Nonparametric Independence Tests and k-sample Tests for Large Sample Sizes Using Package HHG

by Barak Brill, Yair Heller, and Ruth Heller

Abstract Nonparametric tests of independence and k-sample tests are ubiquitous in modern applications, but they are typically computationally expensive. We present a family of nonparametric tests that are computationally efficient and powerful for detecting any type of dependence between a pair of univariate random variables. The computational complexity of the suggested tests is sub-quadratic in sample size, allowing calculation of test statistics for millions of observations. We survey both algorithms and the **HHG** package in which they are implemented, with usage examples showing the implementation of the proposed tests for both the independence case and the k-sample problem. The tests are compared to existing nonparametric tests via several simulation studies comparing both runtime and power. Special focus is given to the design of data structures used in implementation of the tests. These data structures can be useful for developers of nonparametric distribution-free tests.

Introduction

A common question that arises in the analysis of data is whether two random variables, X and Y, are independent. The null hypothesis is

$$H_0 : F_{XY}(x, y) = F_X(x)F_Y(y) \quad \forall \quad x, y \tag{1}$$

where F_X and F_Y are the marginal cumulative distribution functions of X and Y, and F_{XY} is the joint cumulative distribution function. The case where Y is categorical and X is continuous is the k-sample problem. An omnibus consistent test will reject the null hypothesis in (1) for any dependence between X and Y, with probability increasing to one as the sample size tends to infinity.

In recent years, there has been great interest in developing tests of independence that are able to identify complex dependencies based on N independent observations from F_{XY}. For univariate random variables, the first omnibus consistent test was based on summation of a score over all N 2×2 partitions of the sample space where every data point serves as a partition point (Hoeffding, 1948). This test is available via the function `hoeffd` from package **Hmisc** (Harrell Jr et al., 2018). Another classic approach is based on the measure of mutual information following partitioning of the data into a 2-dimensional grid (Paninski, 2003). This approach is taken in the R packages **minet** (Meyer et al., 2008, 2017), **infotheo** (Meyer, 2014), and **entropy** (Hausser and Strimmer, 2009, 2014) with various extensions to the partitioning schemes used.

Recently, several nonparametric omnibus consistent tests have been suggested that have computational complexity at least quadratic in sample size. Reshef et al. (2011), with CRAN package **minerva** (Albanese et al., 2013; Filosi et al., 2017), suggested MIC, which is based on the maximum of penalized estimated mutual information partitions. Gretton et al. (2008), with CRAN package **dHSIC** (Pfister et al., 2018; Pfister and Peters, 2017), suggested HSIC, which is a kernel test based on the empirical estimate of the Hilbert Schmidt norm of the cross-covariance operator. Székely et al. (2007, 2009), with CRAN package **energy** (Rizzo and Szekely, 2017), suggested dCov, which is based on the joint empirical characteristic function. Both kernel and characteristic function methods may be implemented in a scenario where X or Y are multivariate. Heller et al. (2016), with CRAN package **HHG**, suggested tests which aggregate by maximization or by summation the likelihood ratio test (LRT) scores over all possible partitions of data, with m partition points for each variable. The suggested tests aggregate over all partitions of the data of size $m \times m$ (i.e., having $m \times m$ cells) for a range of m values.

For the k-sample problem, Székely and Rizzo (2004) suggested a test based on joint empirical characteristic function, which they implemented in package **energy** as well. Gretton et al. (2012a) suggested a family of consistent two sample tests based on kernels. The function `kmmd` from package **kernlab** (Zeileis et al., 2004) implements this family of tests for several kernel choices. Jiang et al. (2015), with CRAN package **dslice**, suggested the dynamic slicing test statistic, which aggregates by maximization the penalized LRT score with a penalty for fine partitions. Heller et al. (2016), with CRAN package **HHG**, suggested tests which aggregate by maximization or by summation the LRT scores over all possible partitions of data.

The potential advantage in power as sample size increases in the state-of-the-art tests listed above

is hindered by the computational cost. In practice, many of the state-of-the-art tests cannot be applied when sample sizes are in the thousands. The computational problem is compounded when multiple tests are to be carried out in the same analysis, e.g., when the aim is to detect all pairwise associations among the variables in a dataset.

Modifications to some of the tests listed above can be used for large sample sizes. For the HSIC test statistic, it was suggested to compute the quadratic time HSIC test statistic for subsets of the data, and then aggregate the HSIC test statistics towards the test statistic for the null hypothesis in (1) (Gretton et al., 2012a,b). Two approximate HSIC statistics, the Nyström HSIC test statistic and the random fourier feature HSIC, have been shown to have reduced computational complexity while enjoying power comparable to the original HSIC statistic (Zhang et al., 2018). Other computationally efficient ways to compute HSIC were suggested in Jitkrittum et al. (2016b,a). The three computationally efficient adaptations of the HSIC test (Nyström, RFF, FSIC) have an intrinsic trade-off between power and computational complexity given by a resolution parameter of the method. The user is able to 'pay in runtime' for more power. A computationally efficient algorithm for computing the univariate dCov test statistic in $O(Nlog(N))$ time was developed in Huo and Székely (2016). For the suggested test in Gretton et al. (2012a), computationally efficient modifications have been considered in Zhao and Meng (2015) and Chwialkowski et al. (2015). A computationally efficient algorithm for computing the univariate energy test statistic of Székely and Rizzo (2004) in $O(Nlog(N))$ time was developed in Huang and Huo (2017). Jiang et al. (2015) suggested considering only a subset of partition locations for large sample sizes for their dynamic slicing test. In our present work, we suggest a similar modification for the tests in Heller et al. (2016).

This paper describes two main contributions. The first is to provide a method and software for discovering dependence that has reasonable computational time for any sample size. Specifically, we modify the algorithms for the tests of Heller et al. (2016) , which were shown to have good power in complex settings, to aggregate only a representative subset of all partitions of the data, thus achieving a computational complexity which is sub-quadratic in sample size. The suggested tests have power competitive with state-of-the-art methods, but can be computed at a fraction of the time. Second, we extend the algorithms for the tests of Heller et al. (2016) to allow partitions with a different number of partition points on each axis. LRT scores of all partitions of size $m \times l$ of the sample space are aggregated where m and l are the number of partition points of the X and Y variables, respectively. This generalization does not increase the computational complexity, yet it can result in better power for alternatives where the optimal number of partition points in each axis is different.

The paper is organized as follows. We introduce the atom based $MinP$ statistic and detail our novel contributions in the following two sections. Then, in "Usage examples", we present the work flow of the package (function calls and outputs). In "k-sample tests" we present the computationally efficient tests for the K-Sample problem and their work flow. In "Simulation" we compare the novel tests to other state-of-the-art tests in terms of power and runtime. Finally, in "Discussion" we provide some final remarks. Other tests available in **HHG** are detailed in Appendix B.

The atom based test statistics

Suppose we have N sample points, where each sample is a pair (x, y). We split the plane into m parts along the X axis and into l parts along the Y axis. (Note that for now, m and l are fixed; later we will show how we choose the best m and l automatically so that the user will not need to fix them.) We consider the set of all $m \times l$ partitions of the data, where a split point is possible only once every A ordered observations in each variable. The indivisible blocks of observations are called atoms. We assume for simplicity that the number of atoms N_A is an integer multiple of A, $N_A = N/A$. Figure 1 shows an example partition of the sample space based on atoms.

A cell c is defined by four integers, marking its left, right, top and bottom boundaries on an equidistant grid of $N_A \times N_A$ points. A cell with all four boundaries in the interior of the grid will be considered a 'center' cell (cell type 1). A cell with either its top or bottom boundary at the edge of the grid will be considered a 'top' or 'bottom' cell, respectively (cell type 2). A cell with either its left or right boundary at the edge of the grid will be considered a 'left' or 'right' cell, respectively (cell type 3). A set which has two boundaries at the edge of the grid (e.g., 'left' and 'top') will be called a 'corner' cell (cell type 4). Let O_c denote the number of samples observed in a cell, and E_c the expected number of samples when H_0 is true. For a cell of width w atoms and height h atoms, $E_c = \frac{w}{N_A}\frac{h}{N_A}N = whA^2/N$. Let $\mathcal{C}$ be the set of all cells and $\mathcal{C}(w, h, t)$ the set of all cells of size $w \times h$ atoms and type t, where $t \in \{1, 2, 3, 4\}$. We define the function $n(t, w, h, m, l)$, returning the number of

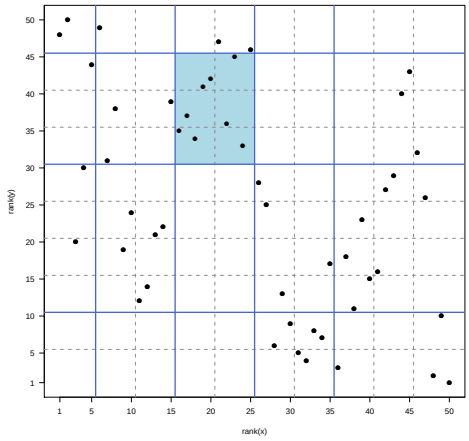

Figure 1: A visualization of a single partition of the atoms plane. The atom size is $A = 5$, so only partitions on the dashed black grid lines are allowed at boundary values $5.5, 10.5, 15.5, \ldots$. For $m = 5$ and $l = 4$, a single 5×4 partition is depicted with blue line boundaries. For $N = 50$ sample points (black dots), we have $N_A = 10$ atoms dividing each axis. The cell coloured in blue, which has width $w = 2$ atoms and height $h = 3$ atoms, has $O_c = 8$ sample points, whereas only $E_c = 2 \times 3 \times 5^2/50 = 3$ are expected under H_0.

$m \times l$ partitions a cell of type t and size w, h participates in:

$$n(t,w,h,m,l) = \begin{cases} \binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3} & \text{t=1 (center cell)} \\ \binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-1}{l-2} & \text{t=2 (top/bottom cell)} \\ \binom{N_A-w-1}{m-2} \cdot \binom{N_A-h-2}{l-3} & \text{t=3 (left/right cell)} \\ \binom{N_A-w-2}{m-2} \cdot \binom{N_A-h-1}{l-2} & \text{t=4 (corner cell)} \end{cases} \tag{2}$$

Let $\tilde{\Gamma}^{m\times l}$ be the set of all $m \times l$ partitions of the plane, where a split point is possible only between atoms. For a given $m \times l$ partition size, the aggregated by sum test statistic is

$$\begin{aligned} S^{m\times l} &= \sum_{\Gamma\in\tilde{\Gamma}^{m\times l}} \sum_{c\in\Gamma} O_c \log\left(O_c/E_c\right) = \sum_{c\in\mathcal{C}} \sum_{\Gamma\in\tilde{\Gamma}^{m\times l}} I\left(c\in\Gamma\right) O_c \log\left(O_c/E_c\right) \\ &= \sum_{t=1}^{4} \sum_{w=1}^{(N_A+1-m)} \sum_{h=1}^{(N_A+1-l)} \sum_{c\in\mathcal{C}(w,h,t)} O_c \log\left(O_c/E_c\right) n\left(t,w,h,m,l\right), \end{aligned} \tag{3}$$

where $I(\cdot)$ is the indicator function. The last equality in (3) demonstrates that for computing $S^{m\times l}$, we can iterate over cells instead of partitions, thus achieving a computational complexity of $\mathcal{O}\left({N_A}^4 + NlogN\right)$, even though the number of possible partitions is $|\tilde{\Gamma}^{m\times l}| = \binom{N_A-1}{m-1} \cdot \binom{N_A-1}{l-1}$.

Since the optimal partition size $m \times l$ is unknown, we propose taking the minimum p-value over the plausible range of partition sizes:

$$MinP = \min_{2\le m,l\le m.max} p_{m\times l}, \tag{4}$$

where $p_{m\times l}$ is the p-value of the test statistic $S^{m\times l}$.

In Appendix A we present the full pseudo-code for the algorithm, including the case when N is not a multiple of A. The pseudo-code also shows how $\{S^{m\times l} : m = 2, \ldots, m.max, l = 2, \ldots, m.max\}$ is computed at the same computational complexity as a single $S^{m\times l}$. The atom based test in (4) is consistent as long as $N_A \to \infty$ and $m.max^2/N \to 0$ (for a proof see Appendix C in Brill, 2016).

The null distribution of $MinP$ and $p_{m\times l}$

In this section, we show how to tabulate the null distribution of our proposed statistic $MinP$. This tabulation requires tabulating the null distribution of $S^{m\times l}$. Fortunately, the test statistics in (3)-(4) are based on the ranked observations and therefore are distribution free. Consequently, the null distributions of $\{S_{m\times l} : 2 \le m, l \le m.max\}$ can be tabulated off-line (prior to seeing the data) in order

to evaluate the p-value of any test statistic that combines $\{p_{m\times l} : 2 \leq m, l \leq m.max\}$.

The tabulation of the null distribution of $S^{m\times l}$ and the $MinP$ test statistics is described in the schematic diagram in Figure 2. The structure of generated null distributions and stored tabulations of null distributions differs. While one generates the vector of $S^{m\times l}$ statistics from a single sample, the package data structure for a null table is constructed by sorted arrays of the marginal distributions. Given a null table of size B repetitions, one can compute all marginal p-values for $S^{2\times 2}, S^{2\times 3}, S^{3\times 2}, \ldots, S^{m.max\times m.max}$ in $O(m.max^2 log(B))$ time once each of the $m \times l$ statistics is computed from data. Then the $MinP$ statistic is simply the minimum of all these p-values. An additional $O(log(B))$ search is required for computing the true p-value of the achieved $MinP$ test statistic. Given the null table, calculating the $MinP$ statistic takes altogether $O(N \log N + N_A^4 + m.max^2 \log B + \log B)$, and since $m.max \leq N_A$, this is at most $O(N \log N + N_A^4 + N_A^2 \log B)$. Since typically $\log B < N_A^2$, the complexity is typically $O(N \log N + N_A^4)$.

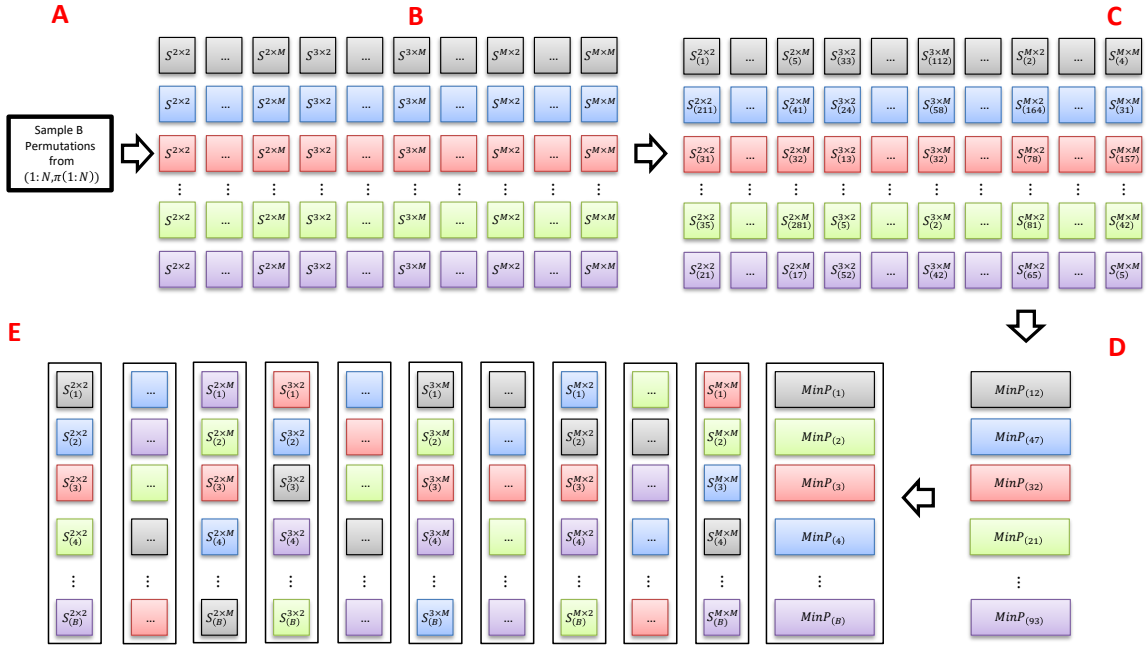

Figure 2: Schematic for the computation of the p-value for the $MinP$ statistic. In step A, sample N pairs without replacement from $\{1,\ldots,N\} \times \{1,\ldots,N\}$, B times. In step B, compute all test statistics for each sample of N pairs, color coded so that we have B rows and $(M-1)^2$ columns of test statistics. In step C, compute the within column rank for each test statistic: the rank is in the subscript for each test statistic, and its p-value is solely determined by the rank and B. In step D, compute $MinP$ for each sample (row) and its rank in the subscript. In step E, each sorted column along with the sorted $MinP$ column is stored individually for fast access and computation of p-values.

In practice, one does not need to maintain all marginal null distributions at a fixed resolution. Only the lower p-values (high $S^{m\times l}$ scores) are used for rejections. Thus, when one simulates a large null table such as $B = 10^6$, marginal ECDFs can be maintained at 0.001 increments of the cumulative probability distribution function for p-values bigger than some parameter for compression α' (e.g., $\alpha' = 0.05$) and at maximum resolution for lower p-values. Using the above parameters as the two different resolutions and α', a null table of 10^6 values is compressed to just over $5 \cdot 10^4$ values. This data structure makes p-value computation via the null table simple and efficient, with null tables sizes being maintainable even for large values of B.

In addition, this data structure is utilized as a combination method for statistics with a nominal false positive rate of α. Importantly, one can utilize this efficient data structure with any set of statistics and a general combination score which takes into account only marginal p-values. For example, Heller et al. (2016) propose another type of combination score for their tests, of the form $-\sum_{m=2}^{m.max} log\,(p_{m\times m})$. The combination score makes use of this data structure as well.

Usage examples

Function calls and input arguments

The test procedure utilizes two function calls. The first function call carries out the tabulation of the null distributions for both the $S^{m\times l}$ statistics and the *MinP* combination statistic. The second function call computes the *MinP* test statistic and its p-value, given the look-up tables for the distributions of $S^{m\times l}$ and *MinP* (see Table 1).

The arguments for the null distribution tabulation are the sample size, the maximal partition size considered, and the number of atoms. These are denoted by the parameters `n`, `mmax`, and `nr.atoms`, respectively. The default parameters for the test are `mmax = min(10,n)` and `nr.atoms = min(40,n)`. In Section "Simulations", we discuss these defaults in terms of power, runtime, and the trade-off between the two. Other parameters include the type of combination score used (*MinP* or Fisher) and the type of data partition used ($m \times m$ or $m \times l$).

The arguments for computing the MinP test statistic via `Fast.independence.test` are two vectors of size `n` for the joint observations of (X, Y), and a null table object produced by the function `Fast.independence.test.nulltable`. All test parameters are kept in the null table object.

Table 1: The novel atom-based tests in the HHG package.

Function Name	Description
`Fast.independence.test.nulltable`	Function creates null table objects for the atoms based omnibus distribution-free test of independence between two univariate random variables. Input arguments are the number of atoms, $m.max$, and the type of partitioning (all $m \times l$ or all $m \times m$ partitions).
`Fast.independence.test`	Performs the atoms based distribution-free test of independence of two univariate random variables, which is computationally efficient for large data sets. Input arguments are two numeric vectors of data, and optionally a null table object (if a null table has not been precomputed, one is generated on the fly).

Output description

The output of the function `Fast.independence.test.nulltable` is a null table data structure. This data structure contains the sorted marginal distributions of $S^{m\times l}$ statistics, the sorted distribution of the *MinP* test score, and the test parameters as in Figure 2.

The output of `Fast.independence.test` is an object of class `"UnivariateStatistic"` containing the results of both the marginal $S^{m\times l}$ tests performed on the data, along with the results of the *MinP* test. The fields `m.stats` and `pvalues.of.single.m` contain the test statistics for the tests of fixed partitions size and their respective P-values, given by (3). The fields `MinP` and `MinP.pvalue` contain the data adaptive test statistic and its P-value, given by (4).

Example code

We begin by computing the a look-up tables of the null distributions. This is done by calling the first function, with the sample size as a parameter:

```
# compute null table, using default m.max,
# number of atoms and number of permutations under the null.

nt = Fast.independence.test.nulltable(n)
```

The number of atoms for the procedure and the maximal partition size are set to their default values. The object `nt` now holds a look-up table which is specific for the selected test parameters and sample size. We will use this object to carry out the test using `Fast.independence.test`:

```
# carry out test, parameters set by null table passed.
res = Fast.independence.test(x,y,nt)

# print results and P-value. P-value given under entry 'MinP.pvalue'.
res
```

The last line prints the output for the test. The output begins by describing the test parameters: test statistic chosen, partition sizes considered, and number of atoms:

```
HHG univariate combined independence statistic
Statistics type combined:
sum of ADP-EQP-ML on Likelihood Ratio scores.

Single m statistics are the sum of scores over All Derived Partitions (ADP) of the data.
Statistics are normalized by the number of possible partitions and sample size.

Minimum partition size: 2  Maximum partition size: 10
Sample size: 1200
Equipartition nr.atoms: 40
```

The output printed shows the $S^{m\times l}$ test statistics along with their marginal p-values:

```
Single m (partition size) statistics:
       m=2       m=3      m=4      m=5      m=6      m=7      m=8      m=9      m=10
l=2   0.000548  0.00117  0.00180  0.00244  0.00308  0.00369  0.00429  0.00486  0.00541
l=3   0.001165  0.00244  0.00374  0.00505  0.00633  0.00759  0.00881  0.00998  0.01111
...     ...       ...          ...      ...      ...      ...      ...      ...      ...
l=10  0.004584  0.00950  0.01453  0.01962  0.02470  0.02973  0.03468  0.03955  0.04434

Single m (partition size) pvalues:
       m=2     m=3     m=4     m=5     m=6      m=7      m=8      m=9      m=10
l=2   0.199  0.1294  0.0995  0.0597  0.05473  0.04478  0.04478  0.03483  0.03483
l=3   0.134  0.0647  0.0398  0.0299  0.02488  0.01493  0.01493  0.01493  0.01493
...    ...    ...     ...     ...      ...      ...      ...      ...      ...
l=10  0.174  0.0846  0.0448  0.0249  0.01493  0.00995  0.00995  0.00995  0.00995
```

Finally, the output shows the MinP test statistic score along with it's p-value, which is the p-value for the test. The selected partition size attaining the smallest p-value is also shown:

```
MinP Statistic - Test statistic is minimum of above single m pvalues:
0.01
Partition size with minimum p-value: 6X6
p-value for MinP test:0.01
```

As stated above, the proposed method is distribution free. As such, the look-up table generated can be used with any data of the same size, as we show in the next example.

```
# generate data of size n:
x.2 = rnorm(n)
y.2 = x.2 + rnorm(n)

# carry out test using exactly the same null table as before.
res2 = Fast.independence.test(x.2,y.2,nt)
```

The main advantage of distribution free tests is that while standard permutation based tests performed on M null hypotheses with B permutations (in each test) require $M \times B$ independent calculations of the test statistics, distribution free tests require only M test statistics and B permutations, $M + B$ in total. This condition makes them ideal for scenarios where a large number of hypotheses are examined simultaneously.

For large values of N_A and table size B, the computation of the look-up table can still be cumbersome. The package vignette shows how this process can be parallelized.

k-sample tests

A special case of the independence problem is the k-sample problem: independence testing for continuous X and categorical Y, where Y has K different categories, $1, \ldots, K$. The null hypothesis can

be formulated as:

$$H_0 : F_X(x|y = 1) = F_X(x|y = 2) = \ldots = F_X(x|y = k), \forall x \tag{5}$$

We modify the *MinP* test statistic in Heller et al. (2016) to consider only partitions at the atom level. For example, instead of considering all possible partitions, we consider only partition points that are found on an equidistant grid of N_A points. Let p_m be the p-value of the test statistic that aggregates by summation or maximization all the LRT scores with partitions of size m of the real line, where a split point is possible only every N_A points. The atom based test statistic is

$$MinP = \min_{2 \leq m \leq m.max} (p_m) . \tag{6}$$

The code snippets below show how to analyze an example. The two vectors `X` and `Y` are of size `1000`. The entries in `Y` are the group labels: 500 zeros and 500 ones.

```
# generate null table
nt = hhg.univariate.ks.nulltable(c(500,500), #group structure
                        variant = 'KSample-Equipartition', #computationally
                        # efficient variant of the K-sample test.
                        mmax = 10, #m.max parameter
                        nr.atoms = 30, #number of atoms
                        nr.replicates = 10^4) #number of replicates

# run test
res = hhg.univariate.ks.combined.test(X,Y,nt)

# Shows Average LRT scores: AVG LRT score of
# m = 2,...,10 cell tables
res
```

As with independence tests in which many k-sample tests with the same sample sizes are carried out, the same null table can be used. We demonstrate this in the next example:

```
# generate sample with the same group structure -
# normal deviates with a shift between groups.
X2 = rnorm(length(Y),0,1)+1*Y

# perform test using the same null table.
res2 = hhg.univariate.ks.combined.test(X2,Y,nt)

# view results
res2
```

Simulations

We used simulations to assess the performance of the atom based tests in terms of both run time and power. Full source code for reproducing the simulation results, graphs and usage examples is found in the supplementary material, and at the GitHub repository (https://barakbri.github.io/HHG_large_sample_framework/).

All run times were measured using the CRAN package **rbenchmark** (Kusnierczyk, 2012). Run times were measured separately from power estimation, as simulation for power estimation has been parallelized via the **doRNG** package (Gaujoux, 2017). All run time experiments were done serially, without parallelization.

The test of independence

In order to assess the power of the *MinP* test statistics along with the actual run time required, we present four different scenarios of dependence between univariate random variables. Figure 3 shows the bivariate relationship along with a representative noisy sample: two monotone relationships (left panels), and two non-monotone relationships (right panels).

The presented methods were run with $N = 2500$, $N_A = 5, 10, 15, 30, 45, 60$, and with summation over all $m \times l$ or $m \times m$ partitions. This allows one to assess the affect of N_A on power and run time, along with the possible affect of summation over tables with a different number of partition points on

the two axes. Reducing the number of atoms N_A, allows one to estimate the breakdown of the method in terms of power.

The parameter $m.max$ was set to the package default ($m.max = 10$), except for $N_A = 5$ where $m.max$ was constrained by the number of atoms to 5. Heller et al. (2016) and Brill (2016) show by various simulation studies that the power of the test is quite robust to the selection of $m.max$ since it selects the optimal m, l adaptively; the test considers all partitions of sizes 2×2 to $m.max \times m.max$ and selects the best partition in the sense that its p-value is minimal.

Methods compared to in the simulation study are dCOV from CRAN package **energy**, MIC from CRAN package **minerva**, and HSIC from CRAN package **dHSIC**. We note that typically faster competitive methods are somewhat degenerate variations on these methods and would have much lower power than the originals.

Figure 4 shows the trade-off between runtime and power. We see that as the number of atoms increase, the run time increases but power is almost the same for 30 atoms or more. We set the default value for the number of atoms in the functions given in Table 1 to be the minimum between sample size and 40, promising the user a result in reasonable time regardless of sample size. Even for a small number of atoms such as $N_A = 10$, the MinP test power is similar to the MinP test with a high number of atoms. For the smallest number of atoms considered, $N_A = 5$, power may drop for complex signals which require fine partitions of the data. For the monotone settings considered, the method maintains competitive power also for $N_A = 5$.

Figure 5 shows the power as a function of sample size. The power of the test for the monotone settings is highest for dCov with the atoms based test a close second, similar to the best one achieved by the competitors. For the Circles setting, power for the $m \times m$ and $m \times l$ variant is similar. The setting is symmetric in practice, and the $m \times m$ variant enjoys higher power since it needs to account for fewer possible selections of partition size under the $MinP$ test statistic. Nevertheless, the loss of power is small when considering all $m \times l$ partitions of the data. For the Sine setting, power differs greatly between $m \times l$ and $m \times m$ variants. The setting is not symmetric in X and Y. One can see that the optimal partition of the plane to capture the dependence requires few partitions of the Y axis but many partitions of the X axis. See Brill (2016) for thorough simulations of different types of bivariate relationships: in the non-symmetric settings, considering $m \times l$ partitions of the data leads to substantial power gain; in symmetric settings, considering $m \times l$ partitions of the data leads to little power loss over considering only $m \times m$ partitions.

This simulation study demonstrated that the presented tests have a power advantage over competitors in settings where the underlying dependence is complex, i.e., settings where in multiple regions the joint density and the product of the marginal densities differ. For those settings, a fine partition of the sample space is optimal. Competing tests have tuning parameters, and the choice of tuning parameter can affect the power of the test. Specifically, tuning parameters in competitor methods include the degree of the L_p norm in dCOV, the kernel bandwidth in dHSIC, and the maximum partition size considered in MIC. The low power achieved by alternative methods in the 'Circles' and 'Sine' settings could be partially attributed to the use of the package default setting for the tuning parameters. The MinP procedure does not have a tuning parameter that can materially affect its performance, since the single best partition size is chosen in a data adaptive manner. For many other scenarios demonstrating this, see Heller et al. (2016) and Brill (2016).

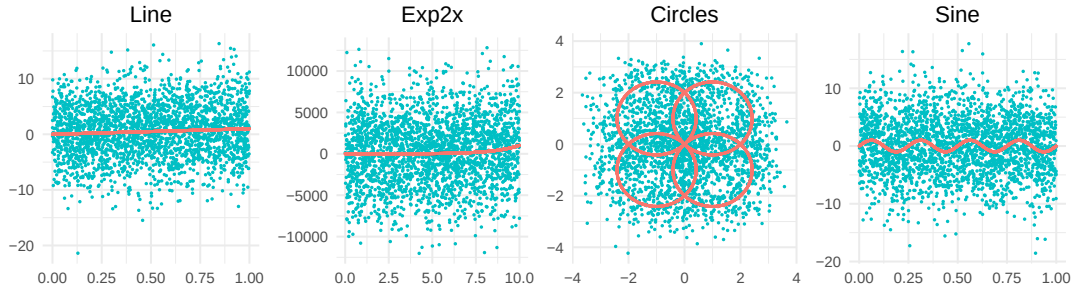

Figure 3: Four bivariate relationships (in red), along with a sample of $N = 2500$ observations in blue.

Figure 6 shows the run-time as a function of sample size and atom size. In the left panel, the total time to carry out a test with 1000 permutations is computed for 4 methods: $MinP$ (including null table construction) with $S^{m \times l}$ test statistic ($N_A = 45$), dCov, HSIC, and MIC (including null table construction). We see that the $O\left(N_A^4\right)$ portion of the computation time of $MinP$ takes the largest portion of the computation, being greater than the $O\left(Nlog\left(N\right)\right)$ part. As such, computation time is constant for all N values considered. For other tests considered, computation time might be shorter for smaller samples, but the computational complexity is quadratic (or above) and hence more

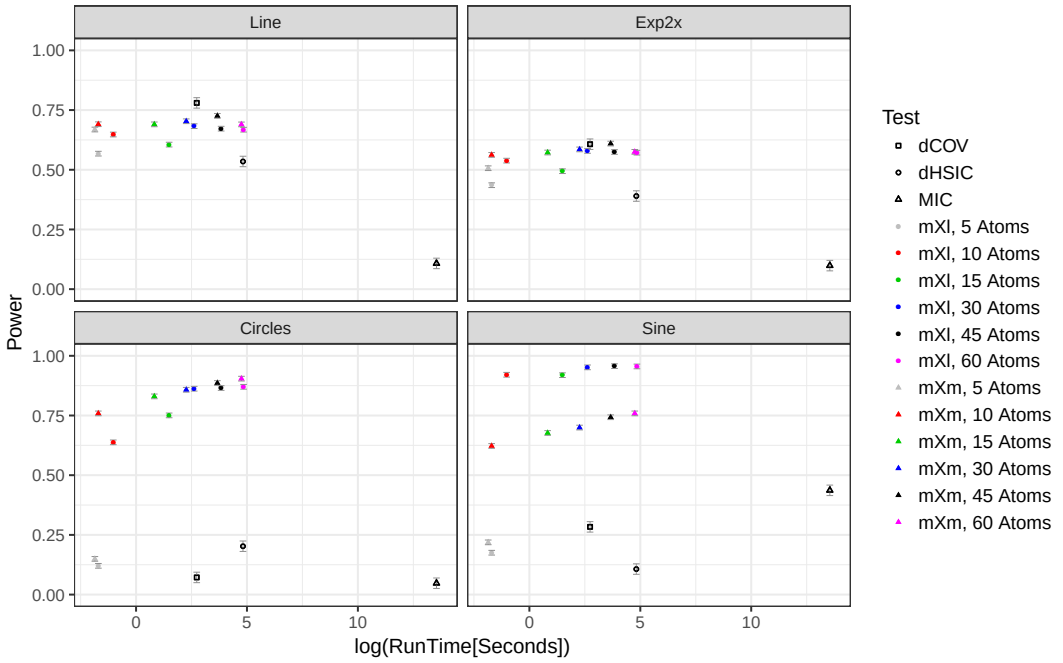

Figure 4: Power and logarithm of run time for $N = 2500$. Run times for each method were computed for a full procedure, including tabulation of the null hypothesis and parameter selection. Specifically, for **dCov** and **HSIC**, the number of permutations was a single run of the test function with 1000 permutations (as computation of kernel distance matrices is done only once for the original data); for **MinP**, runtime consists of 1000 computations for the null table and the test statistic (computation of statistics under the null hypothesis, efficient tabulation of marginal distributions, and $MinP$, computation of $MinP$ for the tested dataset, which was then used for computing the P-value); for **MIC**, run times consist of 1001 computations of the test statistic for the dataset tested and distribution under the null. For a pre-computed null table of size $N = 2500$, running times for MIC and $MinP$ would be one thousandth of the time shown above, since these two tests are distribution free. Power computation is based on 10000 datasets for the MinP procedure and 2000 datasets for alternative tests. Errors bars show 95% confidence intervals.

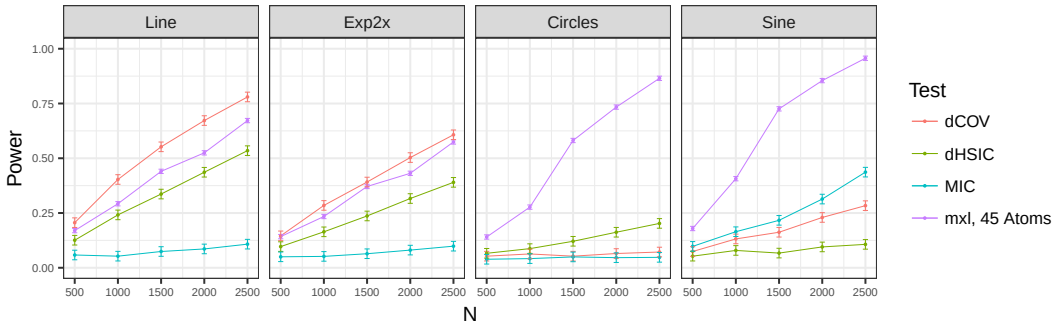

Figure 5: Power versus sample size, in each of the scenarios of Figure 3. Based on 10000 datasets for the MinP procedure, and 2000 datasets for alternative tests. Error bars show 95% confidence intervals.

computationally demanding than *MinP* for larger sample sizes. The right panel presents the run time versus number of atoms for $N = 300$ and $m.max = 10, 15$. The relationship is clearly linear on the log scale. The linear fit shows that the computational complexity is indeed to the fourth order in terms of N_A. Overall, the algorithm has a computational complexity which is $\mathcal{O}\left(N \log N + N_A^4\right)$.

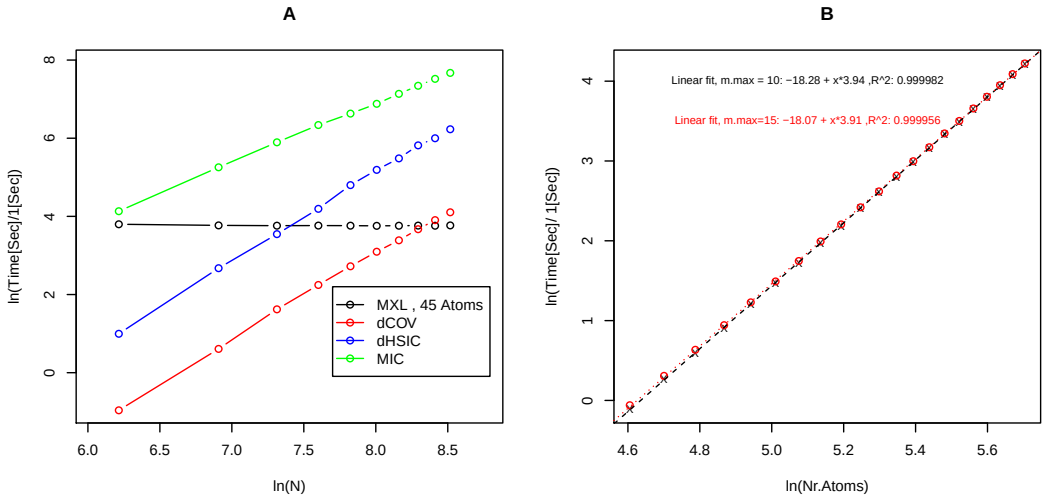

Figure 6: A (Left): Runtime analysis on log scale for different methods for sample sizes $N = 500, 1000, \ldots, 5000$. B (Right): Runtime comparison on log scale, compared with number of atoms. Number of atoms varied between 100 and 300. Computations performed were $S^{m\times l}$ with $mmax = 10$ (black) and $mmax = 15$ (red). Linear fit verifies known computational complexity, $\mathcal{O}\left(N \log N + N_A^4\right)$. Time measurements were performed with 100 and 50 repetitions for left and right panels, respectively.

The k-sample test

In order to assess the power of the *MinP* test statistics, we present a simulation study. Methods compared were the energy two sample test, given by `eqdist.etest` from package **energy**, and the kernel MMD test, given by `kmmd` from **kernlab**.

Let $\sum_{i=1}^{K} p_i N(\mu_i, \sigma_i^2)$ denote the mixture distribution of K Gaussian distributions with means μ_i and variances σ_i^2. We examine the following three settings: the shifted normal, sampling from $N\left(0, 1^2\right)$ and from $N\left(0.075, 1^2\right)$; the 2 component normal mixture, sampling from $0.7N\left(0, 1^2\right) + 0.3N\left(0, 8^2\right)$ and $0.7N\left(0.15, 1^2\right) + 0.3N\left(0, 8^2\right)$; and the 3 component normal mixture, sampling from $f\left(X|Y = 1\right) = \frac{1}{3}N\left(-4, 1\right) + \frac{1}{3}N\left(0, 1\right) + \frac{1}{3}N\left(4, 1\right)$, and $f\left(X|Y = 2\right) = \frac{1}{3}N\left(-4, 0.8^2\right) + \frac{1}{3}N\left(0, 0.8^2\right) + \frac{1}{3}N\left(4, 0.8^2\right)$.

Table 2: A power comparison of the *MinP* test with a different number of atoms, the energy test, and the MMD test. For the three data generations, the power is given in rows 1-3, as well as the run time in row 4. Datasets are samples with two equal groups of size 2500 each to a total of $N = 5000$. For each setting, we simulate 10000 datasets for the MinP procedure and 2000 datasets for alternative tests, with 1000 permutations for each test. Run times are measured over 100 repetitions.

	10 Atoms	25 Atoms	50 Atoms	ENERGY	KMMD
Normal, Shift	0.65	0.70	0.70	0.72	0.53
Mixture, 2 Components	0.72	0.75	0.74	0.21	0.36
Mixture, 3 Components	0.86	0.94	0.94	0.09	0.07
Run time (seconds)	2.96	3.09	3.23	5.22	146.51

Table 2 provides the power result for competitor tests, along with the *MinP* test statistic for $N_A = 10, 25, 50$. The *MinP* test has excellent power in comparison with energy and MMD, and the run time is lower. For additional simulations that include many additional settings, see Brill (2016). In general, the proposed k-sample test performs best when the difference between the distributions is complex and specifically when the density plots of the two distributions intersect multiple times, see Heller et al. (2016) for details.

Discussion

We presented computationally fast and powerful novel tests for detecting complex dependencies among univariate random variables. These tests can be used in the framework suggested in Heller and Heller (2016) in order to construct computationally-fast, distribution-free multivariate tests for powerful identification of complex multivariate relationships. Briefly, one may reduce tests of independence between multivariate X and Y to univariate ones by several methods, such as choosing a reference point in X and Y and testing whether distances between observations and reference points are associated in X and Y, or by choosing a direction and projecting X and Y on it. Heller and Heller (2016) discuss methods of aggregation over several reference points. If the univariate tests utilized are computationally efficient for large sample sizes and consistent, the resulting multivariate tests will also be consistent and computationally efficient.

Using our approach for obtaining a look-up table for $MinP$, the null distribution of any test statistic that combines individual p-values from distribution-free tests can be efficiently tabulated. Steps A to E depicted in Figure 2 can be performed with $S^{m\times l}$ columns replaced by other rank based test statistics $T_1, \ldots, T_M$, and the $MinP$ can be replaced by any p-value combining function $f(p_1, \ldots, p_M)$. For example, package `coin` contains various rank based tests to detect shift or scale differences across distributions. The null distribution tabulation we presented can be useful for constructing a look-up table for a distribution-free combined test for shift and scale.

Appendix A

Let N be the number of observations, and N_A be the number of atoms such that the size of an atom is given by an integer $A = N/N_A$. The locations of splits between atoms are thus given by $T_i = i \cdot A$ for $i \in \{0, 1, 2, \ldots, N_A\}$. If N is not a complete multiple of N_A, we define the locations of splits between atoms to be $T_i = \lfloor i \cdot N/N_A \rfloor$.

Step I (compute ECDF): The empirical CDF can be computed in $O(N_A^2 + N)$ time for all possible split locations: $\hat{F}(T_i, T_j), 0 \le i, j, \leqq N_A$. First, execute Algorithm 1 to compute the matrix $A_{r,s}$.

Algorithm 1 Compute ECDF at all inter-atoms split points

```
procedure COMPUTEECDF
    Initialize A_{i,j} ← 0, for 0 ≤ i, j, ≤ N_A
    Initialize B_{i,j} ← 0, for 0 ≤ i, j, ≤ N_A
    for i = 1 to N do
        AtomRank_x ← ⌈N_A/N · rank(x_i)⌉
        AtomRank_y ← ⌈N_A/N · rank(y_i)⌉
        B_{AtomRank_x, AtomRank_y} ← B_{AtomRank_x, AtomRank_y} + 1
    for s = 0 to N_A do
        for r = 0 to N_A do
            A_{r,s} ← B_{r,s-1} + B_{r-1,s} − B_{r-1,s-1} + B_{r,s}
            B_{r,s} ← A_{r,s}
```

The empirical cumulative distribution function at the split point given by ranks (T_i, T_j) is given by $\hat{F}(T_i, T_j) = A_{i,j}/N$. For a cell of borders $[T_{i1}, T_{i2}] \times [T_{j1}, T_{j2}]$, the expected number of observations is given by:

$$E(T_{i1}, T_{i2}, T_{j1}, T_{j2}) = (T_{i2} - T_{i1}) \cdot (T_{j2} - T_{j1})/N \tag{7}$$

Once the empirical CDF has been tabulated, the observed counts of the cell can be calculated in constant time:

$$O(T_{i1}, T_{i2}, T_{j1}, T_{j2}) = A_{i2,j2} - A_{i1,j2} - A_{i2,j1} + A_{i1,j1} \tag{8}$$

Step II (Aggregate LRT scores of all cells of given size): A cell of the sample space $[T_{i1}, T_{i2}] \times [T_{j1}, T_{j2}]$ is given by its boundaries $i1, i2, j1, j2$. We define a cell to be 'top' or 'bottom' cell if $i2 = N$ or $i1 = 0$. A cell is considered 'right' or 'left' if $j2 = N$ or $j1 = 0$. If a cell is in both categories, it is called 'corner'. If a cell is not 'top','bottom','left', or 'right', it is called a 'center' cell. Let $Type(i1, i2, j1, j2)$ be a function taking the borders of a cell and returning 1 for 'center' cells, 2 for 'top' or 'bottom', 3 for 'left' or 'right', and 4 for 'corner' cells.

We define the width of the cell to be $w(i1, i2) = i2 - i1$ and the height of the cell to be $h(j1, j2) = j2 - j1$. For each size and type, we sum up the scores of that size and type.

Algorithm 2 computes the array $S_{t,w,h}$, which stores the sum of scores of the form $O \cdot ln\left(\frac{O}{E}\right)$ for all cells of type t, width w, and height h.

Algorithm 2 Aggregate contribution to statistic by cell type and size

```
1:  procedure AGGREGATECELLSCORES
2:      Initialize S_{t,w,h} ← 0, ∀t ∈ [1,4], w,h ∈ [1, N_A − 1]
3:      for i1 = 0 to N_A − 1 do
4:          for i2 = i1 + 1 to N_A do
5:              w_0 ← w(i2, i1)
6:              for j1 = 0 to N_A − 1 do
7:                  for j2 = j1 + 1 to N_A do
8:                      t ← Type(i1, i2, j1, j2)
9:                      h_0 ← h(j2, j1)
10:                     o ← O(i1, i2, j1, j2)
11:                     e ← E(i1, i2, j1, j2)
12:                     S_{t,w_0,h_0} ← S_{t,w_0,h_0} + o · ln(o/e)
```

Step III (Computing $S^{m\times l}$): Next, we aggregate over S_{t,w_0,h_0}, over all cell sizes and types, to compute $S^{m\times l}$ with $m,l \in [2, m.max]$.

The number of partitions in which a cell is found is given by the number of possible ways to select additional partitioning locations around that cell, denoted by $n(t,w,h,m,l)$ (2). For example, a 'center' cell requires $m-3$ choices of additional X partition points and $l-3$ additional Y partition points to fully define a $m \times l$ partition. Hence a 'center' cell of size w,h is found in $\binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3}$ partitions. Note that the number of tables in which a cell is found does not depend on its edge locations but only on its size and type. Thus, the contribution for $S^{m\times l}$ of all center cells of size $w \times h$ atoms is given by the sum of their $O \cdot ln\left(\frac{O}{E}\right)$ scores, multiplied by $\binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3}$. The differentiation by cell type is needed since a cell bordering the edge of the sample space requires only one partition point, thus allowing the selection of an extra partition point.

The final step is given in algorithm 3. Algorithm 3 computes $S^{m\times l}$ by summing over all values of the array $S_{t,w,h}$ with weights $n(t,w,h,m,l)$.

Algorithm 3 Compute $S^{m\times l}$ for all $m,l \in [2, m.max]$

```
1:  procedure COMPUTESTATISTIC
2:      Initialize S^{m×l} ← 0, ∀m,l ∈ [2, m.max]
3:      for m = 2 to m.max do
4:          for l = 2 to m.max do
5:              for t = 1 to 4 do
6:                  for w = 1 to N_A − 1 do
7:                      for h = 1 to N_A − 1 do
8:                          S^{m×l} ← S^{m×l} + n(t,w,h,m,l) · S_{t,w,h}
```

Appendix B

In Table 3 we list the additional available tests in the HHG package. Specifically, the multivariate tests of Heller et al. (2013) and the univariate test of Heller et al. (2016).

Bibliography

D. Albanese, M. Filosi, R. Visintainer, S. Riccadonna, G. Jurman, and C. Furlanello. minerva and minepy: a c engine for the mine suite and its r, python and matlab wrappers. *Bioinformatics*, 29(3): 407–408, 2013. URL https://doi.org/10.1093/bioinformatics/bts707. [p424]

B. Brill. Scalable non-parametric tests of independence. Master's thesis, Tel Aviv University, 2016. URL http://primage.tau.ac.il/libraries/theses/exeng/free/2899741.pdf. [p426, 431, 433]

Table 3: Nonparametric tests previously available in the **HHG** package for small to moderate sample sizes

Function Name	Description
`hhg.test`	Implements the multivariate test of independence described in Heller et al. (2013), testing for the independence of two random vectors $\vec{X}$ and $\vec{Y}$,of dimensionality p and q, respectively.
`hhg.test.2.sample` `hhg.test.k.sample`	Adaptation of the above procedure to the k-sample problem (equality of distributions). Given a multivariate measurement $\vec{X}$ and a group factor variable $\vec{Y}$, test whether the distributions $\vec{X}\|Y = 1, \vec{X}\|Y = 2, \ldots, \vec{X}\|Y = k$ are equal. This is the nonparametric extension of the MANOVA problem (sensitive not only to shifts in means).
`hhg.univariate.ind.combined.test`	Univariate test of independence for problem (1), as described in Heller et al. (2016).
`hhg.univariate.ks.combined.test`	Test for univariate equality of distributions (i.e., $X\|Y = 1\ X\|Y = 2\ \ldots\ X\|Y = k$), as described in Heller et al. (2016). This is the nonparametric extensions of the ANOVA problem, which is sensitive to any difference in distribution (not only to shifts in means).

K. P. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. Fast two-sample testing with analytic representations of probability measures. In *Advances in Neural Information Processing Systems*, pages 1981–1989, 2015. [p425]

M. Filosi, R. Visintainer, and D. Albanese. *minerva: Maximal Information-Based Nonparametric Exploration for Variable Analysis*, 2017. URL https://CRAN.R-project.org/package=minerva. R package version 1.4.7. [p424]

R. Gaujoux. *doRNG: Generic Reproducible Parallel Backend for 'foreach' Loops*, 2017. URL https://CRAN.R-project.org/package=doRNG. R package version 1.6.6. [p430]

A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A kernel statistical test of independence. In *Advances in neural information processing systems*, pages 585–592, 2008. [p424]

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012a. [p424, 425]

A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1205–1213. Curran Associates, Inc., 2012b. URL http://papers.nips.cc/paper/4727-optimal-kernel-choice-for-large-scale-two-sample-tests.pdf. [p425]

F. E. Harrell Jr, with contributions from Charles Dupont, and many others. *Hmisc: Harrell Miscellaneous*, 2018. URL https://CRAN.R-project.org/package=Hmisc. R package version 4.1-1. [p424]

J. Hausser and K. Strimmer. Entropy inference and the james-stein estimator, with application to nonlinear gene association networks. *Journal of Machine Learning Research*, 10(Jul):1469–1484, 2009. [p424]

J. Hausser and K. Strimmer. *entropy: Estimation of Entropy, Mutual Information and Related Quantities*, 2014. URL https://CRAN.R-project.org/package=entropy. R package version 1.2.1. [p424]

R. Heller and Y. Heller. Multivariate tests of association based on univariate tests. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information*

Processing Systems 29, pages 208–216. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6220-multivariate-tests-of-association-based-on-univariate-tests.pdf. [p434]

R. Heller, Y. Heller, and M. Gorfine. A consistent multivariate test of association based on ranks of distances. *Biometrika*, 100(2):503–510, 2013. URL https://doi.org/10.1093/biomet/ass070. [p435, 436]

R. Heller, Y. Heller, S. Kaufman, B. Brill, and M. Gorfine. Consistent distribution-free k-sample and independence tests for univariate random variables. *Journal of Machine Learning Research*, 17(29): 1–54, 2016. [p424, 425, 427, 430, 431, 433, 435, 436]

W. Hoeffding. A non-parametric test of independence. *The annals of mathematical statistics*, pages 546–557, 1948. URL https://doi.org/10.1214/aoms/1177730150. [p424]

C. Huang and X. Huo. An efficient and distribution-free two-sample test based on energy statistics and random projections. *arXiv preprint arXiv:1707.04602*, 2017. [p425]

X. Huo and G. J. Székely. Fast computing for distance covariance. *Technometrics*, 58(4):435–447, 2016. URL https://doi.org/10.1080/00401706.2015.1054435. [p425]

B. Jiang, C. Ye, and J. S. Liu. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510):642–653, 2015. URL https://doi.org/10.1080/01621459.2014.920257. [p424, 425]

W. Jitkrittum, Z. Szabó, K. Chwialkowski, and A. Gretton. Interpretable distribution features with maximum testing power. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 181–189. Curran Associates, Inc., 2016a. URL http://papers.nips.cc/paper/6148-interpretable-distribution-features-with-maximum-testing-power.pdf. [p425]

W. Jitkrittum, Z. Szabó, and A. Gretton. An adaptive test of independence with analytic kernel embeddings. *arXiv preprint arXiv:1610.04782*, 2016b. [p425]

A. Karatzoglou, A. Smola, and K. Hornik. *kernlab: Kernel-Based Machine Learning Lab*, 2016. URL https://CRAN.R-project.org/package=kernlab. R package version 0.9-25. [p]

B. B. . S. Kaufman, based in part on an earlier implementation by Ruth Heller, and Y. Heller. *HHG: Heller-Heller-Gorfine Tests of Independence and Equality of Distributions*, 2017. URL https://CRAN.R-project.org/package=HHG. R package version 2.2. [p]

W. Kusnierczyk. *rbenchmark: Benchmarking routine for R*, 2012. URL https://CRAN.R-project.org/package=rbenchmark. R package version 1.0.0. [p430]

D. Lopez-Paz, P. Hennig, and B. Schölkopf. The randomized dependence coefficient. In *Advances in neural information processing systems*, pages 1–9, 2013. [p]

P. E. Meyer. *infotheo: Information-Theoretic Measures*, 2014. URL https://CRAN.R-project.org/package=infotheo. R package version 1.2.0. [p424]

P. E. Meyer, F. Lafitte, and G. Bontempi. minet: Ar/bioconductor package for inferring large transcriptional networks using mutual information. *BMC bioinformatics*, 9(1):461, 2008. URL https://doi.org/10.1186/1471-2105-9-461. [p424]

P. E. Meyer, F. Lafitte, and G. Bontempi. *minet: Mutual Information NETworks*, 2017. URL http://minet.meyerp.com. R package version 3.36.0. [p424]

Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct for R*, 2017. URL https://CRAN.R-project.org/package=foreach. R package version 1.4.4. [p]

L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003. URL https://doi.org/10.1162/089976603321780272. [p424]

N. Pfister and J. Peters. *dHSIC: Independence Testing via Hilbert Schmidt Independence Criterion*, 2017. URL https://CRAN.R-project.org/package=dHSIC. R package version 2.0. [p424]

N. Pfister, P. Bühlmann, B. Schölkopf, and J. Peters. Kernel-based tests for joint independence. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):5–31, 2018. URL https://doi.org/10.1111/rssb.12235. [p424]

D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *science*, 334(6062): 1518–1524, 2011. URL https://doi.org/10.1126/science.1205438. [p424]

M. L. Rizzo and G. J. Szekely. *energy: E-Statistics: Multivariate Inference via the Energy of Data*, 2017. URL https://CRAN.R-project.org/package=energy. R package version 1.7-2. [p424]

C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904. URL https://doi.org/10.2307/1412159. [p]

G. J. Székely and M. L. Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5(16.10), 2004. [p424, 425]

G. J. Székely, M. L. Rizzo, N. K. Bakirov, et al. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007. URL https://doi.org/10.1214/009053607000000505. [p424]

G. J. Székely, M. L. Rizzo, et al. Brownian distance covariance. *The annals of applied statistics*, 3(4): 1236–1265, 2009. URL https://doi.org/10.1214/09-aoas312. [p424]

H. Wickham and W. Chang. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL https://CRAN.R-project.org/package=ggplot2. R package version 2.2.1. [p]

C. Ye, B. Jiang, X. Zhang, and J. S. Liu. *dslice*: an R package for nonparametric testing of associations with application in qtl and gene set analysis. *Bioinformatics*, 31(11):1842–1844, 2015. URL https://doi.org/10.1093/bioinformatics/btv021. [p]

A. Zeileis, K. Hornik, A. Smola, and A. Karatzoglou. kernlab-an s4 package for kernel methods in r. *Journal of statistical software*, 11(9):1–20, 2004. URL https://doi.org/10.18637/jss.v011.i09. [p424]

Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 28(1):113–130, 2018. URL https://doi.org/10.1007/s11222-016-9721-7. [p425]

J. Zhao and D. Meng. Fastmmd: Ensemble of circular discrepancy for efficient two-sample test. *Neural computation*, 27(6):1345–1372, 2015. URL https://doi.org/10.1162/neco_a_00732. [p425]

Barak Brill
Department of Statistics and Operations Research, Tel-Aviv University
Tel-Aviv
Israel
barakbri@mail.tau.ac.il

Yair Heller
Tel-Aviv
Israel
heller.yair@gmail.com

Ruth Heller
Department of Statistics and Operations Research, Tel-Aviv University
Tel-Aviv
Israel
ruheller@gmail.com