix, or data frame.
- `num.permutations`: The number of permutation replications, must be a non-negative integer. Default: `num.permutations = 99`.
- `distance`: If `distance = TRUE`, `x` is considered as a distance matrix, or a list containing distance matrices in the test of mutual independence problem. And `y` is considered as a distance matrix only in the test of independence problem. Default: `distance = FALSE`.
- `size`: A vector recording the sample size of K groups. It is only available for `bd.test`.
- `weight`: A logical value or character string used to choose the form of $\hat{\omega}_k(X_{ik}, X_{jk})$. If `weight = FALSE` or `weight = "constant"`, the result of BCov_N^K-based test is displayed. Alternatively, `weight = TRUE` or `weight = "probability"` indicates the probability weight is chosen while setting `weight = "chisquare"` means selecting the Chi-square weight. From the definitions of $\mathrm{BCov}_N^K, \mathrm{BCov}_{\Delta,N}^K$, and $\mathrm{BCov}_{\chi^2,N}^K$, they are quite similar and could be computed at the same time. Therefore, `bcov.test` simultaneously computes $\mathrm{BCov}_N^K, \mathrm{BCov}_{\Delta,N}^K$, $\mathrm{BCov}_{\chi^2,N}^K$, and their corresponding p values. Users could get other statistics and p values from the `complete.info` element of output without re-running `bcov.test`. At present, this arguments is only available for `bcov.test`. Any unambiguous substring can be given. Default: `weight = FALSE`.
- `seed`: The random seed. Default: `seed = 1`.
- `num.threads`: The number of threads used in the ball test procedures. If `num.threads = 0`, all available cores are used. Default: `num.threads = 0`.
- `kdb.type`: A character string used to choose the K-sample BD statistics. Setting `kdb.type = "sum"`, `kdb.type = "summax"`, or `kdb.type = "max"`, we choose $\mathrm{BD}_N^{\mathcal{S}_K}$, $\mathrm{BD}_N^{\mathcal{MS}_K}$, or $\mathrm{BD}_N^{\mathcal{M}_K}$ to test the equality of distributions. Notice that, the three K-sample BD statistics are very similar, since their only difference is the aggregation strategy for the two-sample BD statistics. Therefore, the `bd.test` function simultaneously computes $\mathrm{BD}_N^{\mathcal{S}_K}, \mathrm{BD}_N^{\mathcal{MS}_K}$, $\mathrm{BD}_N^{\mathcal{M}_K}$, and their corresponding p values. Users could get other

statistics and p values from the `complete.info` element of output without re-running `bd.test`. This arguments is only available for the `bd.test` function. Any unambiguous substring can be given. Default: `kdb.type = "sum"`.

If `num.permutations > 0`, the output is a '`htest`' class object similar to the object returned by the `t.test` function. The output object contains the ball test statistic value (`statistic`), the p value of the test (`p.value`), the number of permutation replications (`replicates`), a vector recording the sample size (`size`), a `list` mainly containing two vectors, where the first vector is $\mathrm{BD}_N^{\mathcal{S}_K}, \mathrm{BD}_N^{\mathcal{MS}_K}$, and $\mathrm{BD}_N^{\mathcal{M}_K}$ for `bd.test` or BCov_N^K, $\mathrm{BCov}_{\Delta,N}^K$, and $\mathrm{BCov}_{\chi^2,N}^K$ for `bcov.test`, and the second vector is the corresponding p values of the tests (`complete.info`), a character string declaring the alternative hypothesis (`alternative`), a character string describing the hypothesis test (`method`), and a character string giving the name and helpful information of the data (`data.name`); if `num.permutations = 0`, only the ball test statistic value is returned.

To give quick examples, we carry out the ball test on two synthetic datasets to check whether the null hypothesis can be rejected when distributions are different or random variables are associated indeed.

For the K-sample test problem ($K = 2$), we generate two univariate datasets $\{X_{1i}\}_{i=1}^{50}$ and $\{X_{2i}\}_{i=1}^{50}$ with different location parameters, where

$$X_{1i} \sim N(0,1),\ X_{2i} \sim N(1,1),\ i = 1, \ldots, 50.$$

The detailed R code is as follows.

```
R> library("Ball")
R> set.seed(1)
R> x <- rnorm(50)
R> y <- rnorm(50, mean = 1)
R> bd.test(x = x, y = y)

    2-sample Ball Divergence Test (Permutation)

data:  x and y
number of observations = 100, group sizes: 50 50
replicates = 99
bd.constant = 0.092215, p-value = 0.01
alternative hypothesis: distributions of samples are distinct
```

In this example, the `bd.test` function yields the BD_N value 0.0922 and the p value 0.01 when the permutation replicate is 99. At the usual significance level of 0.05, we should reject the null hypothesis. Thus, the test result is concordant to the data generation mechanism.

In regard to the test of mutual independence problem, we sample 100 i.i.d. observations from the multivariate normal distribution to perform the mutual independence test based on BCov_N^K. The detailed R code is demonstrated below.

```
R> library("mvtnorm")
R> set.seed(1)
```

```
R> cov_mat <- matrix(0.3, 3, 3)
R> diag(cov_mat) <- 1
R> data_set <- rmvnorm(n = 100, mean = c(0, 0, 0), sigma = cov_mat)
R> data_set <- as.list(as.data.frame(data_set))
R> bcov.test(x = data_set)

    Ball Covariance test of mutual independence (Permutation)

data:  data_set
number of observations = 100
replicates = 99, weight: constant
bcov.constant = 0.00063808, p-value = 0.03
alternative hypothesis: random variables are dependent
```

The output of `bcov.test` shows that BCov_N^K is 6.38×10^{-4} when the constant weight is used. The p value is 0.02, and at the usual significance level of 0.05, we conclude that the three univariate variables are mutually dependent.

4.2. Examples

Wind direction dataset

We consider the hourly recorded wind speed and wind direction in the Atlantic coast of Galicia in winter from 2003 until 2012, provided in the R package **NPCirc** (Oliveira, Crujeiras, and Rodríguez-Casal 2014). In this dataset, there exist 19488 observations, and each observation includes six variables: day, month, year, hour, wind speed, and wind direction (in degrees). It is of interest to see whether there are any wind direction differences between the first and last weeks of 2007–2008 winter season. We select the wind direction records from November 1 to November 7, 2007, as the first week data, and the records from January 25 to January 31, 2008, as the last week data. The missing records in two weeks are discarded.

```
R> library("Ball")
R> data("speed.wind", package = "NPCirc")
R> index1 <- which(speed.wind[["Year"]] == 2007 &
+    speed.wind[["Month"]] == 11 & speed.wind[["Day"]] %in% 1:7)
R> index2 <- which(speed.wind[["Year"]] == 2008 &
+    speed.wind[["Month"]] == 1 & speed.wind[["Day"]] %in% 25:31)
R> d1 <- na.omit(speed.wind[["Direction"]][index1])
R> d2 <- na.omit(speed.wind[["Direction"]][index2])
```

Each wind direction is one-to-one transformed to a two-dimensional point in the Cartesian coordinates, and then, the difference of any two points is measured by the great-circle distance which is programmed in the `nhdist` function in the **Ball** package.

```
R> theta <- c(d1, d2) / 360
R> dat <- cbind(cos(theta), sin(theta))
R> dx <- nhdist(dat, method = "geo")
```

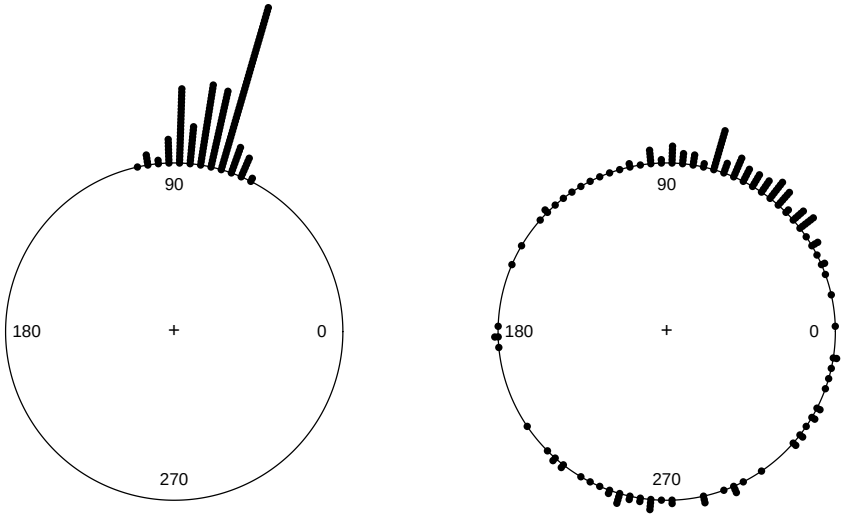

Figure 1: The raw circular data plot of the hourly wind direction dataset. The left panel is corresponding to the first-week wind directions in 2007–2008 winter and the right panel is corresponding to the last week's.

In the final step, we pass the distance matrix and the sample size of two groups to the arguments `x` and `size`, and set `distance = TRUE` to declare that the object passed to the arguments `x` is a distance matrix.

```
R> size_vec <- c(length(d1), length(d2))
R> bd.test(x = dx, size = size_vec, distance = TRUE)

    2-sample Ball Divergence Test (Permutation)

data:  dx
number of observations = 335, group sizes: 168 167
replicates = 99, weight: constant
bd.constant = 0.29515, p-value = 0.01
alternative hypothesis: distributions of samples are distinct
```

As can be seen from the output information of `bd.test`, BD_N is 0.2911 and the p value is 0.01. Consequently, at the usual significance level of 0.05, we should reject the null hypothesis. To further confirm our conclusion, we visualize the wind direction of two groups in Figure 1 with the R package **circular** (Lund and Agostinelli 2017). Figure 1 shows that the hourly wind directions in the first week is concentrated around the 90 degrees but the wind directions of the last week are widely dispersed.

```
R> library("circular")
R> par(mfrow = c(1, 2), mar = c(0, 0, 0, 0))
R> plot(circular(c(d1), units = "degrees"), cex = 0.8, bin = 100,
+    stack = TRUE, shrink = 1.3)
R> plot(circular(c(d2), units = "degrees"), cex = 0.8, bin = 100,
+    stack = TRUE, shrink = 1.3)
```

Brain fMRI dataset

We examine a public fMRI dataset from the 1000 Functional Connectomes Project (https://www.nitrc.org/frs/?group_id=296). This project calls on the principal investigators from the member site to donate neuroimaging data such that the broader imaging community have complete access to a large-scale functional imaging dataset. Given the resting-state fMRI and demographics of 86 individuals donated from ICBM, it is of interest to evaluate whether age is associated with brain connectivity. To properly analyze the dataset, we carry out a preprocessing for the three fMRI of each individual with the **nilearn** (Abraham *et al.* 2014) package in the Python environment. The preprocessing for each individual includes four steps: (i) segment brain into a set of 116 cortical and subcortical regions for each fMRI with the Automated Anatomical Labeling template (Tzourio-Mazoyer *et al.* 2002); (ii) average the voxel-specific time series in each of these regions to form mean regional time series for each fMRI; (iii) compute the 116×116 Pearson correlation coefficient matrix of each fMRI with the 116 mean regional time series, where the Pearson correlation coefficient is a widely-used association measure in the neuroimaging literature (Ginestet, Li, Balachandran, Rosenberg, and Kolaczyk 2017; Ginestet and Simmons 2011; Bullmore and Sporns 2009); (iv) average the three 116×116 matrices of each observation in an element-wise manner and save the averaged matrix to disk such that it can be analyzed with R. In the R environment, the collection of the averaged matrix and demographics are combined into a `list` object, then it is saved to a disk as `niICBM.rda` file.

To achieve our goal, we compute the pairwise distance matrices of the averaged matrices and the age, then save them in the R objects `dx` and `dy`, respectively. Then, we pass `dx` and `dy` to `bcov.test` to perform an independence test, meanwhile, let `distance = TRUE` to declare what the arguments `x` and `y` accepted are distance matrices. The detailed R code is demonstrated below.

```
R> library("Ball")
R> library("CovTools")
R> load("niICBM.rda")
R> dx <- as.matrix(CovDist(niICBM[["spd"]]))
R> dy <- as.matrix(dist(niICBM[["covariate"]][["V3"]]))
R> bcov.test(x = dx, y = dy, distance = TRUE, weight = "prob")

    Ball Covariance test of independence (Permutation)

data:  dx and dy
number of observations = 86
replicates = 99, weight: probability
bcov.probability = 0.017138, p-value = 0.01
alternative hypothesis: random variables are dependent
```

The output message shows that $\mathrm{BCov}^K_{\Delta,N}$ is 0.0171 and the p value is smaller than the usual significance level 0.05, and thus, we conclude that brain connectivity is associated with age. This result is also revealed by recent research that age strongly effects structural brain connectivity (Damoiseaux 2017). In this example, we use the Euclidean distance to measure the difference between age, and the affine invariant Riemannian metric (Pennec, Fillard, and

Ayache 2006) to evaluate the structural difference between the averaged Pearson correlation coefficient matrices. The affine invariant Riemannian metric is implemented in **CovTools** (Lee and You 2019)

5. Numerical studies

In this section, the numerical studies are conducted to assess the performance of the ball test procedures for complex data, including directional data in hyper-sphere spaces, tree-structured data in tree metric spaces, symmetric positive definite matrix data in the space of symmetric positive-definite matrices, and functional data in $\mathcal{L}_\infty$ spaces. Besides, a runtime analysis is provided in Section 5.3. For comparison, we consider energy distance and HHG for the K-sample test problem, while distance covariance, distance multivariance, and HHG for the test of mutual independence problem. The permutation technique helps us obtain the empirical distributions of these statistics under the null hypothesis, and derive their p values. As suggested in Davison and Hinkley (1997), at least 99 and at most 999 random permutations should suffice, and hence, we compute the p value of each test based on 399 random permutations. All models in Sections 5.1 and 5.2 are repeated 500 times to estimate type-I errors and powers. In each replication, all methods use the same dataset and the same non-standard distance to make a fair comparison. The significance level is fixed at 0.05.

5.1. K-sample test

In this section, we investigate the performance of test statistics on revealing the distribution difference with two kinds of complex data, directional data and tree-structured data. They are frequently encountered by scientists interested in wind directions, marine currents (Marzio, Panzera, and Taylor 2014), and cancer evolution (Abbosh *et al.* 2017). To sample directional and tree-structured data, we use the `rmovMF` and `rtree` functions in the R packages **movMF** (Hornik and Grün 2014) and **ape** (Paradis, Schliep, and Schwartz 2019) to draw data from the von Mises-Fisher distribution $M(\mu, \kappa)$ and random tree distribution $T(n, \{b_i\}_{i=1}^n)$, where μ and κ are direction and concentration parameters while n and $\{b_i\}_{i=1}^n$ are the numbers of tree nodes and the branch lengths of tree. The dissimilarities of two directions and two trees are measured by the great-circle distance and the Kendall Colijn metric (Kendall and Colijn 2016), which are programmed in the `nhdist` and `multiDist` functions in the R packages **Ball** and **treespace** (Jombart, Kendall, Almagro-Garcia, and Colijn 2017).

We conduct the numerical analyses for directional data in Models 5.1.1, 5.1.3-5.1.5, and 5.1.9 while tree-structured data in other models. Models 5.1.1 and 5.1.2 are designed for type-I error evaluation, while other models are devoted to evaluating powers. More specifically, Models 5.1.3-5.1.8 focus on the case that any two groups are different, while Models 5.1.9 and 5.1.10 pay attention to the case that only one group is different to other groups. Without loss of generality, we let $K = 4$ for Models 5.1.1-5.1.8 and $K = 10$ for Models 5.1.9 and 5.1.10. Each group has the same sample size ranging from 10 to 50.

- Model 5.1.1: von Mises-Fisher distribution. The direction parameters are $\mu^w = \mu^x = \mu^y = \mu^z = (1, 0, 0)$ and the concentration parameters are $\kappa^w = \kappa^x = \kappa^y = \kappa^z = 30$.
- Model 5.1.2: Random tree distribution with fifteen nodes. $b_i^w, b_i^x, b_i^y, b_i^z, i = 1, \ldots, 15$ are independently sampled from the uniform distribution $U(0, 1)$.

- Model 5.1.3: von Mises-Fisher distribution. The direction parameters are $\mu^w = (0,1,1,1,1)$, $\mu^x = (2,1,1,1,1), \mu^y = (4,1,1,1,1), \mu^z = (6,1,1,1,1)$ and the concentration parameters are $\kappa^w = 1, \kappa^x = 2, \kappa^y = 3, \kappa^z = 4$.

- Model 5.1.4: Mixture von Mises-Fisher distribution. The direction parameters are $\mu^{w_1} = \mu^{x_1} = (1,0), \mu^{w_2} = \mu^{x_2} = (-1,0), \mu^{y_1} = \mu^{z_1} = (0,1), \mu^{y_2} = \mu^{z_2} = (0,-1)$ and the concentration parameters are $\kappa^{w_1} = \kappa^{w_2} = \kappa^{y_1} = \kappa^{y_2} = 30, \kappa^{x_1} = \kappa^{x_2} = \kappa^{z_1} = \kappa^{z_2} = 35$. The four mixture proportions of two von Mises-Fisher distributions are all 0.5.

- Model 5.1.5: Mixture von Mises-Fisher distribution. The direction parameters are $\mu^{w_1} = (1,0,0,0), \mu^{w_2} = (-1,0,0,0), \mu^{x_1} = (0,1,0,0), \mu^{x_2} = (0,-1,0,0), \mu^{y_1} = (0,0,1,0)$, $\mu^{y_2} = (0,0,-1,0), \mu^{z_1} = (0,0,0,1), \mu^{z_2} = (0,0,0,-1)$ and the concentration parameters are all 30. The four mixture proportions of two von Mises-Fisher distributions are all 0.5.

- Model 5.1.6: Random tree distribution with fifteen nodes. $b_i^w, b_i^x, b_i^y, b_i^z, i = 1, \ldots, 15$ are independently sampled from four different uniform distributions: $b_i^w \sim U(0, 0.25), b_i^x \sim U(0, 0.5), b_i^y \sim U(0, 0.75), b_i^z \sim U(0,1), i = 1, \ldots, 15$.

- Model 5.1.7: Random tree distribution with fifteen nodes. $b_i^w, b_i^x, b_i^y, b_i^z, i = 1, \ldots, 15$ are independently sampled from four different uniform distributions: $b_i^w \sim U(0, 12), b_i^x \sim U(2, 10), b_i^y \sim U(3, 8), b_i^z \sim U(4, 6), i = 1, \ldots, 15$.

- Model 5.1.8: Random tree distribution with fifteen nodes. $b_i^w, b_i^x, b_i^y, b_i^z, i = 1, \ldots, 15$ are independently sampled from four different F distributions: $b_i^w \sim F(2,2), b_i^x \sim F(2,3)$, $b_i^y \sim F(2,4)$, $b_i^z \sim F(2,5), i = 1, \ldots, 15$.

Since only one group is different to other groups, it is sufficient to specify the following Models by describing the distributions of two groups.

- Model 5.1.9: von Mises-Fisher distribution. The direction parameters are $\mu^x = (0,1,1,1,1)$, $\mu^y = (2,1,1,1,1)$ and the concentration parameters are $\kappa^x = \kappa^y = 3$.

- Model 5.1.10: Random tree distribution with fifteen nodes. $b_i^x, b_i^y, i = 1, \ldots, 15$ are independently sampled from two different uniform distribution: $b_i^x \sim U(0, 0.5), b_i^y \sim U(0,1), i = 1, \ldots, 15$.

The type-I error rates and power estimates are demonstrated in Figures 2 and 3. From Figure 2, all test methods can control the type-I error rates well around the significance level. Figure 3 shows that $\mathrm{BD}_N^{\mathcal{S}_K}, \mathrm{BD}_N^{\mathcal{MS}_K}$, or $\mathrm{BD}_N^{\mathcal{M}_K}$ outperforms energy distance and HHG in most cases. More specifically, $\mathrm{BD}_N^{\mathcal{S}_K}$ is generally superior to other methods when any two groups are different, and $\mathrm{BD}_N^{\mathcal{MS}_K}$ has an advantage when the relatively rare group distinctions increase the difficult of the K-sample test problem. As for $\mathrm{BD}_N^{\mathcal{M}_K}$, it is more stable compared with $\mathrm{BD}_N^{\mathcal{S}_K}$ and $\mathrm{BD}_N^{\mathcal{MS}_K}$. $\mathrm{BD}_N^{\mathcal{M}_K}$ is better than $\mathrm{BD}_N^{\mathcal{MS}_K}$ when any two groups are different, and better than $\mathrm{BD}_N^{\mathcal{S}_K}$ when the group distinctions are relatively rare.

In Models 5.1.4 and 5.1.5, the empirical powers of energy distance are low and not increasing, yet the K-sample BD statistics and HHG are well-performed. This is not surprising because the great-circle distance is not of strong negative type. We provide an example to illustrate this result as follows. Without loss of generality, we simplify the distributions in Model 5.1.4

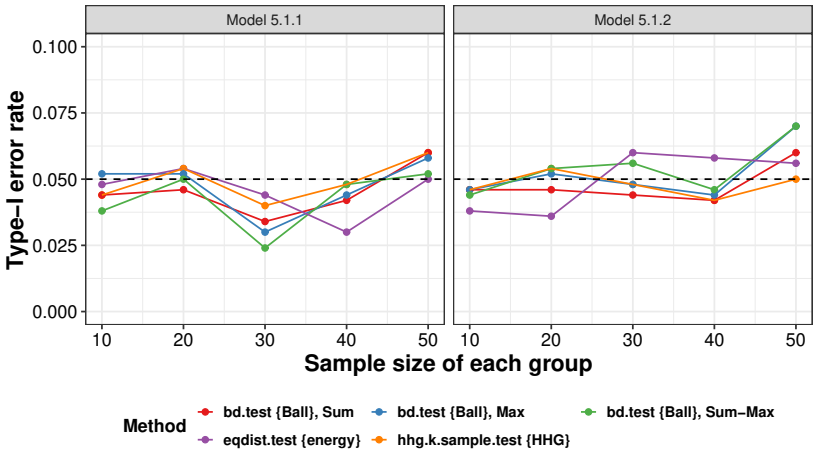

Figure 2: Type-I error rates of the five tests for the K-sample test problem. The black dashed line is the nominal significance level.

to the equal-probability Bernoulli distributions on a circle, where W, X take $(0, 1)$ and $(0, -1)$ while Y, Z take $(1, 0)$ and $(-1, 0)$. Then, the K-sample test problem could be considered as the two-sample test problem between groups X and Y. For the two groups X and Y, the means of the great-circle distance within the group are both $\pi/2$, so is the mean of the great-circle distance between the two groups. The energy distance between X and Y is zero according to its definition (Székely and Rizzo 2013)

$$\mathcal{E}(X, Y) = 2E(d(X, Y)) - E(d(X, X')) - E(d(Y, Y')), \tag{4}$$

where X' and Y' are i.i.d. copy of X and Y, respectively. Thus, energy distance fails to detect the distribution difference. On the contrary, BD_N is larger than zero since all observations in $B((0, 1), (0, 1)) \cup B((0, -1), (0, -1))$ come from the group X and all observations in $B((1, 0), (1, 0)) \cup B((-1, 0), (-1, 0))$ come from the group Y. The result of Model 5.1.5 can be interpreted similarly.

5.2. Test of mutual independence

In this section, we evaluate the performance of test methods on detecting the relationship among complex random objects. The complex random objects attracting our attention are symmetric positive definite matrix and functional curve, commonly encountered in contemporary statistical research, for instance, Dryden, Koloydenko, and Zhou (2009) and Wang, Chiou, and Müller (2016). We generate the two types of random objects with the `genPositiveDefMat` function in the R package **clusterGeneration** (Qiu and Joe 2020) and a series of functions in the R package **fda** (Ramsay, Graves, and Hooker 2020). The `genPositiveDefMat` function can generate a random $d \times d$ symmetric positive definite matrix $SPD_d(\lambda, \rho)(\lambda > 0, \rho > 1)$ whose eigenvalues range from λ to $\rho\lambda$. The R functions in **fda** help construct the functional curve $f(t; \mathbf{c}) = (1, \sin t, \cos t)\mathbf{c}$ and acquire 17 observed points which are equally spaced at interval $[0, 8\pi]$, where $\mathbf{c}$ is a coefficient vector. The typical dissimilarity measurements of two symmetric positive definite matrices and two functional curves are the affine invariant Riemannian metric and the L_∞ norm, which are implemented in the R packages **CovTools** and **fda.usc** (Febrero-Bande and de la Fuente 2012), respectively.

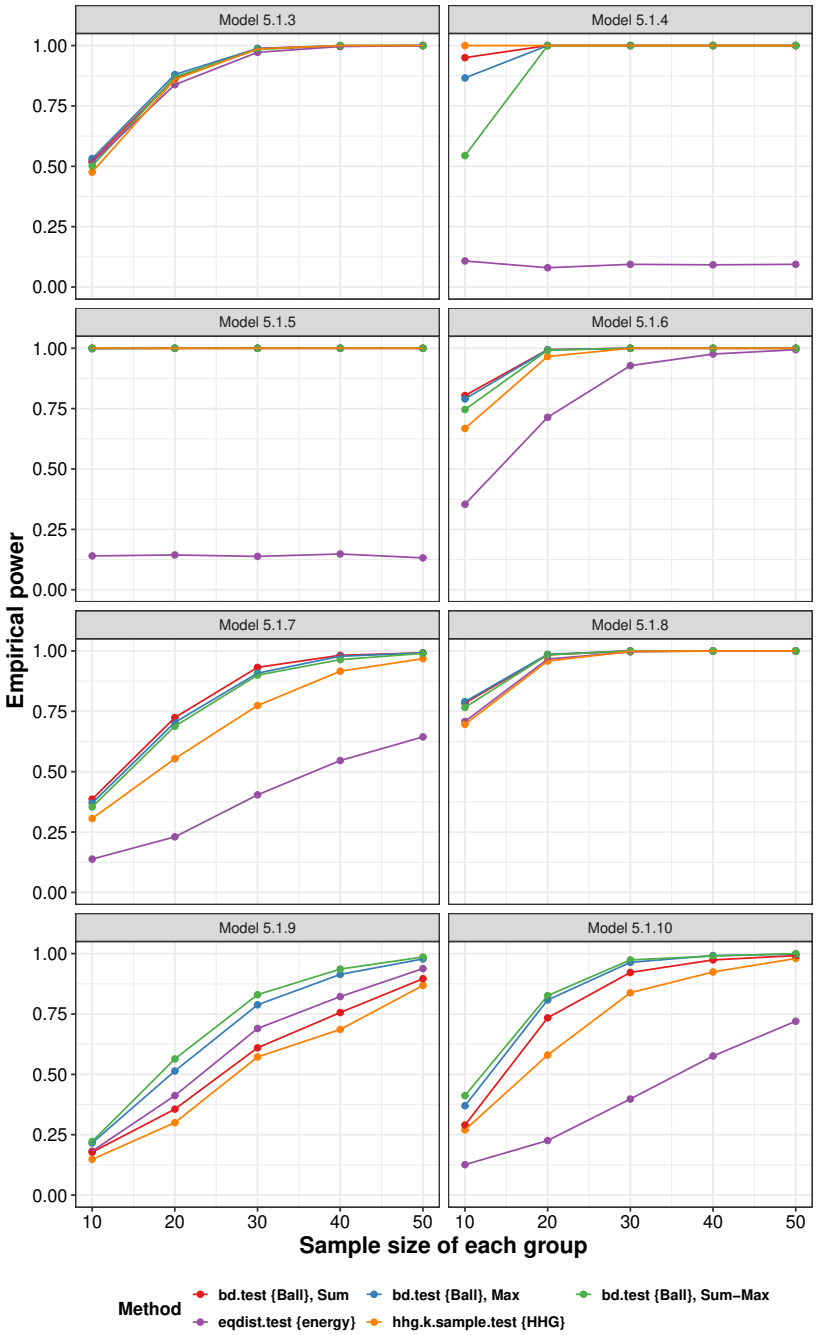

Figure 3: Power estimates of the five tests for the K-sample test problem.

We design Models 5.2.1-5.2.2 for the examination of type-I errors and Models 5.2.3-5.2.10 for the assessment of powers. Let the sample size increase from 40 to 120.

- Model 5.2.1: X, Y are independently sampled from the $SPD_{10}(1, 10)$.
- Model 5.2.2: $\mathbf{c}_1, \mathbf{c}_2$ are independently sampled from the multivariate uniform distribu-

tion on the cube $[0,1]^3$,

$$X(t) = f(t; \mathbf{c}_1), Y(t) = f(t; \mathbf{c}_2).$$

- Model 5.2.3: Z comes from the uniform distribution $U(0, \pi/2)$, and $\epsilon_1, \epsilon_2, \epsilon_3$ are independently sampled from the Chi-square distribution with the degree of freedom 1,

$$X \sim SPD_{10}\left(Z, 1+\epsilon_1\right), Y \sim SPD_{10}\left(Z+\epsilon_2, 1+\epsilon_3\right).$$

- Model 5.2.4: The distributions of $Z, \epsilon_1, \epsilon_2, \epsilon_3$ are the same as in Model 5.2.3.

$$X \sim SPD_{10}\left(Z, 1+\epsilon_1\right), Y \sim SPD_{10}\left(10\left|\cos\left(3Z\right)\right|+\epsilon_2, 1+\epsilon_3\right).$$

- Model 5.2.5: $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ are independent standard normal random vectors, and $\epsilon_1(t)$ and $\epsilon_2(t)$ are independent Gaussian processes,

$$\begin{aligned} &Z_1(t) = f(t; \mathbf{c}_1), Z_2(t) = f(t; \mathbf{c}_2), Z_3(t) = f(t; \mathbf{c}_3), \\ &X(t) = 4Z_1(t)Z_2(t)Z_3(t) + \epsilon_1(t), Y(t) = [Z_1(t) - Z_2(t)]^2 + 2Z_3(t) + \epsilon_2(t). \end{aligned}$$

- Model 5.2.6: X comes from the binomial distribution $B(1, 0.5)$, and $\epsilon(t)$ is a Gaussian process,

$$\begin{aligned} P(\mathbf{c} = (0,0,1)^\top \mid X = 0) = P(\mathbf{c} = (0,0,-1)^\top \mid X = 0) = 0.5, \\ P(\mathbf{c} = (0,1,0)^\top \mid X = 1) = P(\mathbf{c} = (0,-1,0)^\top \mid X = 1) = 0.5, \\ Y(t) = 10f(t; \mathbf{c}) + \epsilon(t). \end{aligned}$$

The following four models are constructed for evaluating the power of test methods in the test of mutual independence problem. To the best of our knowledge, only **multivariance** and **Ball** allow R users to perform the mutual independence test on datasets in metric spaces. And hence, we only compare **Ball** and **multivariance** below.

- Model 5.2.7: The distributions of $Z, \epsilon_1, \epsilon_2, \epsilon_3$ are the same as Model 5.2.3, and ϵ_4, ϵ_5 are independently drawn from Pareto distribution with location parameter 0.8 and shape parameter 1.

$$\begin{aligned} X &\sim SPD_{10}\left(Z, 1+\epsilon_1\right), \\ Y &\sim SPD_{10}\left(Z+\epsilon_4, 1+\epsilon_2\right), \\ Z &\sim SPD_{10}\left(Z+\epsilon_5, 1+\epsilon_3\right). \end{aligned}$$

- Model 5.2.8: W_1, W_2 are independently drawn from the t distribution with the degree of freedom 1,

$$\begin{aligned} X &\sim SPD_{10}(1, |W_1| + 1), \\ Y &\sim SPD_{10}(1, 1 + (W_1 - W_2)^2), \\ Z &\sim SPD_{10}(1, \exp\left(8\sin(W_1 - W_2)\right) + 1). \end{aligned}$$

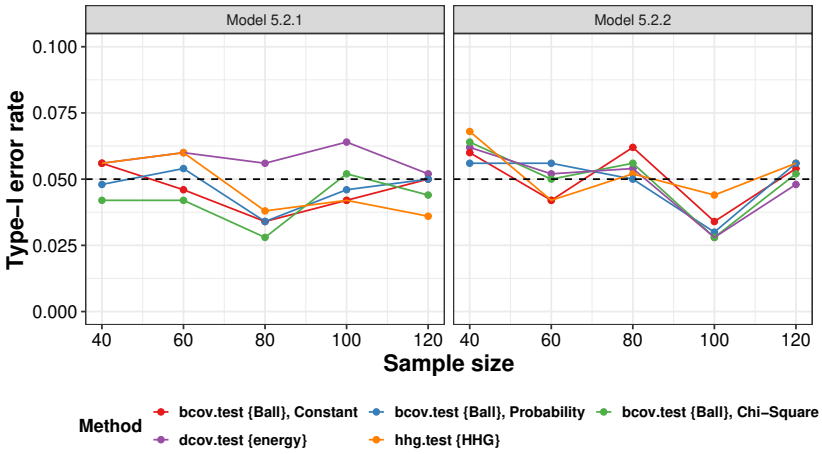

Figure 4: Type-I error rates of the five tests for the test of independence problem. The black dashed line is the nominal significance level.

- Model 5.2.9: Z_1, Z_2 are independently sampled from the binomial distribution $B(1, 0.5)$, $Z_3 = I(Z_1 = Z_2)$. Let $\mathbf{c}_0 = (0, 1, 0)^\top, \mathbf{c}_1 = (0, 0, 1)^\top$,

$$\begin{aligned} X(t) &= I(Z_1 = 0)f(t; \mathbf{c}_0) + I(Z_1 = 1)f(t; \mathbf{c}_1), \\ Y(t) &= I(Z_2 = 0)f(t; \mathbf{c}_0) + I(Z_2 = 1)f(t; \mathbf{c}_1), \\ Z(t) &= I(Z_3 = 0)f(t; \mathbf{c}_0) + I(Z_3 = 1)f(t; \mathbf{c}_1). \end{aligned}$$

- Model 5.2.10: X is sampled from the binomial distribution $B(1, 0.5)$, and $\epsilon_1(t)$ and $\epsilon_2(t)$ are independent Gaussian processes,

$$\begin{aligned} P(\mathbf{c} = (0, 0, 1)^\top \mid X = 0) = P(\mathbf{c} = (0, 0, -1)^\top \mid X = 0) = 0.5, \\ P(\mathbf{c} = (0, 1, 0)^\top \mid X = 1) = P(\mathbf{c} = (0, -1, 0)^\top \mid X = 1) = 0.5, \\ Y(t) = 10f(t;\mathbf{c}) + \epsilon_1(t),\ Z(t) = \epsilon_2(t). \end{aligned}$$

The type-I error rates and empirical power are displayed in Figures 4, 5, and 6. As shown in Figure 4, the type-I error rates of all methods are reasonably controlled around the significance level. From Figure 5, both the BCOV statistics and HHG are competitive and generally exceed distance covariance. From Figure 6, the three BCOV statistics successfully detect the complicated mutual dependence among multiple random objects, and their empirical powers increase as the sample size augments. It is worth noting that Model 5.2.9 is an example of pairwise independence with mutual dependence. The success in revealing the mutual dependence of Model 5.2.9 certifies the power of the BCOV statistics.

To shed a light on the performance difference of the three BCOV statistics, we compare their empirical powers in Models 5.2.3, 5.2.4, and 5.2.7. In Model 5.2.3, the lower bound of the eigenvalues of X is linearly associated with that of Y, and similarly in Model 5.2.7, except that the noise in Model 5.2.7 has an infinite first moment. $\mathrm{BCov}^K_{\chi^2,N}$ has a best performance in Model 5.2.3 on account of the nonlinearity of symmetric positive definite matrices spaces which slightly improves the nonlinearity between X and Y. In Model 5.2.7, BCov^K_N is superior to other methods owing to the approximately linear relationship among random objects and

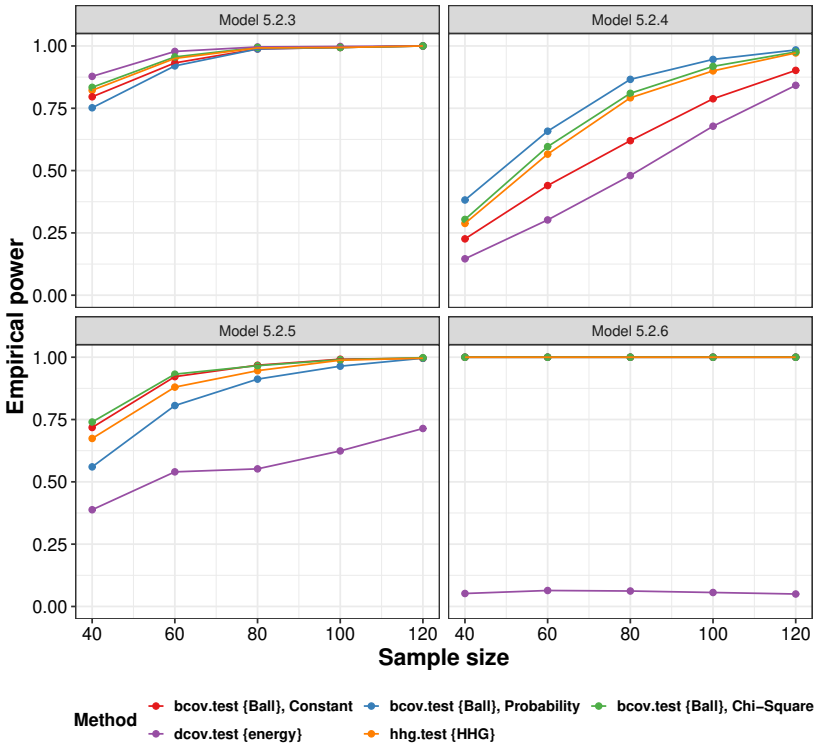

Figure 5: Power estimates of the five tests for the test of independence problem.

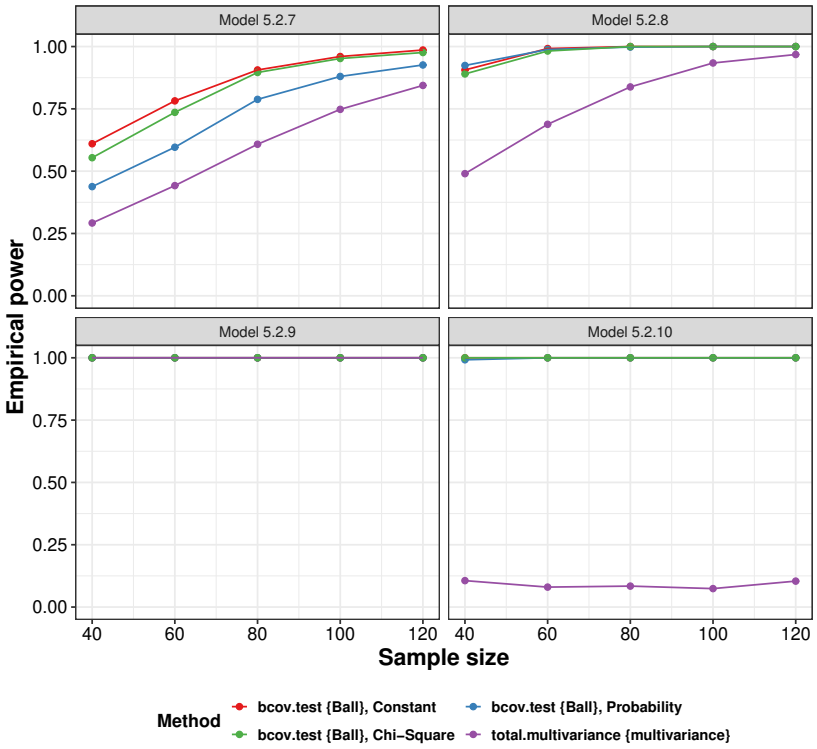

Figure 6: Power estimates of the four tests for the test of mutual independence problem.

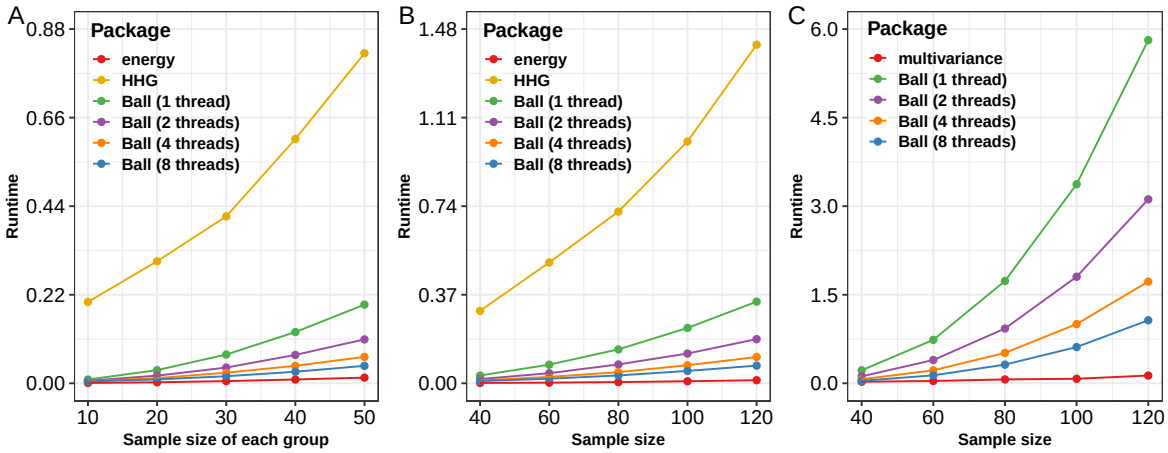

Figure 7: Runtime analysis for **energy**, **multivariance**, **HHG**, and **Ball** in the K-sample test and test of (mutual) independence problems in metric spaces. A) the K-sample test problem ($K = 4$), B) the test of independence problem, C) the test of mutual independence problem. The runtimes (y-axis) are measured in seconds.

its high robustness. As for Model 5.2.4, the lower bounds of the eigenvalues of X and Y have a strongly nonlinear relationship. At this point, $\mathrm{BCov}^K_{\Delta,N}$ turns to be the first place.

It is also worthwhile to take a good look at Models 5.2.6 and 5.2.10. In the two models, the empirical power of distance covariance and distance multivariance stay at a low level as the sample size increases, because the L_∞ norm is not of strong negative type. The following is an explanation of why distance covariance has an unsatisfactory performance in Model 5.2.6. Without loss of generality, we re-define $Y(t) = 10f(t; \mathbf{c})$, then denote $Y(t) \mid X = 0$ and $Y(t) \mid X = 1$ as $Y_1(t)$ and $Y_2(t)$. It is easy to verify that the distance covariance of $(X, Y(t))$ is the constant-multiple energy distance between groups $Y_1(t)$ and $Y_2(t)$. For the two groups $Y_1(t)$ and $Y_2(t)$, the means of the L_∞ norm within the group are both 10, so is the mean of the L_∞ norm between the two groups. According to Equation 4, the energy distance between $Y_1(t)$ and $Y_2(t)$ is 0, and thus, the distance covariance of $(X, Y(t))$ is 0, leading to the failure of detecting association. The performance of distance multivariate in Model 5.2.10 could be explained similarly.

5.3. Runtime analysis

We adopt Models 5.1.1, 5.2.1 and 5.2.7 in Sections 5.1 and 5.2 to assess the runtime performance of **energy** (1.7.6), **multivariance** (2.2.0), **HHG** (2.3.2), and **Ball** (1.3.11) using the **microbenchmark** package (Mersmann 2019). Here, all experiments are conducted with 20 replications, and the averaged runtimes are visualized in Figure 7. The benchmark is a 64-bit Windows platform with Intel Core i7 @ 3.60 GHz.

From Figure 7, **energy** is the fastest package in the K-sample test and the test of independence problems, and **multivariance** is the fastest package in the test of mutual independence problem. As the second fastest package, **Ball** is around four times faster than **HHG** in the K-sample test and test of independence problems when both of them use one thread, even though

$\mathrm{BCov}^{K}_{\chi^2,N}(K = 2)$ and HHG are asymptotically equivalent. Furthermore, we can cut the runtimes of **Ball** down around one third via doubling threads.

In summary, if runtimes are more concerned, **energy** or **multivariance** may be a desirable choice. Otherwise, **Ball** is a preferable choice due to its powerful performance in various complex data with fewer runtime increase, especially for the K-sample test and the test of independence problems.

6. Conclusion

We design a user-friendly R package **Ball** to help data scientists detect the distribution distinction and object association for complex data in metric spaces. Equipped with the novel algorithms, efficient C implementation, advanced multi-threaded technique, and elegant theoretical properties of the ball test statistics, the ball test procedures programmed in the **Ball** package can efficiently analyze complex data in metric spaces.

Future versions of the **Ball** package will endeavor to speed up the ball correlation based generic feature screening procedure (Pan *et al.* 2019). Furthermore, we intend to develop Python and Julia packages to help data scientists conduct the ball test procedures and ball screening procedure with their most familiar program languages.

Acknowledgments

We would like to thank referees for their valuable comments and suggestions which have substantially improved this article.

Dr. Wang's research is partially supported by the National Key Research and Development Program of China(2018YFC1315400), NSFC(11771462, 71991474), and the Science and Technology Program of Guangzhou, China(202002030129). Dr. Pan's research is partially supported by the National Natural Science Foundation of China (11701590), Natural Science Foundation of Guangdong Province of China (2017A030310053) and Young teacher program/Fundamental Research Funds for the Central Universities (17lgpy14). Dr. Zheng's research is partially supported by National Science Foundation, DMS-1830864. Zhu's research is partially supported by the Outstanding Graduate Student Innovation and Development Program of Sun Yat-Sen University (19lgyjs64).

References

Abbosh C, Birkbak NJ, Wilson GA, Jamal-Hanjani M, Constantin T, Salari R, Le Quesne J, Moore DA, Veeriah S, Rosenthal R, *et al.* (2017). "Phylogenetic ctDNA Analysis Depicts Early-Stage Lung Cancer Evolution." *Nature*, **545**(7655), 446–451. doi:10.1038/nature22364.

Abraham A, Pedregosa F, Eickenberg M, Gervais P, Mueller A, Kossaifi J, Gramfort A, Thirion B, Varoquaux G (2014). "Machine Learning for Neuroimaging with **scikit-Learn**." *Frontiers in Neuroinformatics*, **8**, 14. ISSN 1662-5196. doi:10.3389/fninf.2014.00014.

Bergsma W, Dassios A (2014). "A Consistent Test of Independence Based on a Sign Covariance Related to Kendall's Tau." *Bernoulli*, **20**(2), 1006–1028. doi:10.3150/13-bej514.

Berrett TB, Grose DJ, Samworth RJ (2018). ***IndepTest**: Nonparametric Independence Tests Based on Entropy Estimation*. R package version 0.2.0, URL https://CRAN.R-project.org/package=IndepTest.

Berrett TB, Samworth RJ (2019). "Nonparametric Independence Testing via Mutual Information." *Biometrika*, **106**(3), 547–566. ISSN 0006-3444. doi:10.1093/biomet/asz024.

Böttcher B (2020). ***multivariance**: Measuring Multivariate Dependence Using Distance Multivariance*. R package version 2.4.0, URL https://CRAN.R-project.org/package=multivariance.

Brill B, Heller Y, Heller R (2018). "Nonparametric Independence Tests and K-Sample Tests for Large Sample Sizes Using Package **HHG**." *The R Journal*, **10**(1), 424–438. doi:10.32614/rj-2018-008.

Bullmore E, Sporns O (2009). "Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems." *Nature Reviews Neuroscience*, **10**(3), 186–198. doi:10.1038/nrn2575.

Curran J (2018). ***Hotelling**: Hotelling's T^2 Test and Variants*. R package version 1.0-5, URL https://CRAN.R-project.org/package=Hotelling.

Damoiseaux JS (2017). "Effects of Aging on Functional and Structural Brain Connectivity." *NeuroImage*, **160**, 32–40. ISSN 1053-8119. doi:10.1016/j.neuroimage.2017.01.077. Functional Architecture of the Brain.

Davison AC, Hinkley DV (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. doi:10.1017/cbo9780511802843.

Drton M, Weihs L, Meinshausen N (2018). "Symmetric Rank Covariances: A Generalized Framework for Nonparametric Measures of Dependence." *Biometrika*, **105**(3), 547–562. doi:10.1093/biomet/asy021.

Dryden IL, Koloydenko A, Zhou D (2009). "Non-Euclidean Statistics for Covariance Matrices, with Applications to Diffusion Tensor Imaging." *The Annals of Applied Statistics*, **3**(3), 1102–1123. doi:10.1214/09-aoas249.

Efron B, Tibshirani RJ (1994). *An Introduction to the Bootstrap*. Chapman &Hall/CRC.

Febrero-Bande M, de la Fuente M (2012). "Statistical Computing in Functional Data Analysis: The R Package **fda.usc**." *Journal of Statistical Software*, **51**(4), 1–28. ISSN 1548-7660. doi:10.18637/jss.v051.i04.

Geenens G, Lafaye de Micheaux P (2021). "The Hellinger Correlation." *Journal of the American Statistical Association*. doi:10.1080/01621459.2020.1791132. Forthcoming.

Ginestet CE, Li J, Balachandran P, Rosenberg S, Kolaczyk ED (2017). "Hypothesis Testing for Network Data in Functional Neuroimaging." *The Annals of Applied Statistics*, **11**(2), 725–750. doi:10.1214/16-aoas1015.

Ginestet CE, Simmons A (2011). "Statistical Parametric Network Analysis of Functional Connectivity Dynamics during a Working Memory Task." *Neuroimage*, **55**(2), 688–704. doi:10.1016/j.neuroimage.2010.11.030.

Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola AJ (2012). "A Kernel Two-Sample Test." *Journal of Machine Learning Research*, **13**, 723–773.

Gretton A, Bousquet O, Smola A, Schölkopf B (2005). "Measuring Statistical Dependence with Hilbert-Schmidt Norms." In S Jain, HU Simon, E Tomita (eds.), *Algorithmic Learning Theory*, pp. 63–77. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-31696-1. URL https://link.springer.com/book/10.1007/11564089.

Heller R, Heller Y, Gorfine M (2013). "A Consistent Multivariate Test of Association Based on Ranks of Distances." *Biometrika*, **100**(2), 503–510. doi:10.1093/biomet/ass070.

Heller R, Heller Y, Kaufman S, Brill B, Gorfine M (2016). "Consistent Distribution-Free K-Sample and Independence Tests for Univariate Random Variables." *Journal of Machine Learning Research*, **17**(1), 978–1031.

Hornik K, Grün B (2014). "**movMF**: An R Package for Fitting Mixtures of Von Mises-Fisher Distributions." *Journal of Statistical Software*, **58**(10), 1–31. ISSN 1548-7660. doi:10.18637/jss.v058.i10.

Hothorn T, Hornik K, van de Wiel M, Zeileis A (2008). "Implementing a Class of Permutation Tests: The **coin** Package." *Journal of Statistical Software*, **28**(8), 1–23. ISSN 1548-7660. doi:10.18637/jss.v028.i08.

Jin Z, Yao S, Matteson DS, Shao X (2018). ***EDMeasure**: Energy-Based Dependence Measures*. R package version 1.2., URL https://CRAN.R-project.org/package=EDMeasure.

Jombart T, Kendall M, Almagro-Garcia J, Colijn C (2017). "**treespace**: Statistical Exploration of Landscapes of Phylogenetic Trees." *Molecular Ecology Resources*, **17**, 1385–1392. doi:10.1111/1755-0998.12676.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "**kernlab**: An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20. ISSN 1548-7660. doi:10.18637/jss.v011.i09.

Kendall M, Colijn C (2016). "Mapping Phylogenetic Trees to Reveal Distinct Patterns of Evolution." *Molecular Biology and Evolution*, **33**(10), 2735–2743. doi:10.1093/molbev/msw124.

Lê S, Josse J, Husson F (2008). "**FactoMineR**: An R Package for Multivariate Analysis." *Journal of Statistical Software*, **25**(1), 1–18. ISSN 1548-7660. doi:10.18637/jss.v025.i01.

Lee K, You K (2019). ***CovTools**: Statistical Tools for Covariance Analysis*. R package version 0.5.3, URL https://CRAN.R-project.org/package=CovTools.

Lund U, Agostinelli C (2017). ***circular**: Circular Statistics*. R package version 0.4-93, URL https://CRAN.R-project.org/package=circular.

Lyons R (2013). "Distance Covariance in Metric Spaces." *The Annals of Probability*, **41**(5), 3284–3305. doi:10.1214/12-aop803.

Marzio MD, Panzera A, Taylor CC (2014). "Nonparametric Regression for Spherical Data." *Journal of the American Statistical Association*, **109**(506), 748–763. doi:10.1080/01621459.2013.866567.

Mersmann O (2019). ***microbenchmark**: Accurate Timing Functions*. R package version 1.4-7, URL https://CRAN.R-project.org/package=microbenchmark.

Oliveira M, Crujeiras R, Rodríguez-Casal A (2014). "**NPCirc**: An R Package for Nonparametric Circular Methods." *Journal of Statistical Software*, **61**(9), 1–26. ISSN 1548-7660. doi:10.18637/jss.v061.i09.

Pan W, Tian Y, Wang X, Zhang H (2018). "Ball Divergence: Nonparametric Two Sample Test." *The Annals of Statistics*, **46**(3), 1109–1137. doi:10.1214/17-aos1579.

Pan W, Wang X, Xiao W, Zhu H (2019). "A Generic Sure Independence Screening Procedure." *Journal of the American Statistical Association*, **114**(526), 928–937. doi:10.1080/01621459.2018.1462709.

Pan W, Wang X, Zhang H, Zhu H, Zhu J (2020). "Ball Covariance: A Generic Measure of Dependence in Banach Space." *Journal of the American Statistical Association*, **115**(529), 307–317. doi:10.1080/01621459.2018.1543600.

Paradis E, Schliep K, Schwartz R (2019). "**ape** 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in R." *Bioinformatics*, **35**(3), 526–528. doi:10.1093/bioinformatics/bty633.

Pennec X, Fillard P, Ayache N (2006). "A Riemannian Framework for Tensor Computing." *International Journal of Computer Vision*, **66**(1), 41–66. ISSN 1573-1405. doi:10.1007/s11263-005-3222-z.

Pfister N, Peters J (2019). ***dHSIC**: Independence Testing via Hilbert Schmidt Independence Criterion*. R package version 2.1, URL https://CRAN.R-project.org/package=dHSIC.

Qiu W, Joe H (2020). ***clusterGeneration**: Random Cluster Generation (with Specified Degree of Separation)*. R package version 1.3.7, URL https://CRAN.R-project.org/package=clusterGeneration.

Ramsay JO, Graves S, Hooker G (2020). ***fda**: Functional Data Analysis*. R package version 5.1.7, URL https://CRAN.R-project.org/package=fda.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rizzo ML, Székely GJ (2019). ***energy**: E-Statistics: Multivariate Inference via the Energy of Data*. R package version 1.7-7, URL https://CRAN.R-project.org/package=energy.

Scholz F, Zhu A (2019). ***kSamples**: K-Sample Rank Tests and Their Combinations*. R package version 1.2-9, URL https://CRAN.R-project.org/package=kSamples.

Sejdinovic D, Sriperumbudur B, Gretton A, Fukumizu K (2013). "Equivalence of Distance-Based and RKHS-Based Statistics in Hypothesis Testing." *The Annals of Statistics*, **41**(5), 2263–2291. doi:10.1214/13-aos1140.

Strasser H, Weber C (1999). "The Asymptotic Theory of Permutation Statistics." *Mathematical Methods of Statistics*, **8**, 220–250.

Székely GJ, Rizzo ML (2004). "Testing for Equal Distributions in High Dimension." *InterStat*, **5**(16.10), 1249–1272.

Székely GJ, Rizzo ML (2013). "Energy Statistics: A Class of Statistics Based on Distances." *Journal of Statistical Planning and Inference*, **143**(8), 1249–1272. ISSN 0378-3758. doi:10.1016/j.jspi.2013.03.018.

Székely GJ, Rizzo ML, Bakirov NK (2007). "Measuring and Testing Dependence by Correlation of Distances." *The Annals of Statistics*, **35**(6), 2769–2794. doi:10.1214/009053607000000505.

Tzourio-Mazoyer N, Landeau B, Papathanassiou D, Crivello F, Etard O, Delcroix N, Mazoyer B, Joliot M (2002). "Automated Anatomical Labeling of Activations in **SPM** Using a Macroscopic Anatomical Parcellation of the MNI MRI Single-Subject Brain." *Neuroimage*, **15**(1), 273–289. doi:10.1006/nimg.2001.0978.

Verbyla P, Desgranges NIB, Wernisch L (2017). ***kpcalg**: Kernel PC Algorithm for Causal Structure Detection*. R package version 1.0.1, URL https://CRAN.R-project.org/package=kpcalg.

Wang JL, Chiou JM, Müller HG (2016). "Functional Data Analysis." *Annual Review of Statistics and Its Application*, **3**(1), 257–295. doi:10.1146/annurev-statistics-041715-033624.

Wang X, Pan W, Zhang H, Zhu H, Tian Y, Xiao W, Liu C, Zhu J (2021). ***Ball**: Statistical Inference and Sure Independence Screening via Ball Statistics*. R package version 1.3.11, URL https://CRAN.R-project.org/package=Ball.

Weihs L (2019). ***TauStar**: Efficient Computation and Testing of the Bergsma-Dassios Sign Covariance*. R package version 1.1.4, URL https://CRAN.R-project.org/package=TauStar.

A. Merge sort implementation

Merge sort is a classical divide-and-conquer algorithm for sorting. It recursively splits the value array in half until all subarrays only have one element, then merges those subarrays to a sorted array. To adapt to the "count of the smaller number after self" problem, merge sort uses an auxiliary equal-size number array to record the numbers of the smaller element after self. Initialized all elements with 0, the number array is split and merged with the value array. In the merging stage, if an element of the left side value array is to be merged, then, the merged elements of the right side value array must be no larger than the element to be merged. And hence, the corresponding element in the left side number array should add the number of the merged elements of the right side value array. Implemented with C in the **Ball** package, the solution of "count of the numbers after self" problem is given below.

```
C> void count_smaller_number_after_self_solution(double *value, int *number,
+                                                 const int num) {
+     int index[num];
+     for (int i = 0; i < num; ++i) {
+         index[i] = i;
+     }
+     merge_sort(value, index, number, 0, num - 1);
+ }
C> void merge_sort(double *value, int *index, int *number, int start, int end) {
+     if (end - start < 1) return;
+     int mid = (start + end) >> 1;
+     merge_sort(value, index, number, start, mid);
+     merge_sort(value, index, number, mid + 1, end);
+     merge(value, index, number, start, mid, end);
+ }
C> void merge(double *value, int *index, int *number, int start, int mid, int end) {
+     const int left_size = mid - start + 1, right_size = end - mid;
+     double left[left_size], right[right_size];
+     int left_index[left_size], right_index[right_size];
+     int left_merged = 0, right_merged = 0, total_merged = 0;
+     for (int i = start; i <= mid; ++i) {
+         left[i - start] = value[i];
+         left_index[i - start] = index[i];
+     }
+     for (int i = mid + 1; i <= end; ++i) {
+         right[i - mid - 1] = value[i];
+         right_index[i - mid - 1] = index[i];
+     }
+     while (left_merged < left_size && right_merged < right_size) {
+         if (left[left_merged] < right[right_merged]) {
+             number[left_index[left_merged]] += right_merged;
+             value[start + total_merged] = left[left_merged];
+             index[start + total_merged] = left_index[left_merged];
+             ++left_merged;
```

```
+             ++total_merged;
+         } else {
+             value[start + total_merged] = right[right_merged];
+             index[start + total_merged] = right_index[right_merged];
+             ++right_merged;
+             ++total_merged;
+         }
+     }
+     while (left_merged < left_size) {
+         number[left_index[left_merged]] += right_merged;
+         value[start + total_merged] = left[left_merged];
+         index[start + total_merged] = left_index[left_merged];
+         ++left_merged;
+         ++total_merged;
+     }
+     while (right_merged < right_size) {
+         value[start + total_merged] = right[right_merged];
+         index[start + total_merged] = right_index[right_merged];
+         ++right_merged;
+         ++total_merged;
+     }
+ }
```

Affiliation:

Jin Zhu, Wenliang Pan
Department of Statistical Science, School of Mathematics
Southern China Center for Statistical Science
Sun Yat-Sen University
510275 Guangzhou, GD, China
E-mail: zhuj37@mail2.sysu.edu.cn, panwliang@mail.sysu.edu.cn

Wei Zheng
Department of Business Analytics and Statistics
University of Tennessee
Knoxville, Tennessee 37996, United States of America
E-mail: wzheng9@utk.edu

Xueqin Wang *(corresponding author)*
Department of Statistics and Finance/International Institute of Finance
School of Management
University of Science and Technology of China
Hefei, AH 230026, China
E-mail: wangxq20@ustc.edu.cn

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
March 2021, Volume 97, Issue 6 *Submitted:* 2018-11-08
doi:10.18637/jss.v097.i06 *Accepted:* 2019-11-10