Journal of
open research software

GTST: A Python Package for Graph Two-Sample Testing

SOFTWARE
METAPAPER

RAGNAR L. GUDMUNDARSON
GARETH W. PETERS
*Author affiliations can be found in the back matter of this article

]u[ubiquity press

ABSTRACT

The GTST package is a python package for performing graph sample testing. The test infers whether two samples of graphs were generated from the same probability distribution or not. It is a very general framework as it allows comparison between binary, weighted, directed, node-labelled, node attributed and edge-labelled graphs. Up until now, there is no package which offers graph sample testing even though the problem is often encountered in various fields such as risk management, social sciences and molecular science. The flexibility of the test comes from so-called graph kernels which allow one to measure similarities between complex graph data. The difference between the two samples is quantified using an empirical estimate of the maximum mean discrepancy which is a distance on the space of probability measures. Along with testing of graph samples, the package offers various graph kernels, some of which have not been readily available before.

CORRESPONDING AUTHOR:
Ragnar L. Gudmundarson
Centre for Networks & Enterprise Excellence, Edinburgh Business School, Heriot Watt University, UK
ragnarlevigud@gmail.com

KEYWORDS:
graph two sample testing; MMD; graph kernel; kernel

TO CITE THIS ARTICLE:
Gudmundarson RL, Peters GW 2024 GTST: A Python Package for Graph Two-Sample Testing. *Journal of Open Research Software,* 12: 6. DOI: https://doi.org/10.5334/jors.478

(1) OVERVIEW

INTRODUCTION

It is increasingly common to obtain sampled data in the form of graph or network realisations. This is either through construction of graph structures as summary statistics for multivariate spatial-temporal data or directly as a graph-network data set from say a social network realisation or the like. In such settings in practice, one often needs to draw inferences about two samples of such graphs or networks. That is, to test whether the collections of graphs/networks in one sample were generated from the same distribution as the collection of graphs/networks in the second sample.

Kernel methods have proven to be useful in pattern recognition tasks such as classification and can be further extended as an inference procedure to two-sample hypothesis testing on structured data. The method embeds the graphs into a reproducing kernel Hilbert space (RKHS) via a feature map which is then extended further to the embedding of a probability distribution. The two-sample null hypothesis is that the generating mechanism behind the two samples is the same and the test statistic, which is called the maximum mean discrepancy (MMD), is the largest distance between the means of the two sample embeddings [7].

Graph kernels are already well established and widely used for solving classification tasks on graphs and can further be used to compare samples of graphs and to perform graph screening [14]. They provide a very flexible way of comparing graphs as they exist for a wide range of different graph structures, for example, weighted, directed, labelled, and attributed graphs. Their performance depends on their expressiveness, that is, their ability to distinguish non-isomorphic graphs. The difficulty of distinguishing two samples of graphs varies strongly based on the type of graphs.

Graph two-sample hypothesis testing is a problem that frequently arises in various disciplines, for example in bio informatics [1], community detection [6], and risk management [3]. Graph two-sample hypothesis have mostly been performed by using graph statistics such has the degree centrality and shortest paths. Although these methods can often give good performances they fail to take into account various attributes that are often present in real graphs such as node labels, edge labels, node attributes and edge weights. When the kernel two-sample hypothesis testing was introduced [7] a flood gate opened to allow for testing of such attributes and therefore providing a flexible way of performing two-sample hypothesis testing. Luckily, there also exists a vast literature on graph kernels [11, 14]. Until now, there is no package which allows one to estimate graphs from real valued data matrices and perform hypothesis testing in a flexible manner. The package GTST provides three functionalities

1. Functions to estimate the two-sample hypothesis test statistic;
2. Functions to calculate different graph kernels; and
3. A function to estimate graphs from data matrices.

The package allows other network science researches who may not be programmers to use the MMD testing framework.

There exists a python package called GraKel [17] which is dedicated to calculating various graph kernels. The package is very user-friendly so the GTST user can use all graph kernels available in the Grakel package. The GTST package extends the choice of graph kernels available for use in graph two-sample testing, by also developing a collection of kernels not available in GraKel. Such as, the fast random walk kernels which is based on ideas from [10] along with an additional fast random walk kernel for edge-labelled graphs; The Wasserstein Weisfeiler-Lehman Graph kernel [18] whose original code was adjusted for the package needs. The Deep Graph kernel [19], and the graph neural tangent kernel [4] whose original code was adjusted for the package needs. The MONK estimator, which is a robust estimator of the MMD, was developed by MONK [12] and they do provide the code online and in a packaged environment. However, we have adjusted the code slightly to allow for robust comparing of samples of different sizes. The MMDGraph then estimates the *p*-value of test by using a bootstrap or a permutation sampling scheme. The package also allows for estimating graphs using sklearn's graphical lasso [2]. Additional preprocessing can be done by using the nonparanormal transform [13]. The best graph is found by using the EBIC criterion [15]. GTST assumes that the graphs passed are a networkx object [9]. One can additionally use pre-computed kernels to perform tests.

IMPLEMENTATION AND ARCHITECTURE

Let $G(V, S)$ denote a graph with vertex set V and edge set S. In the two-sample testing of graph-valued data, we assume we are given two sets of samples/observations that comprise collections of graph-valued data $\{G_1, \ldots, G_n\}$ and $\{G'_1,\ldots,G'_{n'}\}$ where $G_i, G'_j \in \Omega, \quad \forall i, j$. The graphs in the two samples are all generated independently from two probability spaces $(\Omega, \mathcal{F}, \mathbb{P})$ and $(\Omega, \mathcal{F}, \mathbb{Q})$, and the goal is to infer whether $\mathbb{P} = \mathbb{Q}$. We note that in this general testing framework the vertex sets and edge-sets do not have to be equal, but they can be common if it is desirable for the application. This is therefore a very general testing framework. In the simplest case, where we assume both sets of sample graphs come from a common set of vertices, then the sample space Ω contains all possible edges that can occur in a graph G, that is $\Omega = \{(v_1,v_2),\ldots,(v_1,v_{|V|}),(v_2,v_1),\ldots,(v_{|V|},v_{|V|-1})\}$.[1] As the sample space is discrete we can define the σ-algebra as the power set of Ω, namely, $\mathcal{F} = \mathcal{P}(\Omega)$. The probability function $\mathbb{P} : \mathcal{F} \mapsto [0,1]$ then defines the probability of

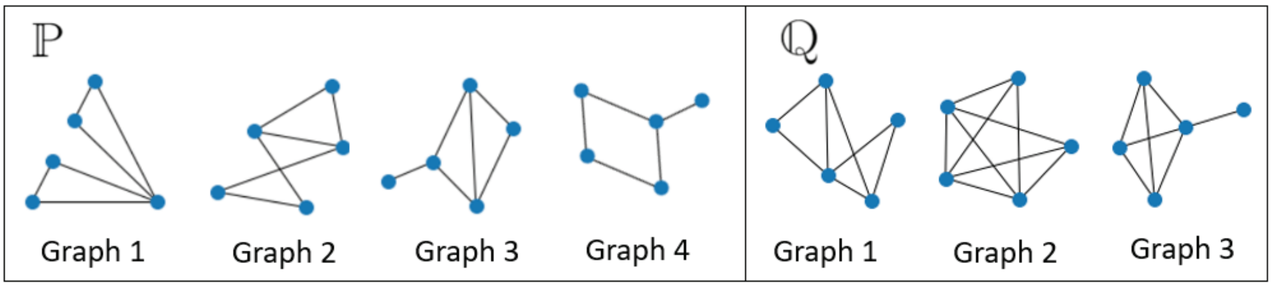

Figure 1 The graph two sample testing scenario. Here we have observed 4 graphs from $\mathbb{P}$ and 3 graphs from $\mathbb{Q}$. The sample space of $\mathbb{P}$ and $\mathbb{Q}$ is the same ($\binom{5}{2}$ possible edges).

obtaining a certain graph in the sample set of graph-valued data. As an example we can define for instance a population distribution to be uniform

$$\mathbb{P}(G(V,S)) = \frac{1}{\binom{M}{|S|}}$$

where $M = \binom{|V|}{2}$ is the total number of possible edges and $G(V, S)$ is a graph with $|V|$ vertices and $|S|$ number of edges. This setting is illustrated in Figure 1.

Now, returning to the concept of two-sample testing for graph-valued data. The goal is to infer whether the two samples of graphs are generated according to the same distribution. This involves developing a statistical test $T(\{G\}_{i=1}^{n}, \{G'\}_{i=1}^{n'})$ to determine from the population samples whether there is sufficient evidence to reject the null hypothesis that both population distributions generating the two samples of graphs are equivalent, where $T(\{G\}_{i=1}^{n}, \{G'\}_{i=1}^{n'}) : \{G\}_{i=1}^{n} \times \{G'\}_{i=1}^{n'} \mapsto \{0,1\}$ is a function that distinguishes between the null hypothesis and the alternative hypothesis:

$$\begin{aligned} H_0 :&\quad \mathbb{P} = \mathbb{Q} \\ H_1 :&\quad \mathbb{P} \neq \mathbb{Q}. \end{aligned}$$

The test statistic used in this case is the largest distance between the expectation of some function w.r.t. to the two probability distributions. Let $\mathcal{H}_B$ be a class of functions $f : \Omega \to R$. The maximum mean discrepancy (MMD) is defined as:

$$\text{MMD}[\mathcal{H}_B, \mathbb{P}, \mathbb{Q}] := \sup_{f \in \mathcal{H}_B} \left(E_{G \sim \mathbb{P}}[f(G)] - E_{G' \sim \mathbb{Q}}[f(G')] \right).$$

Where the class of function, $\mathcal{H}_B$, is the unit ball in a RKHS, then the squared population MMD becomes kernalized:

$$\text{MMD}^2[\mathcal{H}_1, \mathbb{P}, \mathbb{Q}] = E_{\mathbb{P},\mathbb{P}}[k(G,G)] - 2E_{\mathbb{P},\mathbb{Q}}[k(G,G')] + E_{\mathbb{Q},\mathbb{Q}}[k(G',G')],$$

where k is some graph kernel. The kernel matrix plays a central role and is defined as:

$$K_{ij} = k(G_i, G_j),$$

where $\{G_i\}_{i=1}^{2n}$ is the data. Note, both sampels are included in the data. In the context of two sample graph testing, we have two data sources, sample 1 of graphs assumed drawn from P and sample 2 of graphs assumed drawn from Q. It can be good to order the kernel matrix such that **K** has a block structured as follows:

$$K = \begin{bmatrix} K_{\mathbb{P}\mathbb{P}} & K_{\mathbb{P}\mathbb{Q}} \\ K_{\mathbb{Q}\mathbb{P}} & K_{\mathbb{Q}\mathbb{Q}} \end{bmatrix}, \tag{1}$$

where $\mathbf{K}_{\mathbb{P}\mathbb{Q}}$ is the Kernel function evaluated at data points within the sample coming from the unknown distributions $\mathbb{P}$ and $\mathbb{Q}$. $\mathbf{K}_{\mathbb{P}\mathbb{P}}$, $\mathbf{K}_{\mathbb{Q}\mathbb{P}}$ and $\mathbf{K}_{\mathbb{Q}\mathbb{Q}}$ are defined analogously. Note $K_{\mathbb{P}\mathbb{Q}} = K_{\mathbb{Q}\mathbb{P}}^T$.

Different estimates of MMD^2 are:

UNBIASED ESTIMATE

An unbiased estimate of MMD^2 for n and n' samples of graphs is given by:

$$\begin{aligned} \widehat{\text{MMD}}_u^2[\mathcal{H}_B, \mathbb{P}, \mathbb{Q}] = &\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(G_i, G_j) + \frac{1}{n'(n'-1)} \sum_{i=1}^{n'} \sum_{j \neq i}^{n'} k(G'_i, G'_j) \\ &- \frac{2}{nn'} \sum_{i=1}^{n} \sum_{j=1}^{n'} k(G_i, G'_j). \end{aligned} \tag{2}$$

BIASED ESTIMATE

A biased estimate of MMD^2 for n and n' samples of graphs is given by:

$$\begin{aligned} \widehat{\text{MMD}}_b^2[\mathcal{H}_B, \mathbb{P}, \mathbb{Q}] = &\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(G_i, G_j) + \frac{1}{n'^2} \sum_{i=1}^{n'} \sum_{j=1}^{n'} k(G'_i, G'_j) \\ &- \frac{2}{nn'} \sum_{i=1}^{n} \sum_{j=1}^{n'} k(G_i, G'_j). \end{aligned} \tag{3}$$

UNBIASED LINEAR TIME ESTIMATE

An unbiased can be computed in $O(n)$ time. Assume that $n = n'$ and define $n_2 = \lfloor n/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function, then the linear estimate is computed as:

$$\begin{aligned} \widehat{\text{MMD}}_l^2[\mathcal{H}_B, \mathbb{P}, \mathbb{Q}] = &\frac{1}{n_2} \sum_{i=1}^{n_2} \big(k(G_{2i-1}, G_{2i}) + k(G'_{2i-1}, G'_{2i}) - k(G_{2i-1}, G'_{2i}) \\ &- k(G_{2i}, G'_{2i-1}) \big), \end{aligned}$$

while $\widehat{\text{MMD}}_l^2$ has higher variance than $\widehat{\text{MMD}}_u^2$, it is computationally much more appealing.

ROBUST ESTIMATE

Consider the partition $\{S_q\}_{q\in 1:Q}$ of $|S_q| = n/Q$ partitions. By the representer theorem [16] we can express a function f within the RKHS space as:

$$f = \sum_i a_i k(\cdot, x_i) + \sum_i b_i k(\cdot, y_i).$$

A robust MMD estimator is found by solving:

$$\max_{c:c^T K c \le 1} med_{q\in 1:Q}\left\{\frac{1}{|S_q|}[1_q, -1_q]Kc\right\},$$

where $c = [a,b] \in \mathbb{R}^{2n}, K = [K_{\mathbb{PP}}, K_{\mathbb{PQ}}; K_{\mathbb{QP}}, K_{\mathbb{QQ}}] \in \mathbb{R}^{2n\times 2n}$, where $\mathbf{K}_{\mathbb{PQ}}$ is the Kernel function evaluated at data points within the sample coming from the unknown distributions $\mathbb{P}$ and $\mathbb{Q}$, and $\mathbf{1}_q \in \mathbb{R}^n$ is an indicator vector of block q. For more details see [12, 8].

The package includes graph kernels such as:

- **Random walk kernels** which can be used on weighted, directed, undirected, and bipartite graphs with node labels, node attributes, edge labels, and edge attributes.
- **Deep graph kernels** can be used on node labelled binary graphs.
- **Graph neural tangent kernels** can be used on binary graphs with node labels/attributes.
- **Wasserstein Weisfeiler-Lehman graph kernels** can be used on undirected graphs with node labels and can be used on node-attributed graphs with the right embedding scheme as well.

Other graph kernels can be utilized via the graph kernel library GraKel [17] which is dedicated to calculating various graph kernels. The package is very user-friendly so the GTST user can use all graph kernels available in the Grakel package. For more details on various graph kernels see [8, 17, 14].

EXAMPLE

The workflow is as follows: 1) Use two data arrays to estimate two sequences/samples of graphs using the graphical lasso [5]. This step can be skipped if the practitioner already has the samples of graphs in a networkx format [9]. 2) Select a graph kernel. 3) Select an estimator of the MMD or try multiple estimators to obtain a p-value. A quick example for the random walk kernel is:

```
import numpy as np
import networkx as nx
import GTST

# Generate random samples
n1 = n2 = 50 # sample sizes
# generate two samples
g1 = [nx.fast_gnp_random_graph (30,0.3) for
_ in range (n1)]
g2 = [nx.fast_gnp_random_graph (30,0.4) for
_ in range (n2)]

# Random Walk, r is number of eigen-pairs, c
is the discount constant
MMD_out = GTST.MMD()
MMD_out.fit(G1 = g1,
            G2 = g2,
            kernel = 'RW_ARKU_PLUS',
            mmd_estimators = 'MMD_u',
            r = 6,
            c = 0.001)
print (f" RW_ARKU_plus {MMD_out.p_values}")
```

For more exampels please see https://github.com/ragnarlevi/GTST.

QUALITY CONTROL

The requirements for running test for GTST is listed in the *requirements_dev.txt* in the Github page. The tests involve testing if kernels matrices are positive semi definite and whether the kernels, test statistic, and permutation method for the p-value are able to reject the null when the null is "extremely" false using known random data sets. Once the required testing packages have been installed, the tests can be performed by running the command:

```
pytest
```

in the root folder, which will take around 15 minutes. This will generate a coverage report which can be found in the *htmlcov* directory. To view it run:

```
cd htmlcov python - m http.server
```

and open the localhost link (something like http://localhost:8000/) in a browser.

To test GTST in a clean environment for all python versions from 3.7–3.10, we use Tox. This can be achieved by running

```
tox
```

in the root directory. Note that this takes significantly longer to run, so is best performed as a final check. Whenever the code is pushed to the remote repository, the Tox test suite is automatically run using GitHub actions. To investigate this process, consult the file found at *.github/workflows/tests.yml* on the github page. The output of the report can be found in the github actions tab.

A coverage report was made by using coveralls locally, but can be found by clicking the coverage shield on the github page.

(2) AVAILABILITY

OPERATING SYSTEM

This package can be run on any operating system where python can be run (GNU/Linux, Mac OSX, Windows).

PROGRAMMING LANGUAGE

python 3.7+

ADDITIONAL SYSTEM REQUIREMENTS

None

DEPENDENCIES

Packages that are required are scipy>=1.6.1, scikit-learn>=1.0.2, tqdm>=4.59.0, numpy>=1.20.1, networkx>=2.5, POT>= 0.8.2.

The Grakel package is optional but is recommended to be installed so an extra suite of graph kernels can be used.

SOFTWARE LOCATION

Archive

Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.8037197
Licence: MIT
Publisher: Ragnar Leví Guðmundarson
Version published: 0.0.6
Date published: 14/06/23

Code repository

Name: Github
Persistent identifier: https://github.com/ragnarlevi/GTST
Licence: MIT
Date published: 11/11/22

LANGUAGE

English

(3) REUSE POTENTIAL

The code was originally used in the paper [8] to compare pairwise asset return process relationships to study and understand risk and return in portfolio management practice. This allows one to statistically test for significance of any detected differences in portfolio diversification between any portfolio investment strategy when applying differing investment screening criteria or optimal investment strategies. We further remark that this package can further be used in other fields other than portfolio comparison. For example, the package allows, in a straight forwards manner, to compare different community structures, to detect changes in communities, to detect change-point events, to test for differences in traffic networks, and to compare ego-networks of entities.

Every function has docstrings to ensure clarity. Examples for all graph kernels, some Grakel kernels and estimators can be found on the Github page, along with an example of the graph estimation functionality. GTST is released under the MIT license and welcomes any contributions. We encourage users to submit feedback using GitHub issue tracker, or by emailing Ragnar.

NOTE

1 We are assuming that a node can not be connected to itself.

ACKNOWLEDGEMENTS

We acknowledge the support from Professor Dimitris Christopoulos and Dr. George Tzougas. This work was supported by Edinburgh Business School, Heriot-Watt. We would also like to thank the reviewers for valuable feedback.

COMPETING INTERESTS

The authors have no competing interests to declare.

LIST OF CONTRIBUTORS

RLG wrote the software package, developed the methodology, wrote the paper, and tested the software. GWP guided the problem formulations and solutions conceptually.

AUTHOR AFFILIATIONS

Ragnar L. Gudmundarson orcid.org/0000-0002-5341-9206
Centre for Networks & Enterprise Excellence, Edinburgh Business School, Heriot Watt University, UK
Gareth W. Peters orcid.org/0000-0003-2768-8979
Department of Statistics & Applied Probability, University of California, Santa Barbara, US

REFERENCES

1. **Bassett DS, Bullmore E, Verchinski BA, Mattay VS, Weinberger DR, Meyer-Lindenberg A.** Hierarchical organization of human cortical networks in health and schizophrenia. *The Journal of Neuroscience*. 2008; 28(37): 9239–9248. DOI: https://doi.org/10.1523/JNEUROSCI.1929-08.2008
2. **Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G.**

API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013; 108–122. URL: https://scikit-learn.org/.

3. **Carreno JG, Cifuentes R.** Identifying complex core-periphery structures in the interbank market. *Journal Of Network Theory In Finance*; 2017. DOI: https://doi.org/10.21314/JNTF.2017.035
4. **Du SS, Hou K, Salakhutdinov RR, Poczos B, Wang R, Xu K.** Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In: Wallach H, Larochelle H, Beygelzimer A, dAlché-Buc F, Fox E, Garnett R, (eds.), *Advances in Neural Information Processing Systems*. 2019; 32: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/663fd3c5144fd10bd5ca6611a9a5b92d-Paper.pdf.
5. **Friedman J, Hastie T, Tibshirani R.** Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*. 2007; 9(3): 432–441. DOI: https://doi.org/10.1093/biostatistics/kxm045
6. **Girvan M, Newman MEJ.** Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*. 2002; 99(12): 7821–7826. DOI: https://doi.org/10.1073/pnas.122653799
7. **Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola A.** A kernel two-sample test. *Journal of Machine Learning Research*. 2012; 13(25): 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html.
8. **Gudmundarson RL, Peters G.** Assessing portfolio diversification via two-sample graph kernel inference. a case study on the inuence of ESG screening. *SSRN Electronic Journal*; 2023. DOI: https://doi.org/10.2139/ssrn.4348306
9. **Hagberg A, Conway D.** *Networkx: Network analysis with python*; 2020. URL: https://networkx.github.io.
10. **Kang U, Tong H, Sun J.** Fast random walk graph kernel. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics; 2012. DOI: https://doi.org/10.1137/1.9781611972825.71
11. **Kriege NM, Johansson FD, Morris C.** *A survey on graph kernels*. 2020; 5. DOI: https://doi.org/10.1007/s41109-019-0195-3
12. **Lerasle M, Szabo Z, Mathieu T, Lecue G.** {MONK} outlier-robust mean embedding estimation by median-of-means. *PMLR*. 2019; 97: 3782–37. URL: http://proceedings.mlr.press/v97/lerasle19a.html.
13. **Liu H, Lafferty J, Wasserman L.** The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*. 2009; 10(80): 2295–2328. URL: http://jmlr.org/papers/v10/liu09a.html.
14. **Nikolentzos G, Siglidis G, Vazirgiannis M.** Graph kernels: A survey. *J. Artif. Int. Res*. 2022; 72: 943–1027. DOI: https://doi.org/10.1613/jair.1.13225
15. **Orzechowski P, Moore JH.** Ebic: A scalable biclustering method for large scale data analysis. *GECCO 2019 Companion – Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion*. 2019; 31–32. DOI: https://doi.org/10.1145/3319619.3326762
16. **Schölkopf B, Herbrich R, Smola AJ.** A generalized representer theorem. In: Helmbold D, Williamson B (eds.), *Computational Learning Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg. 2001; 416–426. DOI: https://doi.org/10.1007/3-540-44581-1_27
17. **Siglidis G, Nikolentzos G, Limnios S, Giatsidis C, Skianis K, Vazirgiannis M.** Grakel: A graph kernel library in python. *Journal of Machine Learning Research*. 2020; 21(54): 1–5. URL: https://github.com/ysig/GraKeL.
18. **Togninalli M, Ghisu E, Llinares-López F, Rieck B, Borgwardt K.** *Wasserstein Weisfeiler-Lehman Graph Kernels*. NIPS '19, Curran Associates Inc., Red Hook, NY, USA; 2019. URL: https://proceedings.neurips.cc/paper/2019/file/73fed7fd472e502d8908794430511f4d-Paper.pdf.
19. **Yanardag P, Vishwanathan SVN.** *Deep graph kernels*. Association for Computing Machinery. 2015; 1365–1374. DOI: https://doi.org/10.1145/2783258.2783417

TO CITE THIS ARTICLE:
Gudmundarson RL, Peters GW 2024 GTST: A Python Package for Graph Two-Sample Testing. *Journal of Open Research Software,* 12: 6. DOI: https://doi.org/10.5334/jors.478

Submitted: 23 July 2023 **Accepted:** 21 February 2024 **Published:** 18 March 2024

COPYRIGHT:
© 2024 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See http://creativecommons.org/licenses/by/4.0/.

Journal of Open Research Software is a peer-reviewed open access journal published by Ubiquity Press.