SoftwareX 22 (2023) 101326

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

dcor: Distance correlation and energy statistics in Python

Carlos Ramos-Carreño [a,*], José L. Torrecilla [b]

[a] Department of Computer Science, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain
[b] Department of Mathematics, Facultad de Ciencias, Universidad Autónoma de Madrid, Madrid, Spain

ARTICLE INFO

Article history:
Received 2 November 2022
Received in revised form 20 January 2023
Accepted 25 January 2023

Keywords:
Energy statistics
Energy distance
Distance correlation
Hypothesis testing
Python

ABSTRACT

This article presents **dcor**, an open-source Python package dedicated to distance correlation and other statistics related to energy distance. These energy statistics include distances between distributions and the associated tests for homogeneity and independence. Some of the most efficient algorithms for the estimation of these measures have been implemented relying on optimization techniques such as vectorization, compilation, and parallelization. The performance of these estimators is evaluated by comparison with alternative implementations in other packages. The package is also designed to be compatible with the packages conforming the scientific Python ecosystem. With that purpose in mind, **dcor** is an early adopter of the Python array API standard.

© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Code metadata

Current code version	v0.6
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00356
Permanent link to Reproducible Capsule	https://zenodo.org/record/7484447
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	NumPy, Numba, SciPy, Joblib
If available Link to developer documentation/manual	https://dcor.readthedocs.io/
Support email for questions	vnmabus@gmail.com

1. Introduction

The so-called energy distance is a metric between the distributions of two random vectors introduced by Gabor J. Székely [1]. The name energy refers to some connections with the notion of Newton's potential energy [2]. It has a number of desirable properties such as rotational invariance, scale equivariance, and characterizing the equivalence of distributions (i.e., it is equal to zero if and only if the distributions are identical) [3,4]. The first application of a distance between distributions is usually testing homogeneity. In this sense, nonparametric tests based of energy distance have been proposed for testing the equality of multiple multivariate distributions [5] or for change point detection in time series [6], among others. Energy distance has also been used to provide a nonparametric extension of the classical ANOVA [7] and goodness-of-fit tests for several distributions [8–10]. Meanwhile, this metric can be also considered on its own, for example in hierarchical clustering [11].

In general, the statistics related to the energy distance are called energy statistics, or E-statistics [4]. The most popular energy statistics are distance covariance and correlation, which measure independence between two random vectors [12]. They can be seen as a generalization of the classical notions of covariance and Pearson's correlation as they are able to capture nonlinear dependencies and are defined for vectors of arbitrary dimensions. These measures have become very popular in recent years and have been used in many application areas such causal analysis [13], feature selection [14,15], robust statistics [16], and testing independence [4].

In this paper we present the functionalities of **dcor**, a Python package dedicated to E-statistics [17]. The library **dcor** is an open source project that seeks to provide statistical tools based on energy statistics to the Python community in an easy-to-use way. It contains different estimators for some E-statistics,

* Corresponding author.
E-mail addresses: carlos.ramos@uam.es (Carlos Ramos-Carreño), joseluis.torrecilla@uam.es (José L. Torrecilla).

https://doi.org/10.1016/j.softx.2023.101326
2352-7110/© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

with special attention to distance correlation and covariance, and nonparametric tests for both homogeneity and independence. These statistical tools are briefly described in Section 2. A complete description of all functionalities is provided in the package documentation, linked in the metadata.

During the design of the library, we have tried to maximize compatibility with other tools of the scientific Python ecosystem. Another focal point has been the quality and efficiency of the code, paying special attention to automated testing, code vectorization and compilation, or parallelization, among others. Section 3 is devoted to the implementation details related to performance and extensibility. Moreover, the computational efficiency of the principal measures in **dcor** is compared with the reference R package **energy** [18] and recent alternative implementations in both R and Python. Finally, the impact of the package is addressed in Section 4, and some general conclusions are drawn in Section 5.

2. Functionalities of the package

The main goal of the **dcor** package is to provide efficient estimators for distance correlation and other E-statistics, such as energy distance and partial distance correlation. In addition, hypothesis tests for homogeneity and independence of distributions based on these measures are provided. A brief description of these functionalities is given below.

2.1. Energy distance

Energy distance is a metric between the distributions of two random vectors X, Y that take values in $\mathbb{R}^d$ [3]. It is defined as

$$\mathcal{E}(X,Y) = 2\mathbb{E}(\|X-Y\|) - \mathbb{E}(\|X-X'\|) - \mathbb{E}(\|Y-Y'\|), \tag{1}$$

where $\|\cdot\|$ stands for the Euclidean norm, and X' and Y' denote independent and identically distributed copies of X and Y, respectively.

Alternatively, denoting their respective characteristic functions by $\phi_X(t) = \mathbb{E}[e^{itX}]$ and $\phi_Y(t) = \mathbb{E}[e^{itY}]$, this measure can be rewritten as

$$\mathcal{E}(X,Y) = \frac{1}{c_d}\int_{\mathbb{R}^d} \frac{|\phi_X(t)-\phi_Y(t)|^2}{\|t\|^{d+1}}dt, \tag{2}$$

where $c_d = \frac{\pi^{(1+d)/2}}{\Gamma((1+d)/2)}$ is half the surface area of the unit sphere in $\mathbb{R}^d$.

Let $\{x_i\}_{i=1}^{N_X}$ and $\{y_i\}_{i=1}^{N_Y}$ be two samples of X and Y with sizes N_X and N_Y, respectively. An estimator of the energy distance based on the sample means can be defined as

$$\begin{aligned}\mathcal{E}_{N_X,N_Y}(X,Y) = {} & \frac{2}{N_X N_Y}\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y}\|x_i-y_j\| \\ & - \frac{1}{N_X^2}\sum_{i=1}^{N_X}\sum_{j=1}^{N_X}\|x_i-x_j\| - \frac{1}{N_Y^2}\sum_{i=1}^{N_Y}\sum_{j=1}^{N_Y}\|y_i-y_j\|.\end{aligned} \tag{3}$$

The function `energy_distance` implements this estimator. In order to improve the robustness of the estimation, the **dcor** package allows the use of other centrality measures, such as a trimmed mean or the median, as proposed in [19]. Moreover, an unbiased version of this estimator based on the use of U-statistics is also available, as proposed in [19,20].

2.2. Distance covariance and correlation

Distance covariance $\mathcal{V}$, and distance correlation $\mathcal{R}$, are dependency measures between two random vectors, X and Y, with finite first moments [12,21]. Unlike the classical Pearson correlation, these measures can detect nonlinear dependencies. Indeed, they are equal to 0 if and only if the random vectors are independent. Furthermore, $\mathcal{V}$ and $\mathcal{R}$ can be defined for vectors of arbitrary dimensions, $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$.

In this context, the squared distance covariance $\mathcal{V}^2(X,Y)$ is defined as a weighted distance between the joint characteristic function $\phi_{X,Y}(t,s)$ and the product of the marginals $\phi_X(t)$ and $\phi_Y(t)$. The distance covariance $\mathcal{V}(X,Y)$ is then the nonnegative number that verifies

$$\mathcal{V}^2(X,Y) = \int_{\mathbb{R}^{p+q}} |\phi_{X,Y}(t,s) - \phi_X(t)\phi_Y(s)|^2 w(t,s)dtds, \tag{4}$$

where $w(t,s) = (c_p c_q \|t\|_p^{1+p}\|s\|_q^{1+q})^{-1}$, $\|\cdot\|_d$ is the euclidean norm in $\mathbb{R}^d$ and c_d is again half the surface area of the unit sphere in $\mathbb{R}^d$.

Analogously to classical Pearson correlation, distance correlation $\mathcal{R}(X,Y)$ is defined from the distance covariance as

$$\mathcal{R}^2(X,Y) = \begin{cases} \frac{\mathcal{V}^2(X,Y)}{\sqrt{\mathcal{V}^2(X,X)\mathcal{V}^2(Y,Y)}} & \text{if } \mathcal{V}^2(X,X)\mathcal{V}^2(Y,Y) > 0 \\ 0 & \text{if } \mathcal{V}^2(X,X)\mathcal{V}^2(Y,Y) = 0. \end{cases} \tag{5}$$

In spite of the apparent complexity of these definitions, distance covariance and correlation have a simple parameter-free estimator. This is an advantage over other popular dependency measures, such as mutual information, which require the estimation of additional smoothing parameters [22,23]. Given a sample $\{(x_i,y_i)\}_{i=1}^N$ of N observations of the joint random vector (X,Y), we define the double centered distance matrices A and B as

$$A_{i,j} = a_{i,j} - \frac{1}{N}\sum_{l=1}^N a_{il} - \frac{1}{N}\sum_{k=1}^N a_{kj} + \frac{1}{N^2}\sum_{k,l=1}^N a_{kl},$$

$$B_{i,j} = b_{i,j} - \frac{1}{N}\sum_{l=1}^N b_{il} - \frac{1}{N}\sum_{k=1}^N b_{kj} + \frac{1}{N^2}\sum_{k,l=1}^N b_{kl},$$

where $a_{ij} = \|x_i - x_j\|_p$ and $b_{ij} = \|y_i - y_j\|_q$. Then, the sample distance covariance is the square root of

$$\mathcal{V}_N^2(x,y) = \frac{1}{N^2}\sum_{i,j=1}^N A_{i,j}B_{i,j}. \tag{6}$$

Likewise, the (squared) sample distance correlation is the standardized (squared) sample covariance

$$\mathcal{R}_N^2(x,y) = \begin{cases} \frac{\mathcal{V}_N^2(x,y)}{\sqrt{\mathcal{V}_N^2(x,x)\mathcal{V}_N^2(y,y)}}, & \text{if } \mathcal{V}_N^2(x,x)\mathcal{V}_N^2(y,y) > 0, \\ 0, & \text{if } \mathcal{V}_N^2(x,x)\mathcal{V}_N^2(y,y) = 0. \end{cases} \tag{7}$$

Both estimators $\mathcal{V}_N$ and $\mathcal{R}_N$ converge almost surely to their population counterparts $\mathcal{V}$ and $\mathcal{R}$ when N tends to infinity [12].

The functions `distance_covariance` and `distance_correlation` implement the estimators (6) and (7), respectively. The library also provides unbiased and bias-corrected estimators of $\mathcal{V}^2(X,Y)$ and $\mathcal{R}^2(X,Y)$, in functions `u_distance_covariance_sqr` and `u_distance_correlation_sqr` [24]. Note that in this case one cannot take the square root as the estimator can take negative values. In addition, an affinely invariant version of distance correlation [25] is already available as `distance_correlation_af_inv`.

Despite of its simplicity, the main drawback of the estimator defined in Eq. (6) is its high computational cost. Due to the evaluation of the distance matrices, the complexity in both time and memory is $O(N^2)$. As an alternative, [26,27] propose two different estimators of the distance covariance based on the AVL-tree structure [28] and the mergesort algorithm, respectively. Both proposals has complexity $O(N \log N)$, although they are restricted

to univariate distributions ($p = q = 1$). These fast algorithms are available in the distance covariance and correlation functions by means of the parameter `method`, and are used by default when possible.

2.3. Partial distance correlation

Partial distance covariance and correlation are extensions of the aforementioned dependency measures that allow for controlling for the effects of a third random vector Z of arbitrary dimension [29].

Partial distance covariance is defined as

$$\mathcal{V}^*(X, Y; Z) = \begin{cases} \mathcal{V}^2(X, Y) - \frac{\mathcal{V}^2(X,Z)\mathcal{V}^2(Y,Z)}{\mathcal{V}^2(Z,Z)} & \text{if } \mathcal{V}^2(Z, Z) \neq 0 \\ \mathcal{V}^2(X, Y) & \text{if } \mathcal{V}^2(Z, Z) = 0, \end{cases}$$

while partial distance correlation is

$$\mathcal{R}^*(X, Y; Z) = \begin{cases} \frac{\mathcal{R}^2(X,Y) - \mathcal{R}^2(X,Z)\mathcal{R}^2(Y,Z)}{\sqrt{1-\mathcal{R}^4(X,Z)}\sqrt{1-\mathcal{R}^4(Y,Z)}} & \text{if } \mathcal{R}^4(X, Z) \neq 1 \text{ and } \mathcal{R}^4(Y, Z) \neq 1 \\ 0 & \text{if } \mathcal{R}^4(X, Z) = 1 \text{ or } \mathcal{R}^4(Y, Z) = 1. \end{cases}$$

The estimators of these quantities proposed in [24] are provided by functions `partial_distance_covariance` and `partial_distance_correlation`.

Finally, note that these dependency measures should be used carefully as they present some undesirable or counterintuitive properties [24, Sec. 4.2]. In particular, $\mathcal{R}^*(X, Y; Z) = 0$ does not always implies that X and Y are conditionally independent given Z and vice versa.

2.4. Hypothesis testing

One of the main applications of the measures between distributions is hypothesis testing. In this sense, measures based on E-statistics are no exception. For example, [8] proposes a goodness-of-fit test for multivariate normality based on the energy distance, and a test of independence can be found in [30], both implemented in the **energy** R package. Meanwhile, in the **dcor** package we can find three different tests: one for homogeneity and two for independence.

First, energy distance is used to define a permutation test of homogeneity [5]. This test is based on the properties of the statistic

$$T = \frac{N_X N_Y}{N_X + N_Y} \mathcal{E}_{N_X,N_Y}(X, Y), \tag{8}$$

where $\mathcal{E}_{N_X,N_Y}$ is the estimator of the energy distance defined in Eq. (3). This test, and its extension for more than two populations, is implemented in the function `homogeneity.energy_test`.

Second, a permutation test for detecting independence between two distributions is constructed using the distance covariance [12]. The statistic used for this test is $N\mathcal{V}_N^2$, where $\mathcal{V}_N^2$ is the estimator of the distance covariance defined in Eq. (6). Function `independence.distance_covariance_test` contains an implementation of this test.

Finally, an asymptotic test of independence for high dimensional vectors [31] is provided by `independence.distance_correlation_t_test`. This test relies on the convergence of the statistic

$$\mathcal{T}_N = \frac{\mathcal{R}_N^*}{\sqrt{1 - (\mathcal{R}_N^*)^2}} \sqrt{\frac{N(N-3)}{2} - 1}, \tag{9}$$

where $\mathcal{R}_N^*$ is the bias-corrected version of the estimator of the squared distance correlation proposed in [24].

Some additional parameters of these tests can be adjusted by the user, including the number of repetitions used in the permutation tests.

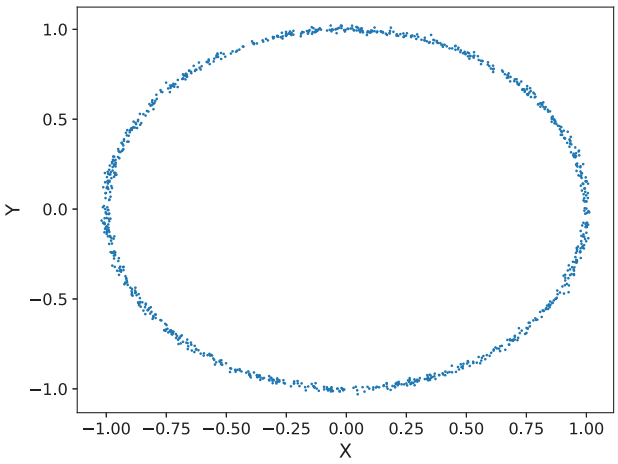

Fig. 1. Ring of data generated for the illustrative example.

2.5. Illustrative example

This section shows how to use the **dcor** package through a brief example with a toy dataset.

1. Generation of the dataset. In this case, we generate 1000 pairs of coordinates (x, y) conforming an annulus (see Fig. 1).

```
import numpy as np
import dcor

n_samples = 1000
random_state = np.random.default_rng(123456)
u = random_state.uniform(-1, 1, size=n_samples)

y = (np.cos(u * np.pi)
  + random_state.normal(0, 0.01, size=n_samples))
x = (np.sin(u * np.pi)
  + random_state.normal(0, 0.01, size=n_samples))
```

Hence, the random vectors X and Y are non independent, but identically distributed.

2. Distances. Computation of $\mathcal{E}(X, Y)$ and $\mathcal{R}(X, Y)$ with the AVL algorithm. Remember that fast implementations are only valid for one-dimensional variables.

```
dcor.energy_distance(x, y)
Out:
  0.00152....

dcor.distance_correlation(x, y, method="avl")
Out:
  0.2097....
```

The close-to-zero value of the energy distance reflects the homogeneity of the distributions of X and Y. Meanwhile, the distance correlation indicates a small dependency between them.

3. Hypothesis testing. Finally, we test homogeneity and independence to check if the above conclusions are statistically significant.

```
dcor.homogeneity.energy_test(
  x, y, num_resamples=100, random_state=random_state)
```

```
Out:
  HypothesisTest(pvalue=0.3069..., statistic=0.7649...)

dcor.independence.distance_covariance_test(
   x, y, num_resamples=100, random_state=random_state)
Out:
 HypothesisTest(pvalue=0.0099..., statistic=13.5334...)
```

Tests in **dcor** return an object with the p-value and the value of the statistic. As expected, with 95% confidence, we can accept the homogeneity (p-value $\simeq 0.31$) and reject the independence (p-value $\simeq 0.099$).

To conclude, let us point out that the **dcor** package owns detailed examples for most functionalities in the online documentation.[1] These examples includes different datasets, measurements of error and performance, and background about the statistical methods.

3. Implementation details

The package **dcor** is implemented in Python 3.8 and released in Github [17] under a MIT license. Releases are available both in PyPI[2] and the conda-forge channel from Anaconda.[3] In order to guarantee the quality of the library, it includes a comprehensive and automated collection of tests, and a complete documentation. The code contains also type annotations for static analyzers. Some additional features aiming at improving performance and extensibility are described below.

3.1. Performance

The functions of the **dcor** package make a careful use of vectorization to guarantee improved performance. In addition, fast algorithms for distance covariance and correlation described in Section 2.2 make use of the **Numba** library [32] to compile and optimize them, what entails further speed gains.

In this section we compare the performance of the **dcor** implementations of energy distance and distance correlation with their counterparts in other R and Python proposals. The R package **energy** [18] is the reference library when working with E-statistics. It is more similar to **dcor** than any other software in terms of approach and scope. A detailed comparison between **dcor** and **energy** is available in the documentation. The recent **dcortools** [33] addresses the subset of E-statistics relying on distance covariance and correlation with a very careful implementation. On the other hand, there are no Python packages devoted to energy statistics or subsets of them. Nonetheless, some isolated functionalities can be found in some statistical libraries. This is the case of **statsmodels** [34], **hyppo** [35], and **pingouin** [36], which include some few E-statistics among extensive catalogs of statistical tools.

Since the complexity in terms of the dimensions, p and q, of X and Y does not depends on the distance algorithms themselves, but to the implementation of the Euclidean distance in each language (R and Python), we study only the effect of varying the sample size. Then, we consider $p = q = 1$ and the following sample sizes $N = N_X = N_Y$: 10, 50, 100, 250, 500, 750, 1000. Datasets are generated randomly form a standard normal distribution, as data distribution has no effect in the computational cost. For each example, we show the minimum of 100 independent single calls of each estimator for each sample size. The choice of the minimum instead of the sample mean or other statistics, responds to the objective of discarding the effects of other running processes (see, e.g. [37] for further details).

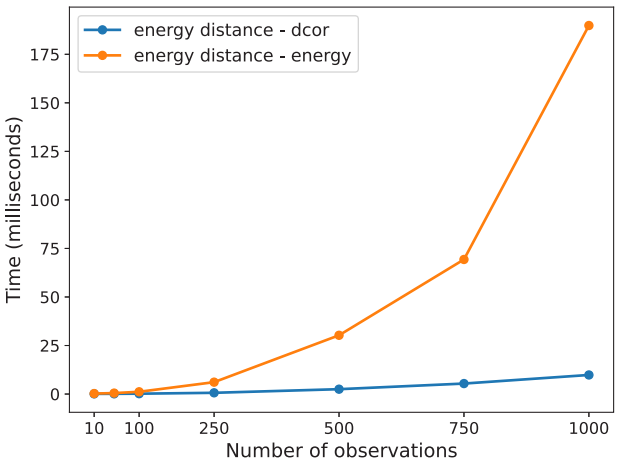

Fig. 2. Performance comparison between Python (**dcor** package) and R (**energy** package) implementations of energy distance.

On the one hand, Fig. 2 shows the comparison between **dcor** (Python) and **energy** (R) implementations of the energy distance following the expression in Eq. (3). At this moment, no other package in this study have a valid method for energy distance. On the other hand, Fig. 3 is dedicated to distance correlation implementations. Left panel presents the implementations of the original estimator defined in Eq. (6) joint with fast versions following both the AVL-based algorithm (**dcor**, **energy**, and **dcortools**) proposed in [26], and the fast mergesort-based implementation (**dcor**, and **hyppo**) proposed in [27]. The right panel focuses on fast algorithms, using a more appropriate scale to appreciate the differences. Distance covariance is not explicitly considered in the comparison as some of these packages do not expose that functionality to end users. Nonetheless, the performance of distance correlation directly depends on the performance of the distance covariance.

In all cases, **dcor** implementations clearly outperforms those in the reference package **energy** and their Python counterparts (**statsmodels**, **hyppo**, and **pingouin**). The recent R library **dcortools** is the only comparable with **dcor**. It exhibits also a very good performance, being slightly inferior to **dcor** with small sample sizes, but obtaining the best results with more than 250 observations. As expected, fast algorithms obtain the best results, although they are limited to univariate random vectors. In this context, **dcor** is the only library that implements the two fast algorithms, AVL and mergesort. Their results are very similar although the mergesort-based implementation shows a slightly, but consistently, better performance than the AVL approach.

Finally, the **dcor** package can exploit some additional opportunities for parallelization. A typical example is the repeated computation of a dependence measure, such as distance covariance, which naturally arises in variable selection or in correlation analysis. In this context, the function `rowwise` can be used to handle each pair of variables in parallel (employing several CPUs by means of the **Numba** library) when a fast algorithm for distance covariance is used. Another occasion for parallelization appears in permutation tests, which also involve repeating calls to the same function. The difference here is that the distances involved are the same for each permutation, although in a different order, so they need to be computed only once. We can then leverage the **Joblib** library [38] to launch the permutations in parallel with a configurable number of jobs. Note that these examples belong to the so called embarrassingly parallel problems, where the resultant speedup is directly proportional to the number of physical CPUs available.

[1] https://dcor.readthedocs.io/en/latest/auto_examples/index.html
[2] https://pypi.org/project/dcor
[3] https://anaconda.org/conda-forge/dcor

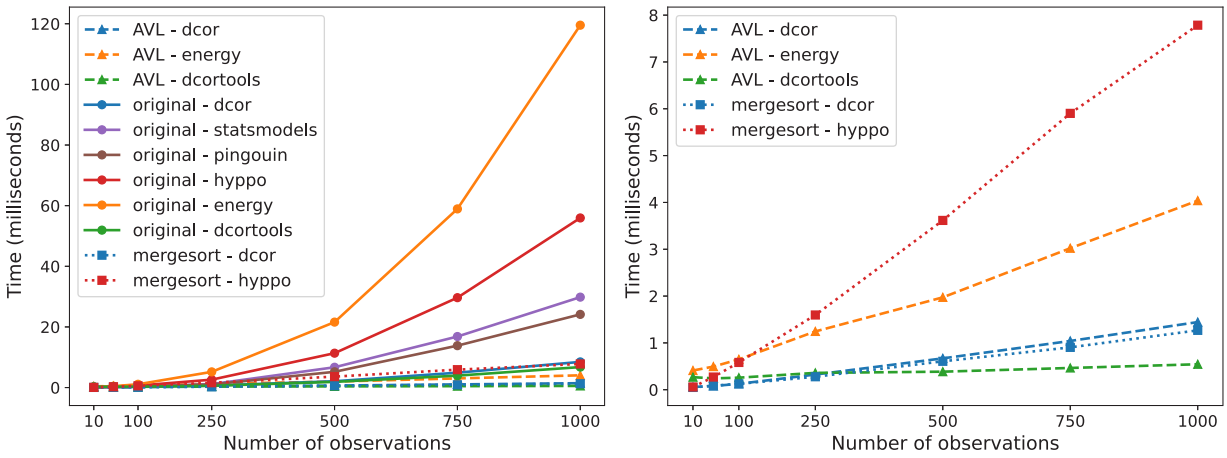

Fig. 3. Performance comparison among Python (packages **dcor**, **statsmodels**, **hyppo**, and **pingouin**) and R (packages **energy** and **dcortools**) implementations of distance correlation. On the left, all estimators available in these packages. On the right, only fast algorithms.

3.2. Extensibility

A major concern in the scientific Python ecosystem is the proliferation of different frameworks, each with its own similar, but incompatible, data structures. As a result, several implementations of the same non-trivial algorithms may arise, with the subsequent maintenance burden and fragmentation of the community. In the development process of the **dcor** package we have addressed this problem in two ways.

First and foremost, **dcor** adheres to the Python array API standard [39]. This specification was created to offer a common interface for different implementations of arrays and tensors in Python. This is a joint effort of the Python scientific community that tries to integrate, among others, **NumPy** CPU-based arrays [40], **CuPy** GPU-based ones [41], **Dask** distributed arrays [42] and the different types of tensors available in deep learning libraries, such as **Pytorch** [43] or **Tensorflow** [44]. The package **dcor** accepts objects that follow this standard, and therefore, it could be used in combination with these libraries when they finish their standardization effort. In order to guarantee that integration, **dcor** has been tested against the `numpy.array_api` module, which contains a minimal reference implementation of the Python array API standard.

Furthermore, special care has been taken in **dcor** to allow the use of arrays of arbitrary numeric type, including non-floating point types such as `Fraction` and `Decimal`. As a consequence, the original types will be preserved along all computations and results.

The only exception to these approaches are those few functions which are compiled, as currently **Numba** supports only a subset of array and number types.

4. Impact and applications

During the last few years, E-statistics, and in particular distance correlation, have attracted quite a bit of attention, accumulating thousands of citations and being used in many different application areas. The package **dcor** gives the Python scientific community access to a comprehensive set of significant measures and tests based on E-statistics in a unified framework, previously available only in R through the **energy** package. In addition, the library presents some differential characteristics such as the adoption of the Python array API standard, and fast algorithms and code optimizations which lead to better performances than those of their R and Python counterparts in most cases. As a result, **dcor** could be very useful in the wide variety of situations where E-statistics are. Indeed, it is already being used in a number of relevant scientific publications, as well as several open source scientific packages.

Up to now, the use of **dcor** has been particularly frequent in areas related to machine learning. For example, distance correlation is used in causal inference [45,46] to detect the dependence structure of the data. The energy distance and distance correlation have been considered for word embeddings in natural language processing problems [47,48]. Recent works in bias detection [49] make use of the distance correlation to uncover attributes that act as proxies for sensitive data. Furthermore, distance covariance and correlation are commonly used in dimensionality reduction strategies, both in multivariate [50,51] and functional [52] frameworks.

Meanwhile, the package has also proven to be useful in identifying medicinal plants [53], analyzing correlations between the Sustainable Development Goals of the United Nations [54] or in a financial context, for either diversifying investments [55] or optimizing technical indicators for prediction [56].

5. Conclusions

This article presents the package **dcor**, a package that provides functionalities based on E-statistics to the Python scientific community. In addition to include a varied set of statistical measures and hypothesis tests, the library has been designed with efficiency and extensibility criteria in mind. Some examples of this approach are the flexibility about different array and numeric types and the auxiliary tools for parallelization. The efficiency criterion is also reflected in code optimizations like vectorization and compilation, which allow **dcor** to obtain better performance than the alternatives in R.

The interest in these statistical tools has been shown by the good reception of the package in many different application areas since its first release in 2017. This attention made **dcor** a reference package for E-statistics in Python, and solidified our commitment with the development and expansion of the package. To this respect, our future plans include incorporating further developments derived from the theory of E-statistics, enriching the existing documentation with more tutorials and additional usage examples, and continue optimizing the existing functionalities.

To conclude, we want to emphasize our involvement with the open-source community, both addressing user comments and encouraging practitioners to contribute to the development of the **dcor** package.

CRediT authorship contribution statement

Carlos Ramos-Carreño: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **José L. Torrecilla:** Conceptualization, Validation, Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors acknowledge financial support from the Spanish Ministry of Science and Innovation, from projects PID2019-106827GB-I00/AEI/10.13039/501100011033 and PID2019-109387GB-I00. This research was also supported by an FPU grant (Formación de Profesorado Universitario) from the Spanish Ministry of Science, Innovation and Universities (MICIU) with reference FPU18/00047. Also, we would like to thank the reviewers for taking the time and effort necessary to review this work. We sincerely appreciate all valuable comments and suggestions, which helped us to improve the quality of the manuscript, documentation and code.

References

[1] Szekely GJ. Potential and kinetic energy in statistics. Lecture notes, Budapest Institute of Technology; 1989.
[2] Szekely G. E-statistics: the energy of statistical samples. Technical Report 02–16, Bowling Green State University, Department of Mathematics and Statistics; 2002, http://dx.doi.org/10.13140/RG.2.1.5063.9761.
[3] Székely GJ, Rizzo ML. Energy statistics: A class of statistics based on distances. J Statist Plann Inference 2013;143(8):1249–72. http://dx.doi.org/10.1016/j.jspi.2013.03.018.
[4] Rizzo ML, Székely GJ. Energy distance. Wiley Interdiscip Rev Comput Stat 2016;8(1):27–38.
[5] Székely G, Rizzo ML. Testing for equal distributions in high dimensions. InterStat 2004;5(16.10):1249–72.
[6] Kim AY, Marzban C, Percival DB, Stuetzle W. Using labeled data to evaluate change detectors in a multivariate streaming environment. In: Special section: visual information analysis for security, Signal Process In: Special section: visual information analysis for security, 2009;89(12):2529–36.http://dx.doi.org/10.1016/j.sigpro.2009.04.011,
[7] Rizzo ML, Székely GJ. DISCO analysis: A nonparametric extension of analysis of variance. Ann Appl Stat 2010;4(2):1034–55. http://dx.doi.org/10.1214/09-AOAS245.
[8] Székely GJ, Rizzo ML. A new test for multivariate normality. J Multivariate Anal 2005;93(1):58–80. http://dx.doi.org/10.1016/j.jmva.2003.12.002.
[9] Rizzo ML. New goodness-of-fit tests for Pareto distributions. ASTIN Bull 2009;39(2):691–715. http://dx.doi.org/10.2143/AST.39.2.2044654.
[10] Yang G. The energy goodness-of-fit test for univariate stable distributions (Ph.D. thesis), Bowling Green State University; 2012, URL https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?p10_etd_subid=49947&clear=10.
[11] Szekely GJ, Rizzo ML. Hierarchical clustering via joint between-within distances: Extending Ward's minimum variance method. J Classification 2005;22(2):151–83. http://dx.doi.org/10.1007/s00357-005-0012-9.
[12] Székely GJ, Rizzo ML, Bakirov NK. Measuring and testing dependence by correlation of distances. Ann Statist 2007;35(6):2769–94. http://dx.doi.org/10.1214/009053607000000505, MR2382665.
[13] Zhang X, Podobnik B, Kenett DY, Eugene Stanley H. Systemic risk and causality dynamics of the world international shipping market. Phys A Stat Mech Appl 2014;415:43–53. http://dx.doi.org/10.1016/j.physa.2014.07.068.
[14] Yenigün CD, Rizzo ML. Variable selection in regression using maximal correlation and distance correlation. J Stat Comput Simul 2015;85(8):1692–705. http://dx.doi.org/10.1080/00949655.2014.895354.
[15] Berrendero JR, Cuevas A, Torrecilla JL. Variable selection in functional data classification: A maxima-hunting proposal. Statist Sinica 2016;26(2):619–38. http://dx.doi.org/10.5705/ss.202014.0014.
[16] Kasieczka G, Shih D. Robust jet classifiers through distance correlation. Phys Rev Lett 2020;125(12):122001. http://dx.doi.org/10.1103/PhysRevLett.125.122001.
[17] Ramos-Carreño C. dcor: Distance correlation and related E-statistics in Python. 2022, Zenodo. http://dx.doi.org/10.5281/zenodo.3468124.
[18] Rizzo M, Szekely G. Energy: E-statistics: Multivariate inference via the energy of data. 2022, URL https://CRAN.R-project.org/package=energy.
[19] James NA, Kejariwal A, Matteson DS. Leveraging cloud data to mitigate user experience from 'breaking bad'. In: 2016 IEEE international conference on big data. 2016, p. 3499–508. http://dx.doi.org/10.1109/BigData.2016.7841013.
[20] Matteson DS, James NA. A nonparametric approach for multiple change point analysis of multivariate data. J Amer Statist Assoc 2014;109(505):334–45. http://dx.doi.org/10.1080/01621459.2013.849605.
[21] Székely GJ, Rizzo ML. Brownian distance covariance. Ann Appl Stat 2009;3(4):1236–65. http://dx.doi.org/10.1214/09-AOAS312, MR2752127.
[22] Vergara JR, Estévez PA. A review of feature selection methods based on mutual information. Neural Comput Appl 2014;24(1):175–86. http://dx.doi.org/10.1007/s00521-013-1368-0.
[23] Laarne P, Zaidan MA, Nieminen T. Ennemi: Non-linear correlation detection with mutual information. SoftwareX 2021;14:100686. http://dx.doi.org/10.1016/j.softx.2021.100686.
[24] Székely GJ, Rizzo ML. Partial distance correlation with methods for dissimilarities. Ann Statist 2014;42(6):2382–412. http://dx.doi.org/10.1214/14-AOS1255, MR3269983.
[25] Dueck J, Edelmann D, Gneiting T, Richards D. The affinely invariant distance correlation. Bernoulli 2014;20(4):2305–30. http://dx.doi.org/10.3150/13-BEJ558.
[26] Huo X, Székely GJ. Fast computing for distance covariance. Technometrics 2016;58(4):435–47. http://dx.doi.org/10.1080/00401706.2015.1054435.
[27] Chaudhuri A, Hu W. A fast algorithm for computing distance correlation. Comput Statist Data Anal 2019;135:15–24. http://dx.doi.org/10.1016/j.csda.2019.01.016.
[28] Adelson-Velskii GM, Landis EM. An algorithm for organization of information. Proc USSR Acad Sci 1962;146(2):263–6.
[29] Székely GJ, Rizzo ML. The energy of data. Annu Rev Stat Appl 2017;4(1):447–79. http://dx.doi.org/10.1146/annurev-statistics-060116-054026.
[30] Bakirov NK, Rizzo ML, Székely GJ. A multivariate nonparametric test of independence. J Multivariate Anal 2006;97(8):1742–56. http://dx.doi.org/10.1016/j.jmva.2005.10.005.
[31] Székely GJ, Rizzo ML. The distance correlation T-test of independence in high dimension. J Multivariate Anal 2013;117:193–213. http://dx.doi.org/10.1016/j.jmva.2013.02.012.
[32] Lam SK, Pitrou A, Seibert S. Numba: A LLVM-based Python JIT compiler. In: Proceedings of the second workshop on the LLVM compiler infrastructure in HPC. LLVM '15, New York, NY, USA: Association for Computing Machinery; 2015, p. 1–6. http://dx.doi.org/10.1145/2833157.2833162.
[33] Edelmann D, Fiedler J. Dcortools: Providing fast and flexible functions for distance correlation analysis. 2022, URL https://CRAN.R-project.org/package=dcortools.
[34] Seabold S, Perktold J. statsmodels: Econometric and statistical modeling with Python. In: 9th Python in science conference. 2010.
[35] Panda S, Palaniappan S, Xiong J, Bridgeford E, Mehta R, Shen C, et al. hyppo: A multivariate hypothesis testing Python package. 2021, URL https://github.com/neurodata/hyppo.
[36] Vallat R. Pingouin: Statistics in Python. J Open Source Softw 2018;3(31):1026. http://dx.doi.org/10.21105/joss.01026.
[37] Chen J, Revels J. Robust benchmarking in noisy environments. In: Proceedings of the 20th annual IEEE high performance extreme computing conference. 2016.
[38] Team JD. Joblib: Running Python functions as pipeline jobs. 2020, URL https://joblib.readthedocs.io/.
[39] Consortium for Python Data API Standards. Python Array API Standard. URL https://data-apis.org/array-api.
[40] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585(7825):357–62. http://dx.doi.org/10.1038/s41586-020-2649-2.
[41] Okuta R, Unno Y, Nishino D, Hido S, Loomis C. CuPy: A NumPy-compatible library for NVIDIA GPU calculations. In: Proceedings of workshop on machine learning systems (LearningSys) in the thirty-first annual conference on neural information processing systems. 2017, URL http://learningsys.org/nips17/assets/papers/paper_16.pdf.
[42] Dask Development Team. Dask: Library for dynamic task scheduling. 2016, URL https://dask.org.

[43] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems. Vol. 32. Curran Associates, Inc.; 2019.
[44] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow, Large-scale machine learning on heterogeneous systems. 2015, http://dx.doi.org/10.5281/zenodo.4724125.
[45] Markham A, Chivukula A, Grosse-Wentrup M. MeDIL: A Python package for causal modelling. In: Proceedings of the 10th international conference on probabilistic graphical models. PMLR; 2020, p. 621–4.
[46] Runge J. Tigramite – Causal inference and causal discovery for time series datasets. 2022, URL https://github.com/jakobrunge/tigramite.
[47] Zhelezniak V, Shen A, Busbridge D, Savkov A, Hammerla N. Correlations between word vector sets. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing. Hong Kong, China: Association for Computational Linguistics; 2019, p. 77–87. http://dx.doi.org/10.18653/v1/D19-1008.
[48] Kayal S. Unsupervised sentence-embeddings by manifold approximation and projection. In: Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume. Online: Association for Computational Linguistics; 2021, p. 1–11. http://dx.doi.org/10.18653/v1/2021.eacl-main.1.
[49] Synthesized. FairLens: Identify bias and measure fairness of your data. 2022, Synthesized. URL https://github.com/synthesized-io/fairlens.
[50] Menvouta EJ, Serneels S, Verdonck T. Direpack: A Python 3 package for state-of-the-art statistical dimension reduction methods. 2020, arXiv. arXiv:2006.01635. http://dx.doi.org/10.48550/arXiv.2006.01635.
[51] Böhm JN, Berens P, Kobak D. Attraction-repulsion spectrum in neighbor embeddings. J Mach Learn Res 2022;23(95):1–32.
[52] Ramos-Carreño C, Torrecilla JL, Carbajo-Berrocal M, Marcos P, Suárez A. scikit-fda: A Python package for functional data analysis. 2022, arXiv. arXiv:2211.02566. http://dx.doi.org/10.48550/arXiv.2211.02566.
[53] Kharyuk P, Nazarenko D, Oseledets I, Rodin I, Shpigun O, Tsitsilin A, et al. Employing fingerprinting of medicinal plants by means of LC-MS and machine learning for species identification task. Sci Rep 2018;8(1):17053. http://dx.doi.org/10.1038/s41598-018-35399-z.
[54] Laumann F, von Kügelgen J, Kanashiro Uehara TH, Barahona M. Complex interlinkages, key objectives, and nexuses among the Sustainable Development Goals and climate change: A network analysis. Lancet Planet Health 2022;6(5):e422–30. http://dx.doi.org/10.1016/S2542-5196(22)00070-5.
[55] Benowitz M. Hedgecraft: A Portfolio Management Algorithm for the 21st Century. URL https://github.com/mayabenowitz/Hedgecraft.
[56] Richardson J. TuneTA: Intelligently Optimizes Technical Indicators and Optionally Selects the Least Intercorrelated for Use in Machine Learning Models. URL https://github.com/jmrichardson/tuneta.