Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

QuadratiK: A Python and R package for clustering on the sphere and goodness-of-fit tests

Giovanni Saraceno [a,1,2], Raktim Mukhopadhyay [a,b,1], Marianthi Markatou [a,c,*]

[a] Department of Biostatistics, University at Buffalo, NY, USA
[b] Institute for Artificial Intelligence and Data Science, University at Buffalo, NY, USA
[c] Department of Medicine, Jacobs School of Medicine and Biomedical Sciences, University at Buffalo, NY, USA

ARTICLE INFO

Dataset link: https://github.com/rmj3197/QuadratiK-Performance, https://github.com/rmj3197/QuadratiK-Performance, https://github.com/giovsaraceno/simulations-rpkb, https://github.com/rmj3197/QuadratiK-Performance

MSC:
62G10
62H30

Keywords:
Kernel-based quadratic distances
Clustering on the sphere
Two-sample test
k-sample test
R
Python

ABSTRACT

We introduce `QuadratiK`, an open-source software, implemented in `R` and `Python`. `QuadratiK` supports normality tests, and two and k-sample tests, using kernel-based quadratic distances. The software also includes tests for uniformity on the d-dimensional sphere and a clustering algorithm using the Poisson kernel-based densities. Functions for generating random samples from these densities are included. These methods are encoded via object-oriented and extensively unit-tested implementations. `QuadratiK` offers graphical functions that enhance user experience by facilitating the validation, visualization, and interpretation of clustering results. We compare `QuadratiK` with related available libraries and provide illustrative code examples. In summary, `QuadratiK` offers a powerful suite of tools in `R` and `Python`, enabling researchers and practitioners to perform meaningful analyses and derive valid and reproducible inference across a wide range of fields. The `R`[3] and `Python`[4] codes are available under the GPL-3.0 license. Finally, we propose a dashboard application, a graphical user interface to the implemented methods, with the aim to facilitate the usage of the software among practitioners from different domains.

Code metadata

Current code version	R Version - 1.1.3, Python Version - 1.1.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00651
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/0347457/tree/v1
Legal Code License	GNU General Public License (GPL-3.0)
Code versioning system used	git
Software code languages, tools, and services used	R, Python, C++
Compilation requirements, operating environments & dependencies	Requires R version >= 3.5.0; Requires Python >= 3.9, $! = 3.9.7$, < 3.14
If available Link to developer documentation/manual	R Version - https://docs.ropensci.org/QuadratiK/, Python Version - https://quadratik.readthedocs.io/en/latest/
Support emails for questions	giovanni.saraceno@unipd.it, raktimmu@buffalo.edu

* Correspondence to: 726 Kimball Tower, Dept. of Biostatistics, SPHHP, 3435 Main Street, Buffalo, NY 14214.
E-mail addresses: giovanni.saraceno@unipd.it (Giovanni Saraceno), raktimmu@buffalo.edu (Raktim Mukhopadhyay), markatou@buffalo.edu (Marianthi Markatou).
1 These authors contributed equally as first authors.
2 Present address: Department of Statistical Sciences, University of Padova, Italy.
3 https://github.com/ropensci/QuadratiK
4 https://github.com/rmj3197/QuadratiK

https://doi.org/10.1016/j.softx.2025.102155
Received 2 December 2024; Received in revised form 28 March 2025; Accepted 31 March 2025
Available online 19 May 2025

Software metadata

Current software version	R Version - 1.1.3, Python Version - 1.1.3
Permanent link to executables of this version	R Version - https://cran.r-project.org/src/contrib/QuadratiK_1.1.3.tar.gz, Python Version - https://github.com/rmj3197/QuadratiK/releases/tag/1.1.3
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/0347457/tree/v1
Legal Software License	GNU General Public License (GPL-3.0)
Computing platforms/Operating Systems	Linux, macOS, Microsoft Windows, Unix-like
Installation requirements & dependencies	R Version: Installation Requirements - https://docs.ropensci.org/QuadratiK/, Dependencies - https://github.com/ropensci/QuadratiK/blob/main/DESCRIPTION, Python Version: Installation Requirements and Dependencies - https://quadratik.readthedocs.io/en/latest/getting_started/installation.html
If available, link to user manual - if formally published include a reference to the publication in the reference list	R Version - https://docs.ropensci.org/QuadratiK/, Python Version - https://quadratik.readthedocs.io/en/latest/
Support email for questions	giovanni.saraceno@unipd.it, raktimmu@buffalo.edu

1. Introduction

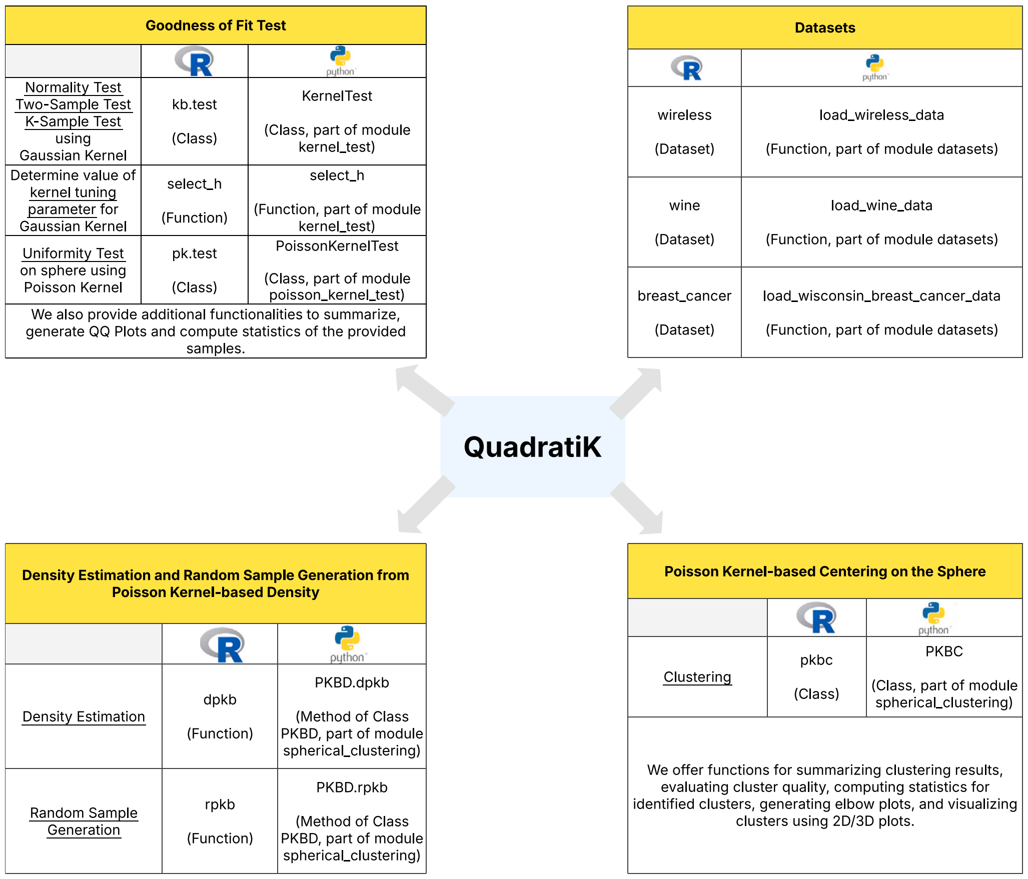
The Goodness-of-Fit (GoF) problem holds significant importance in statistical research for two main reasons: firstly, GoF tests are a fundamental method for evaluating whether data align with a presumed probability distribution, and secondly, they enable the comparison of two or more samples to identify differences or similarities between groups or conditions. Typically, a GoF test procedure is formulated by calculating a distance-like measure between the null hypothesis distribution and the observed data. Some of the commonly used tests in this category are the Kolmogorov–Smirnov test, performed through the R function `ks.test` in the `stats` package, the Cramér–von Mises test, in the R package `goftest` [1], and Anderson–Darling test, available in the R package `goft` [2]. These tests are also available in the Python library `scipy` [3]. The R package `vsgoftest` performs GoF tests of various families of distributions (uniform, normal, lognormal, exponential, gamma, Weibull, Pareto, Fisher, Laplace and Beta) based on Shannon entropy and the Kullback–Leibler divergence. These tests were developed by [4,5]. The `GoFKernel` package contains an implementation of Fan's test [6]. The two-sample problem has a significant body of literature. Graph-based ranking strategies have been used to extend univariate two-sample GoF tests to high-dimensional multivariate settings. Ref. [7] built on the univariate run-based test introduced by [8], the univariate two-sample Kolmogorov–Smirnov test, and modified Kolmogorov–Smirnov test for scale alternatives, which are all available in the R package `GSAR`, [9] proposes a two-sample test based on interpoint distances, implemented in the R package `crossmatch`. The R package `energy`, developed by [10] offers the energy statistic, initially proposed by [11], as well as a collection of test statistics for multivariate inference based on energy statistics. Additionally, [12] developed the Maximum Mean Discrepancy (MMD) test statistic using kernel mean embedding properties, which can be executed via the R package `kernlab`. For the k-sample testing problem, the R package `kSamples` [13] includes several nonparametric rank score k-sample tests, such as Kruskal–Wallis [14], van der Waerden scores, normal scores [15], and the Anderson–Darling test. The R package `coin` [16] provides an implementation of permutation tests tailored against location and scale alternatives, and for survival distributions. Recently, [17] construct k-sample tests by using any dependence measure, such as the distance covariance and the Hilbert–Schmidt independence criterion which are proved to be equivalent to the two-sample statistics Energy and MMD, respectively. These tests are available in the Python package `hyppo` [18].

When data are represented as points on the surface of a unit hypersphere $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\|^2 = 1\}$, the R package `circular` [19] provides Rayleigh's test, Rao's Spacing test, Kuiper's and the Watson's tests of uniformity for circular data. Ref. [20] proposes a test for uniformity based on nonnegative trigonometric sums, available in the R package `CircNNTSR` [20].

We present `QuadratiK` that provides a suite of methods for multivariate analysis and inference using quadratic distances [21–24]. This package is designed to handle high-dimensional and large sample size datasets efficiently, leveraging optimized `C++` implementations to accelerate calculations. The encoded quadratic distance-based GoF methods depend on a tuning parameter that is intimately connected with the power of the tests. The software offers the capability to choose the optimal tuning parameter and employs parallel computing to ensure feasible computational times. It includes a clustering algorithm for spherical data based on Poisson Kernel-Based Densities (PKBDs), functions for computing density values and for generating random samples from PKBD based on three sampling methods. To aid interpretation of the clustering results, the software provides graphical functions for validating and visualizing these results. Implemented in both R and `Python`, `QuadratiK` aims to integrate into existing workflows, promoting widespread adoption and application of these advanced tools. Finally, we propose a dashboard application, a graphical user interface to the implemented methods, with the aim to facilitate further usage of the presented software among practitioners.

2. Software description

The `QuadratiK` package implements methods based on kernel-based quadratic distances with an object-oriented approach. In the `Python` implementation, the package is organized into three main modules: `kernel_test`, `poisson_kernel_test` and `spherical_clustering`. The `KernelTest` class, in the `kernel_test` module, executes normality, two-sample, and k-sample tests using the Gaussian kernel, while the `select_h` function is used for the optimal choice of the tuning parameter. The `PoissonKernelTest` class, within the `poisson_kernel_test` module, handles uniformity tests for spherical data. The `spherical_clustering` module contains the `PKBC` and `PKBD` classes. `PKBC` performs spherical data clustering using the Poisson kernel-based density. It also provides descriptive statistics and elbow plots, and aids the evaluation of the clustering results. `PKBD` provides the methods `dpkb` and `rpkb` for computing density and generating random samples from PKBD, respectively. The `Python` package leverages the `joblib` package for parallel processing, enhancing computational efficiency. Additionally, tools for data visualization and descriptive analysis can be found in the `tools` module. Finally, a user interface can be used via the `UI` class, facilitating interaction without the need for programming expertise. In the R implementation, the package structure mirrors that of the `Python` package. The function `kb.test` performs kernel-based quadratic distance one-sample normality tests, two-sample and k-sample comparisons using the Gaussian kernel. The `select_h` function is included for optimal choice of the tuning parameter, enhancing performance through parallel processing with the `doParallel` package. Additional discussion on the `select_h` function is presented in Appendix A. The function



QuadratiK

Goodness of Fit Test

	R	python
Normality Test Two-Sample Test K-Sample Test using Gaussian Kernel	kb.test (Class)	KernelTest (Class, part of module kernel_test)
Determine value of kernel tuning parameter for Gaussian Kernel	select_h (Function)	select_h (Function, part of module kernel_test)
Uniformity Test on sphere using Poisson Kernel	pk.test (Class)	PoissonKernelTest (Class, part of module poisson_kernel_test)

We also provide additional functionalities to summarize, generate QQ Plots and compute statistics of the provided samples.

Datasets

R	python
wireless (Dataset)	load_wireless_data (Function, part of module datasets)
wine (Dataset)	load_wine_data (Function, part of module datasets)
breast_cancer (Dataset)	load_wisconsin_breast_cancer_data (Function, part of module datasets)

Density Estimation and Random Sample Generation from Poisson Kernel-based Density

	R	python
Density Estimation	dpkb (Function)	PKBD.dpkb (Method of Class PKBD, part of module spherical_clustering)
Random Sample Generation	rpkb (Function)	PKBD.rpkb (Method of Class PKBD, part of module spherical_clustering)

Poisson Kernel-based Centering on the Sphere

	R	python
Clustering	pkbc (Class)	PKBC (Class, part of module spherical_clustering)

We offer functions for summarizing clustering results, evaluating cluster quality, computing statistics for identified clusters, generating elbow plots, and visualizing clusters using 2D/3D plots.

Fig. 1. An overview of the organizational structure of QuadratiK in R and Python, highlighting the different core functionalities of the package.

pk.test is designed to test uniformity using the Poisson kernel, and the pkbc function performs clustering using the Poisson kernel-based density. Additional utility functions dpkb and rpkb provide density computation and random sample generation from this distribution. Fig. 1 illustrates the structure and functionalities offered by the QuadratiK package, in both R and Python.

The codes in R and Python are based on best practices of software development, following the standards for the R and Python communities, respectively, and are subjected to unit testing with continuous integration. Both the R and Python versions include vignettes for all its main functionalities, available at https://docs.ropensci.org/QuadratiK/articles/ for R and at https://quadratik.readthedocs.io/en/latest/user_guide/basic_usage.html for Python, offering detailed descriptions of the implemented methods along with usage examples on simulated and real datasets. These vignettes serve as a comprehensive resource to guide users in applying the software effectively. For detailed information and examples, users can refer to the documentation on CRAN for the R or at https://docs.ropensci.org/QuadratiK/, and https://quadratik.readthedocs.io/en/latest/ for the Python package. The GitHub repositories https://github.com/rmj3197/QuadratiK and https://github.com/ropensci/QuadratiK also offer a platform, for the Python and R users respectively, to request help or features by raising issues. Furthermore, the repositories also provide user documentation and contribution guidelines.

2.1. Comparisons with existing packages

This section compares the QuadratiK package with various relevant packages implemented in both R and Python with related functionalities. The energy package implements GoF tests based on the energy statistics for multivariate and univariate inference. Additionally, the package also includes k-groups and hierarchical clustering algorithms based on the energy distance. Similar to our package, this package implements GoF tests as well as clustering algorithms. The package kernlab implemented in R, includes a number of methods for classification, regression, clustering, novelty detection, quantile regression and dimensionality reduction. We are interested in this package because kernlab also implements the MMD test statistic for performing the two-sample test. The package GSAR provides a set of statistical methods for self-contained gene set analysis. It includes several two-sample multivariate nonparametric tests such as multivariate Kolmogorov–Smirnov test of means and multivariate radial Kolmogorov–Smirnov test of variance. The Python library StaTDS [25] provides a collection of classical statistical tests, commonly used in data science applications. However, notice that QuadratiK encodes state-of-the-art methods designed to handle large and high-dimensional datasets, thus distinguishing its content from traditional approaches available in StaTDS. The hyppo package implemented in Python consists of methods for multivariate k-sample tests. This package includes python implementation of methods also available in other R packages such as the MMD and Energy tests. The package sphunif includes implementation of more than 35 uniformity tests on the sphere. Similar to our package, C++ is also used for implementing the methods in this package. In Appendix B, a small simulation study is conducted to show the computational time needed by the test for uniformity for different combinations of dimension and sample size, including a discussion on how to handle larger datasets. Finally, the package Directional provides a collection of functions for directional data including hypothesis testing, regression and discriminant analysis. It provides functions for evaluating the density value and for generating random samples from the PKBD. In QuadratiK three sampling methods based on the angular central Gaussian distribution (rejacg), the von Mises–Fisher distribution (rejvmf) and the projected Saw distribution (rejpsaw)

Table 1
Comparison of various relevant packages. *Active Development* is determined by the release of any new version of the package between January 1, 2024, and Feb 28, 2025.

Packages	Programming Language	Active Development	Package Repository	Unit Tests (Coverage %)	Continuous Integration
energy	R/C++	✓	CRAN	✗	✗
kernlab	R	✓	CRAN	✗	✗
GSAR	R	✓	Bioconductor	✗	✗
sphunif	R/C++	✓	CRAN	✓ (85%)	✓
hyppo	Python	✓	PyPI	✓ (97%)	✓
StaTDS	Python	✓	PyPI	✓ (78%)	✗
Directional	R	✓	CRAN	✗	✗
QuadratiK	R/C++	✓	CRAN	✓ (100%)	✓
	Python	✓	PyPI	✓ (98%)	✓

are available, while only the first method is available in Directional. In Appendix C, we describe a simulation study for comparing the performance of QuadratiK with the Directional package in terms of generating random samples from the PKBD.

In Table 1, we present a brief comparison of the various packages. The comparison is in terms of implementation aspects such as programming language, active development, package repository, unit tests and coverage and continuous integration.

3. Illustrative examples

3.1. Generate random samples from PKBD

In this example, we generate observations from the Poisson kernel-based distribution on the sphere, S^{d-1}, using the Python package as shown in Listing 1. In the Python package two algorithms are available to generate random samples using the PKBD class, the acceptance-rejection algorithm using rejvmf and rejacg. In the example, we consider mean direction $\mu = (0, 1, 1)$ and dimension $d = 3$ with concentration parameter $\rho = 0.95$. We sample $n = 500$ observations from the available methods. The generated samples are depicted in Fig. 2. In addition to the rejvmf and rejacg methods, the R package also includes an algorithm to generate samples from PKBD using rejpsaw.

```python
# Import the PKBD class from the
    spherical_clustering module in the
    QuadratiK package
from QuadratiK.spherical_clustering import
    PKBD
# Instantiate the PKBD class
pkbd = PKBD()
# Generate 500 samples from PKBD using
    rejvmf
samples_rejvmf = pkbd.rpkb(
    n=500, mu=[0, 1, 1], rho=0.95, method="
    rejvmf", random_state=42
)
# Generate 500 samples PKBD using rejacg
samples_rejacg = pkbd.rpkb(
    n=500, mu=[0, 1, 1], rho=0.95, method="
    rejacg", random_state=42
)
```

Listing 1: Python code for generating random samples from PKBD

3.2. Dashboard application

We have also developed a graphical user interface (GUI) for the QuadratiK package in Python, utilizing the streamlit framework. The primary goal of the graphical interface is to facilitate user interaction with the methods available in the package without the need for programming in either R or Python. The dashboard is also available for potential users to explore at https://dashboard-quadratik.apps.buffalo.edu/.

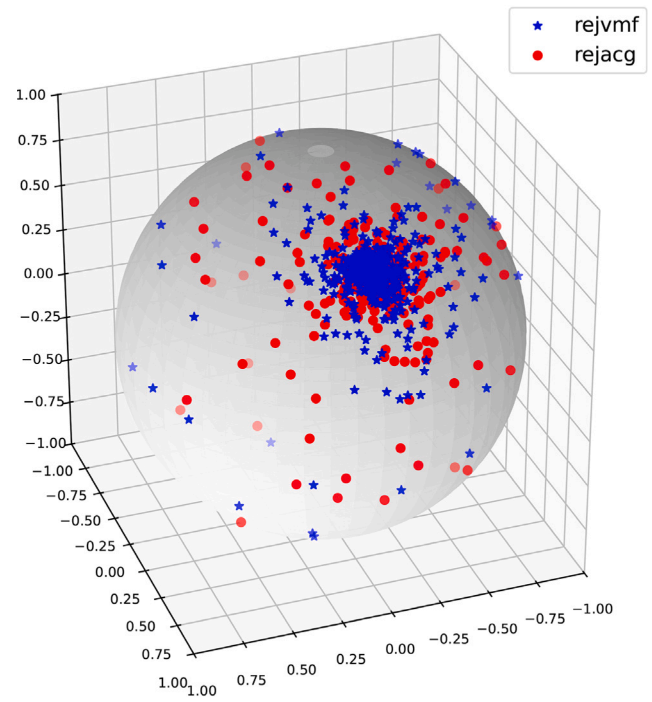
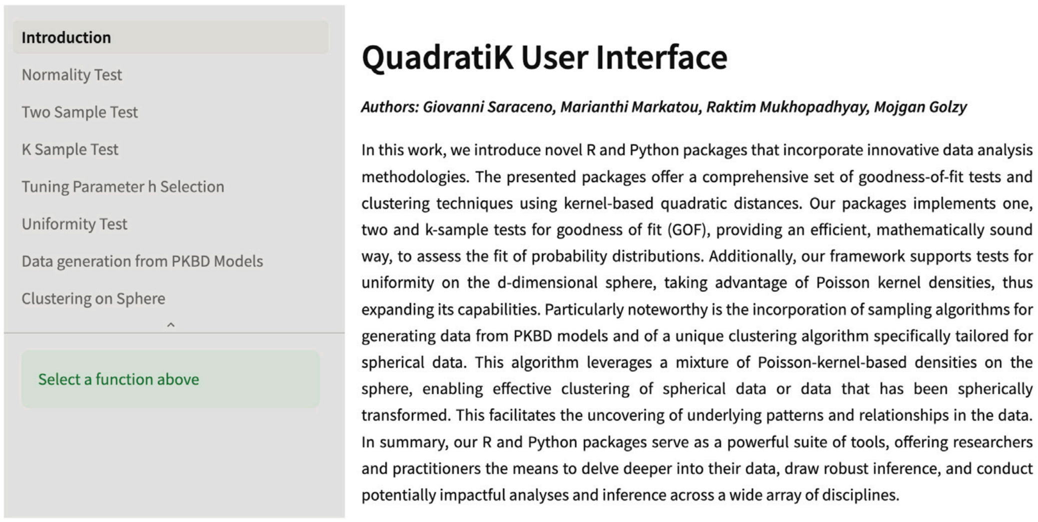


Fig. 2. Samples generated from PKBD generated using rejvmf (blue colored star ★) and rejacg (red colored dot ●) methods displayed on the surface of the unit sphere. The code for the visualization is given in Appendix D. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```python
# Import the UI class from the QuadratiK.ui
    module
from QuadratiK.ui import UI
# Create an instance of the UI class and
    call its run method to start the user
    interface
UI().run()
# Output showing how the dashboard can be
    accessed on the local machine
You can now view your Streamlit app in your
    browser.
  Local URL: http://localhost:8502
  Network URL: http://128.205.147.7:8502
```

Listing 2: Python code for instantiating the dashboard application.



QuadratiK User Interface

Authors: Giovanni Saraceno, Marianthi Markatou, Raktim Mukhopadhyay, Mojgan Golzy

In this work, we introduce novel R and Python packages that incorporate innovative data analysis methodologies. The presented packages offer a comprehensive set of goodness-of-fit tests and clustering techniques using kernel-based quadratic distances. Our packages implements one, two and k-sample tests for goodness of fit (GOF), providing an efficient, mathematically sound way, to assess the fit of probability distributions. Additionally, our framework supports tests for uniformity on the d-dimensional sphere, taking advantage of Poisson kernel densities, thus expanding its capabilities. Particularly noteworthy is the incorporation of sampling algorithms for generating data from PKBD models and of a unique clustering algorithm specifically tailored for spherical data. This algorithm leverages a mixture of Poisson-kernel-based densities on the sphere, enabling effective clustering of spherical data or data that has been spherically transformed. This facilitates the uncovering of underlying patterns and relationships in the data. In summary, our R and Python packages serve as a powerful suite of tools, offering researchers and practitioners the means to delve deeper into their data, draw robust inference, and conduct potentially impactful analyses and inference across a wide array of disciplines.

Fig. 3. The first page of the dashboard application displaying the supported functionalities in the left pane and general information on the software at the center. The dashboard can be accessed at https://dashboard-quadratik.apps.buffalo.edu/.

The interface supports all available functionalities of the package as shown in Fig. 3. Specifically, it includes the ‘Normality Test’, ‘Two Sample Test’, ‘K-Sample Test’, ‘Tuning Parameter Selection’, ‘Uniformity Test’,‘Data generation from PKBD Models’ and ‘Clustering on Sphere’. Details on the functionalities and their usage instructions can be found at https://quadratik.readthedocs.io/en/latest/user_guide/dashboard_application_usage.html.

4. Impact

The `QuadratiK` package provides a computationally efficient global framework for inference and data analysis based on quadratic distances, making these tools accessible for very large, high-dimensional datasets. It allows the exploration of new research questions, both building on the theory of the methods presented and on their practical use. The goodness-of-fit methods included in the software are based on the theory of quadratic distances, and the selection of their hyperparameters is intimately connected with the statistical concepts of level and power of tests; thus, arbitrary suboptimal selection of these parameters is avoided. Therefore, the software facilitates the use of optimal methods for the analysis of data.

Considering the MMD and energy distances, the main difference between these two metrics lies in the function space. Our framework includes as special cases the MMD and energy distances through an appropriate choice of the kernel function; furthermore, the selection of the kernel tuning parameter via the proposed algorithm allows for the identification of alternative hypotheses supported by the collected data. Our proposed tests outperform, in terms of power, the MMD and energy tests for alternatives close to the null hypothesis and for distributions with heavy tails. In terms of computational time, the encoded tests exhibit comparable time to methods available in literature. For more details on the two-sample and k-sample tests, [24] provides a detailed theoretical comparison as well as extensive simulation studies investigating the performance of the tests in terms of level and power. Furthermore, [23] discusses in detail tests of uniformity on the d-dimensional sphere. The clustering algorithm offers an efficient and robust method for analyzing spherical or spherically transformed data, that is, data where directions or angular relationships are of interest. This algorithm has been utilized for assessing quality of life predictors for survival in oncology studies (e.g., bladder cancer [26]), generating causal maps for multi-document summarization [27], and developing a patent of event prediction systems through causal map generation and visualization [28]. The clustering algorithm shows outstanding performance in the case of high overlap among clusters. Some examples are illustrated in Appendix E, that show the performance of the PKBD clustering on several datasets, reflecting a variety of sample sizes, dimensions and number of clusters. The addition of its graphical tools enhance the interpretation and validation of clustering results. The user-friendly dashboard application favors the research workflow, reducing the need for extensive programming knowledge and supporting reproducibility through detailed documentation and unit-tested implementations. The package has potential applications in directional statistics (e.g., modeling wind directions), computational biology (e.g., clustering gene expression profiles or protein interaction networks), environmental sciences (e.g., studying global climate patterns or ocean currents), image analysis (e.g., image segmentation with spherical projection), and social network analysis (e.g., exploring relationships between groups in high-dimensional space). To enhance reproducibility and accessibility, we have made the analyses fully available through a reproducibility research capsule via Code Ocean (https://codeocean.com/capsule/0347457/tree/v1). This allows users to easily reproduce results presented in the package in a controlled environment, further promoting transparency and broader adoption of the software.

5. Conclusions

The `QuadratiK` package offers unique methods that are not currently incorporated into any other available packages. A comparison with other tools in the literature shows that the development infrastructure of `QuadratiK` is at least as good as, if not better than, existing solutions, as illustrated in Table 1. Discussions on computation time for tuning parameter selection and uniformity test on the sphere are presented in Appendix A and Appendix B respectively. Further comparisons of the PKBD data generation algorithm that is included in our software with that included in other packages are presented in Appendix C. We have also included Appendix E, which investigates the performance of the clustering algorithm on a number of datasets of varied sample size and dimensions. The implementation of `QuadratiK` in both `R` and `Python`, together with the Dashboard application, stand out as unique features, further enhancing accessibility and usability across different disciplines.

Table A.2
Computational time for finding the optimal value of h using the `select_h` function in `R` and `Python`.

Language	Skewness Parameter (λ) for Group 2	Time (secs)					
		Mean	SD	Min	Median	IQR	Max
R	0.050	17.890	1.425	16.853	17.161	0.930	20.971
	0.500	19.280	1.194	16.924	19.534	0.363	21.026
Python	0.050	32.572	0.339	32.176	32.456	0.391	33.232
	0.500	29.175	2.406	24.469	30.205	2.614	31.408

CRediT authorship contribution statement

Giovanni Saraceno: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Raktim Mukhopadhyay:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation. **Marianthi Markatou:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

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.

Acknowledgments

The authors acknowledge help provided by Dr. Yang Chen, Associate Director, GI Statistics, Takeda Pharmaceutical Company. The last author acknowledges financial support in the form of a research award from KALEIDA Health Foundation, USA (Grant Number 82114), that supported the work of the first two authors.

Appendix A. Computational time of tuning parameter selection

The tuning parameter selection algorithm is based on grid search-based power analysis, where the power performance is investigated over a family of alternative distributions and a set of possible values of h. The smallest value of h which achieves power greater than 0.5 is chosen as the optimal value.

One of the limitations pertains to the computational time required to complete the optimal selection of the parameter, particularly in light of large samples. In most of the cases, the time needed to obtain the optimal parameter value is not onerous. As an illustration, we generated two 4-dimensional samples with 200 observations each, where the first sample comes from a standard normal distribution and the second sample follows a skew normal distribution with skewness parameter 0.5 or 0.05. The measured computational time are reported in Table A.2.

One might encounter scenarios characterized by limited sample size and weak signaling where power is not greater than 0.5, which is the threshold considered by the mid-power analysis. In such cases, the value of h that corresponds to the highest obtained power for all the combinations is selected by the algorithm. For a theoretical description of tuning parameter selection using the mid-power analysis, please see Lindsay et al. (2014).[5]

In these scenarios, it might be necessary to run the algorithm for all the combinations in the grid search of the algorithm, which can lead to a higher computational time.

[5] B. Lindsay, M. Markatou, S. Ray, Kernels, degrees of freedom, and power properties of quadratic distance goodness-of-fit tests, Journal of the American Statistical Association 109 (505) (2014) 395–410.

Table B.3
Computation time of performing the Uniformity Test on the Sphere on data of various dimensions using `QuadratiK` in `Python`. The code was executed with setting the option `n_jobs = 4`.

n	d	Computation Time in `Python` (secs)			
		Mean	SD	Median	IQR
100	3	0.064	0.007	0.064	0.011
	5	0.071	0.007	0.072	0.010
	10	0.078	0.008	0.080	0.010
1000	3	3.112	0.141	3.146	0.032
	5	3.164	0.153	3.196	0.052
	10	3.175	0.149	3.212	0.054
5000	3	69.995	3.500	71.316	1.928
	5	69.861	3.247	71.279	1.702
	10	70.062	3.453	71.405	1.517
10000	3	267.311	12.985	271.143	4.232
	5	271.614	21.070	273.707	14.820
	10	270.248	23.863	276.066	9.635

Appendix B. Computation time of uniformity test on the sphere

The `sample_hypersphere` function from the `tools` module in `QuadratiK` is used for generating data of different dimensions, which is then used for performing the uniformity test in `Python`. The computation time for performing the uniformity test is shown in Table B.3. The code was executed on an Apple MacBook Air (2020) with an M1 processor @ 3.2 GHz and 16 GB RAM, running macOS Sequoia. Additionally, the reported computation time is based on 10 runs of the algorithm on the dataset.

The reader might be interested in knowing whether datasets larger than the maximum size mentioned above can be used to perform the uniformity test. In short, the answer is *Yes*; however, a few caveats should be kept in mind.

- When datasets of size larger than the maximum size reported above are used for performing the uniformity test, the number of parallel jobs for critical value estimation should ideally be set to 1; otherwise, multiple parallel jobs will try to access large chunks of memory and will cause the system to crash. This is particularly important for the implementation in `Python` and when the code is run on an off-shelf computing machine. As a consequence of serializing the critical value estimation, the computation time will increase.
- The computation time is directly proportional to the number of iterations to be used for critical value estimation (`num_iter`).

Additionally, executing uniformity tests on large datasets can be computationally expensive in terms of both time and memory. For instance, when performing a uniformity test on a dataset (represented as X) with dimensions $50,000 \times 4$, the dot product step involving $X \cdot X^T$ results in a 50000×50000 matrix. Storing this matrix alone requires approximately 20 GB of memory, along with memory requirements for storing other values, making the computation significantly

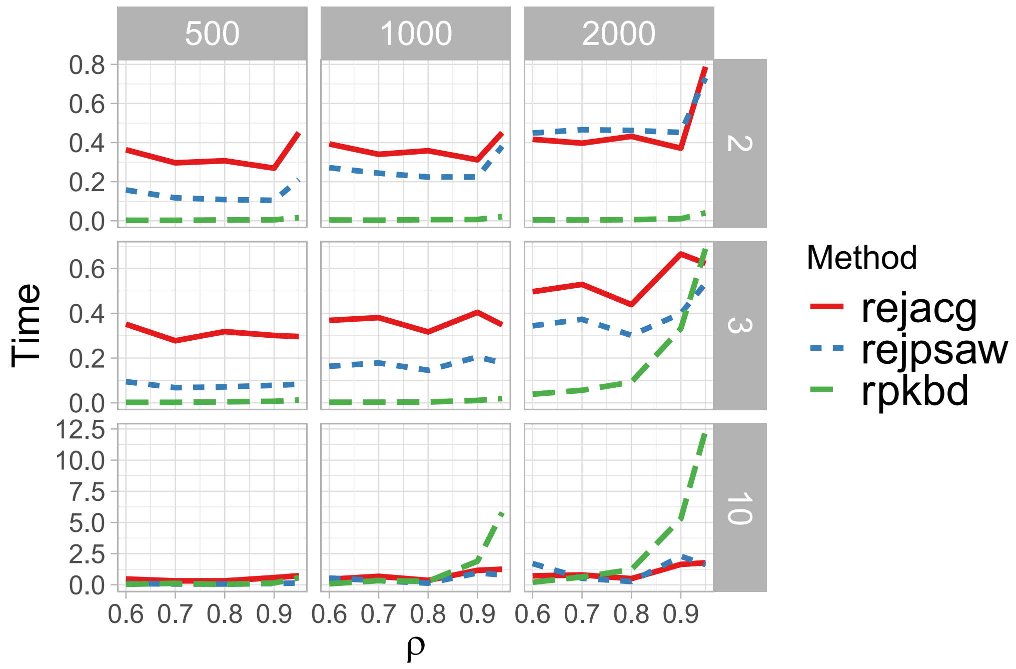


Fig. C.4. Average computational time versus ρ for the generated random samples using the function `rpkb` in `QuadratiK` and `rpkbd` in `Directional` package, for $d = 2, 3, 10$ and $n = 500, 1000, 2000$.

resource-intensive. In a commercial off-the-shelf computing machine, such an amount of RAM might not be available, necessitating the use of "swapping", where data from RAM is temporarily moved to disk storage. However, swapping is significantly slower than working directly in RAM, leading to performance bottlenecks. This makes uniformity testing on large datasets more challenging.

Appendix C. Simulation study: random sample generation from PKBD

We consider a simulation study to compare the performance of the `QuadratiK` package with the `Directional` package for generating random samples from the PKBD in R. We generated $n = 500, 1000, 2000, 5000, 10000$ random observations from PKBD with dimension $d = 2, 3, 5, 10$, concentration parameter $\rho = 0.6, 0.7, 0.8, 0.9, 0.95$ and as mean direction a vector with 1 as first element and 0 in the remaining, that is $\boldsymbol{\mu} = (1, 0, \dots, 0)$. For each configuration, we considered $N = 100$ replications where samples are generated using the function `rpkbd` of the `Directional` package and with the function `rpkb` in `QuadratiK` using the three available methods `rejvmf`, `rejacg` and `rejpsaw`. To ensure fairness, we used identical seed values for both methods to facilitate a direct comparison. Notice that `rpkbd` and `rejacg` implement the same algorithm.

Figs. C.4 and C.5 present the average computational time needed by the functions for generating the random samples versus the value of ρ, across different dimensions, for low and large sample sizes, respectively. The method `rejvmf` demonstrates the highest execution time and it is not displayed for a better visualization. The `rpkbd` function requires less computational time when the dimension is low. However, the difference is in order of seconds, or less than a second. For higher dimension, when the value of ρ increases the computational time needed by `rpkbd` increases as well, while the methods `rejacg` and `rejpsaw` are consistent with respect to ρ. Finally, the `Directional` package provides a function for estimating, via maximum likelihood, the concentration parameter ρ and the mean direction μ of the PKBD model. Note that, the clustering algorithm in `QuadratiK` can be used for the same purpose choosing the number of clusters as 1.

The detailed results and associated scripts for the simulation can be found in the GitHub repository at https://github.com/giovsaraceno/simulations-rpkb.

Appendix D. Code for generating visualization in Fig. 2

```python
import matplotlib.pyplot as plt
import numpy as np

# Plot samples on unit sphere
phi, theta = np.mgrid[0 : np.pi : 100j, 0 :
    2 * np.pi : 100j]
x = np.sin(phi) * np.cos(theta)
y = np.sin(phi) * np.sin(theta)
z = np.cos(phi)
fig = plt.figure(figsize=(5, 5))
ax = fig.add_subplot(111, projection="3d")
ax.view_init(azim=70, elev=30)
ax.plot_surface(x, y, z, color="white",
    alpha=0.8, linewidth=0)
ax.scatter(
    samples_rejvmf[:, 0],
    samples_rejvmf[:, 1],
    samples_rejvmf[:, 2],
    color="b",
    s=25,
    marker="*",
    label="rejvmf",
)
ax.scatter(
    samples_rejacg[:, 0],
    samples_rejacg[:, 1],
    samples_rejacg[:, 2],
    color="red",
    s=25,
    marker="o",
    label="rejacg",
)
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
ax.set_zlim([-1, 1])
ax.set_aspect("equal")
ax.tick_params(axis="both", labelsize=8)
plt.legend(loc="upper right", fontsize=14)
plt.tight_layout()
```

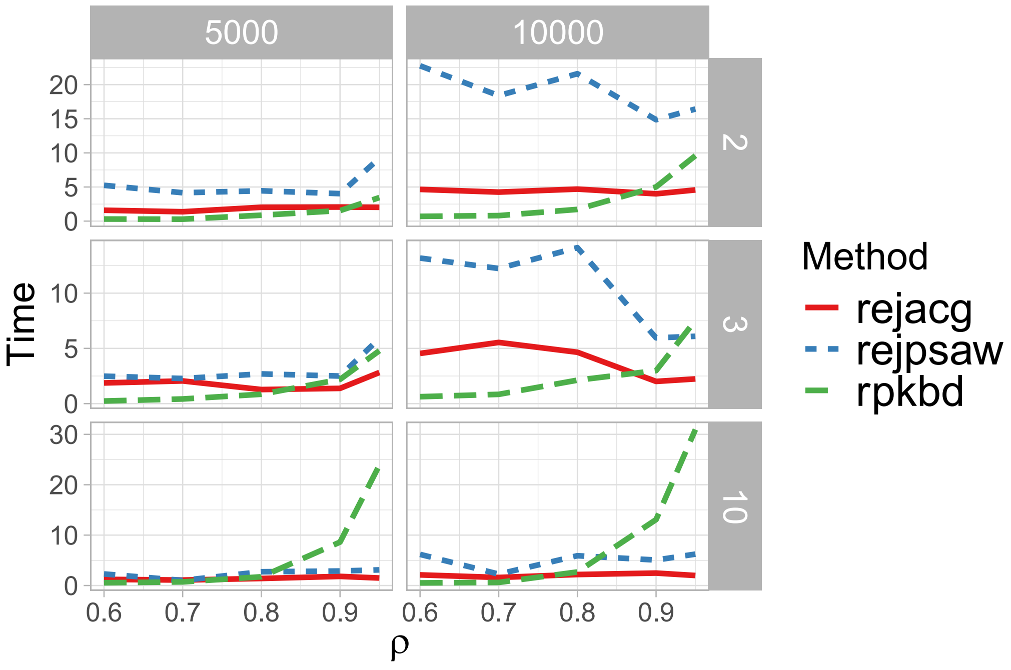


Fig. C.5. Average computational time versus ρ for the generated random samples using the function `rpkb` in `QuadratiK` and `rpkbd` in `Directional` package, for $d = 2, 3, 10$ and $n = 5000, 10000$.

Appendix E. Performance of PKBD clustering on real world datasets

Introduction

We have applied the PKBD clustering to the following datasets that reflect variety in terms of sample size, dimensions, and number of clusters. In what follows, we provide a description of the various datasets, describe in detail the preprocessing steps, and provide the results. The associated datasets and scripts can be accessed in the GitHub repository at https://github.com/rmj3197/QuadratiK-Performance.

1. Birch Dataset II (Zhang et al. (1997)[6]): This dataset contains $300,000 \times 2$ dimensional synthetic data vectors that represent one of the three patterns shown in Fig. E.6. The dataset can be found at - https://cs.joensuu.fi/sipu/datasets/.
2. 20 Newsgroups Text Dataset: This dataset is a part of the `scikit-learn`.[7] package. This data set is a collection of approximately 18,000 news posts, partitioned (nearly) evenly across 20 different newsgroups. More information of the dataset can be found at: http://qwone.com/~jason/20Newsgroups/
3. Reddit Text Dataset: This text dataset is a part of Massive Text Embedding Benchmark (MTEB).[8] This dataset contains titles of Reddit posts from subreddits. For this work, we use the subset of the "test" set of the data (data split number 13 out of 25) available from https://huggingface.co/datasets/mteb/reddit-clustering.

[6] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: A new data clustering algorithm and its applications, Data Mining and Knowledge Discovery 1,(2) (1997) 141–182.

[7] F. Pedregosa, et al. Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830

[8] N. Muennighoff, N. Tazi, L. Magne, N. Reimers, MTEB: Massive text embedding benchmark, in: A. Vlachos, I. Augenstein (Eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp.2014–2037. doi:10.18653/v1/2023.eacl-main.148

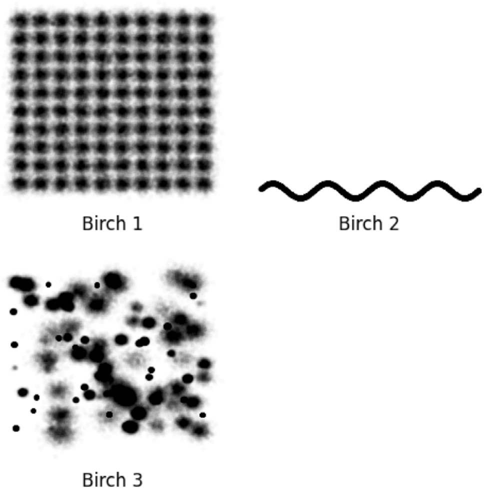


Fig. E.6. The vectors in the Birch dataset represent one of the three clusters.

4. StackExchange Text Dataset: This dataset is also part of the MTEB, and contains titles of questions posted on various StackExchanges. The clustering is performed to identify the different StackExchange communities from which the question titles are collected. In this work, we have used a subset of the "validation" set (data split number 9 out of 25) of the dataset. The dataset can be found at https://huggingface.co/datasets/mteb/stackexchange-clustering.
5. 3 Newsgroups Text Dataset: In this dataset we specifically select three groups from the 20 Newsgroups Text Dataset. The groups selected are: `omp.os.ms-windows.misc`, `rec.sport.hockey`, and `soc.religion.christian`, which represent news related to the operating system MS Windows, Hockey, and Christianity. We apply the clustering algorithm just to a subset of the data consisting of these three groups.

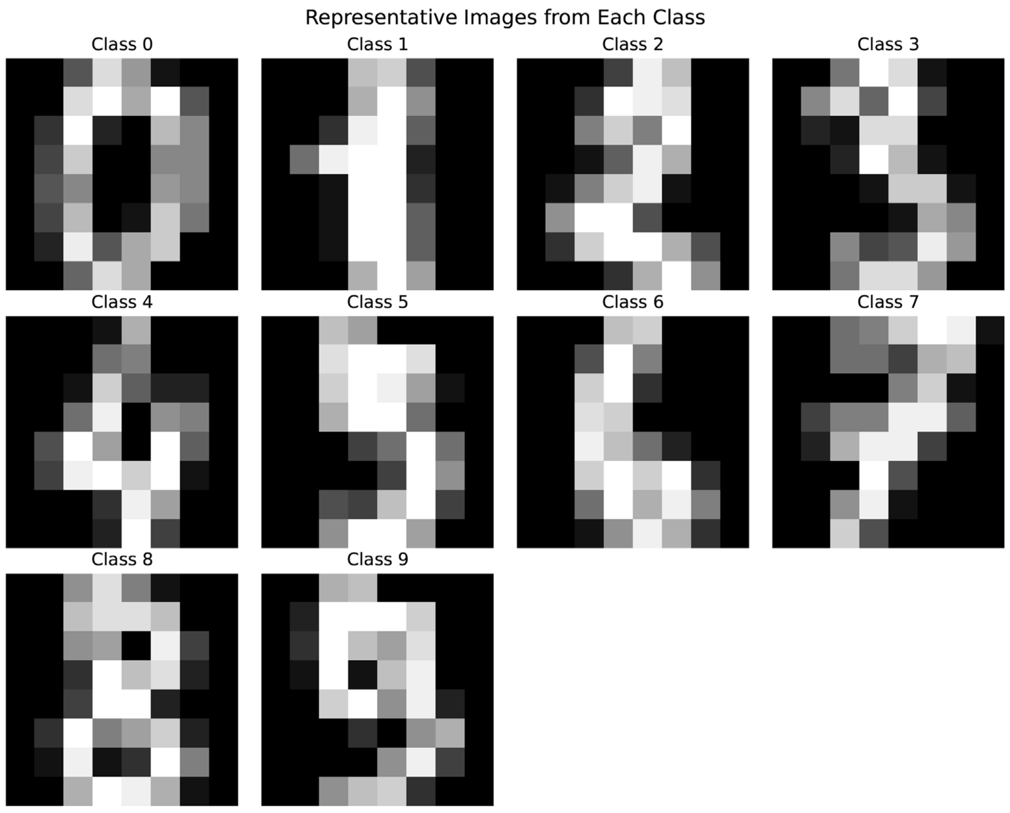


Fig. E.7. Example images for each digit from the Hand-written Digits Image Dataset.

6. arXiV Text Dataset: This dataset was introduced in He et al. (2019)[9] and is now also part of the MTEB. The dataset was originally curated for classification purposes, where 10,000 words from papers were extracted to train classifiers to predict the paper category among 11 arXiv classes. In our work, we use the "test" set of the data obtained from https://huggingface.co/datasets/mteb/ArxivClassification.
7. Hand-written Digits Image Dataset: This 8 × 8 pixels image dataset is introduced in Kaynak (1995).[10] The dataset contains images of handwritten digits from 0–9, representing the 10 classes where each class refers to a digit. It is available in the `scikit-learn` package and also available on UCI ML Repository at https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits. Representative images from the various classes are shown in Fig. E.7.
8. Detect AI Generated vs Student Generated Text Dataset: This dataset is taken from Kaggle at https://www.kaggle.com/datasets/prajwaldongre/llm-detect-ai-generated-vs-student-generated-text. The dataset consists of text written by either a student or a large language model (LLM). The goal is to identify the clusters representing the LLM-generated and the student-written text.

Data preprocessing

For clustering text datasets, we have used the workflow depicted in Fig. E.8. The text is first converted into embeddings using the `text-embedding-3small` embedding model provided by OpenAI.[11] The obtained embeddings are used as the data vector to perform the PKBD clustering. The `text-embedding-3small` embedding model used in our work supports a maximum of 8191 tokens. We removed any text containing five or fewer tokens. Additionally, we set a maximum limit of 7000 tokens. If a text exceeded this limit, it was trimmed to fit within the constraint; otherwise, it was kept as is. The tokenization was performed using the `cl100k_base` tokenizer in `tiktoken Python` library. Embeddings of different sizes are used for various datasets shown to demonstrate that the PKBD clustering algorithm can also be applied to high-dimensional data. In the case of the image dataset, every image is an 8×8 matrix with every entry of the matrix representing a pixel value. The 8 × 8 matrix is reshaped or flattened, forming a 1-dimensional array consisting of 64 values that is used for clustering.

Results

The performance of the PKBD clustering algorithm in terms of various metrics is shown in Table E.4. The code was executed on an Apple MacBook Air (2020) with an M1 processor @ 3.2 GHz and 16 GB RAM, running macOS Sequoia. Additionally, the reported computation time is based on 10 runs of the algorithm on the dataset.

Data availability

All datasets used in the main article are randomly generated using a specified seed and can be reproduced by using the same values. Regarding the appendices: for Appendix A the data is randomly generated, and the relevant scripts and data can be found in the repository: https://github.com/rmj3197/QuadratiK-Performance under the folder "Tuning

[9] J. He, L. Wang, L. Liu, J. Feng, H. Wu, Long document classification from local word glimpses via recurrent attention learning, IEEE Access 7 (2019) 40707–40718. doi:10.1109/ACCESS.2019.2907992

[10] C. Kaynak, Methods of combining multiple classifiers and their application to handwritten digit recognition, Master's thesis, Fen Bilimleri Enstitüsü (1995)

[11] OpenAI, Vector embeddings - OpenAI API, [Online; accessed 2025-03-390 01] (2025). https://platform.openai.com/docs/guides/embeddings/

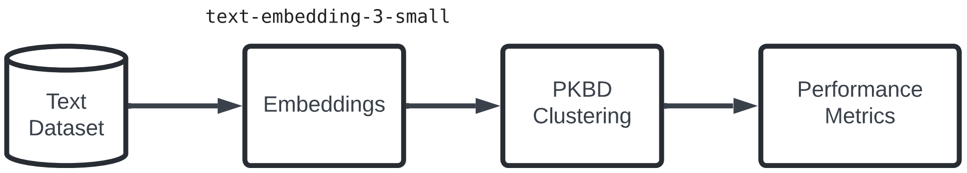


Fig. E.8. Pipeline used for clustering text datasets.

Table E.4
Performance of the PKBD-based clustering method for various datasets using different evaluation metrics. The computation time is calculated over 10 runs of the PKBD clustering algorithm using the same dataset.

Dataset	K	*n*	*d*	ARI	V Measure	Macro Precision	Macro Recall	Computation Time in Python (secs)				Computation Time in R (secs)			
								Mean	SD	Median	IQR	Mean	SD	Median	IQR
Birch Dataset II	3	300000	2	0.308	0.310	0.618	0.619	305.066	21.262	312.934	12.588	315.230	27.547	315.884	40.149
20 Newsgroups Text Dataset	20	18148	256	0.400	0.563	0.604	0.559	106.621	13.767	105.314	2.841	254.849	31.104	260.025	27.582
Reddit Text Dataset	19	9230	384	0.513	0.626	0.652	0.588	74.046	5.899	75.966	6.173	202.684	17.13	205.049	24.267
StackExchange Text Dataset	15	8661	128	0.674	0.722	0.767	0.765	25.151	1.902	25.282	2.548	47.609	8.854	47.033	15.361
3 Newsgroups Text Dataset	3	2865	256	0.909	0.862	0.969	0.969	0.761	0.173	0.746	0.126	1.380	0.763	1.102	0.075
arXiv Text Dataset	11	2500	128	0.383	0.490	0.564	0.567	10.756	1.605	10.800	1.381	12.041	1.145	12.107	1.599
Hand-written Digits Image Dataset	10	1797	64	0.656	0.737	0.804	0.788	3.431	0.559	3.472	0.461	2.216	0.297	2.116	0.330
Detect AI vs Student Generated Text Dataset	2	1102	128	1.000	1.000	1.000	1.000	0.315	0.065	0.343	0.112	0.195	0.013	0.192	0.010

The PKBD clustering algorithm is known to provide encouraging results even in the presence of high overlap between clusters (Golzy and Markatou (2020)). Cluster overlap is measured using the cosine similarity of the centroid vectors. To exemplify the performance in the presence of overlap between clusters, we present the overlap for the Birch dataset, the 3-Cluster Newsgroups dataset, and the UCI ML Handwritten Digits dataset.
Birch Dataset II: Cluster 1 vs Cluster 2: 0.763, Cluster 1 vs Cluster 3: 0.999, Cluster 2 vs Cluster 3: 0.781
3 Newsgroups Text Dataset: Cluster 1 vs Cluster 2: 0.489, Cluster 1 vs Cluster 3: 0.487, Cluster 2 vs Cluster 3: 0.399
Hand-written Digits Image Dataset: Since a large number of pairwise cosine similarity needs to be presented in this case, we simplify and conserve space by reporting the associated descriptive statistics: Min: 0.696, Q1: 0.793, Median: 0.828, Q3: 0.861, Max: 0.933.

Parameter Selection". For Appendix B, the data is randomly generated, and the relevant scripts and data can be found in the repository: https://github.com/rmj3197/QuadratiK-Performance under the folder "Uniformity Test Notebook". For Appendix C, the data is randomly generated, and the scripts used for generation can be found in the GitHub repository: https://github.com/giovsaraceno/simulations-rpkb. For Appendix E, all datasets are listed in the appendix along with their descriptions, and the processed datasets are also included in the GitHub repository: https://github.com/rmj3197/QuadratiK-Performance under the folder "Clustering Data and Notebooks".

References

[1] Faraway J, Marsaglia G, Marsaglia J, Baddeley A. goftest: Classical goodness-of-fit tests for univariate distributions. 2021, URL https://CRAN.R-project.org/package=goftest R package version 1.2-3.
[2] Gonzalez-Estrada E, Villasenor-Alva J. goft: Tests of fit for some probability distributions. 2020, URL https://CRAN.R-project.org/package=goft R package version 1.3.6.
[3] Virtanen P, Gommers R, Oliphant T, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.
[4] Vasicek O. A test for normality based on sample entropy. J R Stat Soc Ser B Stat Methodol 1976;38(1):54–9. http://dx.doi.org/10.1111/j.2517-6161.1976.tb01566.x.
[5] Song K-S. Goodness-of-fit tests based on Kullback-Leibler discrimination information. IEEE Trans Inform Theory 2002;48(5):1103–17. http://dx.doi.org/10.1109/18.995548.
[6] Fan Y. Testing the goodness of fit of a parametric density function by Kernel method. Econometric Theory 1994;10(2):316–56. http://dx.doi.org/10.1017/S0266466600008434.
[7] Friedman J, Rafsky L. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. Ann Statist 1979;7(4):697–717.
[8] Wald A, Wolfowitz J. On a test whether two samples are from the same population. Ann Math Stat 1940;11(2):147–62. http://dx.doi.org/10.1214/aoms/1177731909.
[9] Rosenbaum P. An exact distribution-free test comparing two multivariate distributions based on adjacency. J R Stat Soc Ser B Stat Methodol 2005;67(4):515–30.
[10] Rizzo M, Szekely G. energy: E-statistics: Multivariate inference via the energy of data. 2022, URL https://CRAN.R-project.org/package=energy R package version 1.7-11.
[11] Rizzo M, Székely G. Energy distance. WIREs Comput Stat 2016;8(1):27–38. http://dx.doi.org/10.1002/wics.1375.
[12] Gretton A, Borgwardt K, Rasch M, Schölkopf B, Smola A. A kernel two-sample test. J Mach Learn Res 2012;13(1):723–73.
[13] Scholz F, Zhu A. kSamples: K-sample rank tests and their combinations. 2019, URL https://CRAN.R-project.org/package=kSamples R package version 1.2-9.
[14] Kruskal W, Wallis W. Use of ranks in one-criterion variance analysis. J Amer Statist Assoc 1952;47(260):583–621. http://dx.doi.org/10.1080/01621459.1952.10483441.
[15] Sidak Z, Sen P, Hajek J. Theory of rank tests. Elsevier; 1999.
[16] Hothorn T, Hornik K, van de Wiel M, Zeileis A. Implementing a class of permutation tests: The coin package. J Stat Softw 2008;28(8):1–23. http://dx.doi.org/10.18637/jss.v028.i08.
[17] Panda S, Shen C, Perry R, Zorn J, Lutz A, Priebe C, et al. Universally consistent K-sample tests via dependence measures. 2024, arXiv:1910.08883. URL https://arxiv.org/abs/1910.08883.
[18] Panda S, Palaniappan S, Xiong J, Bridgeford E, Mehta R, Shen C, et al. hyppo: A multivariate hypothesis testing python package. 2024, arXiv:1907.02088. URL https://arxiv.org/abs/1907.02088.
[19] Agostinelli C, Lund U. R package circular: Circular statistics (version 0.5-1). 2024, URL https://CRAN.R-project.org/package=circular.

[20] Fernández-Durán J, Gregorio-Domínguez M. CircNNTSR: An R package for the statistical analysis of circular, multivariate circular, and spherical data using nonnegative trigonometric sums. J Stat Softw 2016;70(6):1–19. http://dx.doi.org/10.18637/jss.v070.i06.

[21] Lindsay B, Markatou M, Ray S. Kernels, degrees of freedom, and power properties of quadratic distance goodness-of-fit tests. J Amer Statist Assoc 2014;109(505):395–410.

[22] Golzy M, Markatou M. Poisson kernel-based clustering on the sphere: Convergence properties, identifiability, and a method of sampling. J Comput Graph Statist 2020;29(4):758–70.

[23] Ding Y, Markatou M, Saraceno G. Poisson kernel-based tests for uniformity on the d-dimensional sphere. Statist Sinica 2023. http://dx.doi.org/10.5705/ss.202022.0347.

[24] Markatou M, Saraceno G. A unified framework for multivariate two-sample and k-sample kernel-based quadratic distance goodness-of-fit tests. 2024, arXiv:2407.16374. URL https://arxiv.org/abs/2407.16374.

[25] Luna C, Moya AR, Luna JM, Ventura S. StaTDS library: Statistical tests for Data Science. Neurocomputing 2024;595:127877. http://dx.doi.org/10.1016/j.neucom.2024.127877, URL https://www.sciencedirect.com/science/article/pii/S0925231224006489.

[26] Golzy M, Rosenb G, Krusec R, Hooshmandd K, Mehrc D, Murrayb K. Holistic assessment of quality of life predicts survival in older patients with bladder cancer. Oncology 2023;17:141–9. http://dx.doi.org/10.1016/j.urology.2022.12.036.

[27] Strelnikoff S, Jammalamadaka A, Warmsley D. Causal maps for multi-document summarization. In: 2020 IEEE international conference on big data (big data). 2020, p. 4437–45. http://dx.doi.org/10.1109/BigData50022.2020.9377731.

[28] Strelnikoff S, Jammalamadaka A, Warmsley DM. Method and system for event prediction via causal map generation and visualization. 2024, US Patent 11, 907, 307.