Getting Started With DataSimilarity: Quantifying Similarity of Datasets and Multivariate Two- and k-Sample Testing

Marieke Stolte
TU Dortmund University and LMU Munich

Luca Sauer
TU Dortmund University

Jörg Rahnenführer
TU Dortmund University

Andrea Bommert
TU Dortmund University

Abstract

Quantifying the similarity of two or more datasets is a common task in various applications of statistics and machine learning, including two- or k-sample testing and meta- or transfer learning. The **DataSimilarity** package contains a variety of methods for quantifying the similarity of datasets. The package includes 40 methods of which 17 are implemented for the first time in R. The remaining are wrapper functions for methods with already existing implementations that unify and simplify the various input and output formats of different methods and bundle the methods of many existing R packages in a single package. In this vignette, we show the basic workflow for using the package.

Keywords: dataset similarity, two-sample testing, multi-sample testing.

1. Introduction

The challenge of quantifying how similar two or more datasets are arises in various contexts where two or more datasets should be compared. This could be in the context of transferring results of a prediction model from one dataset to another, as well as for assessing how close simulated data is to a real-world dataset. The most common usage is for two- or k-sample testing. Formally, the two-sample problem is defined as the testing problem

$$H_0 : F_1 = F_2 \text{ vs. } H_1 : F_1 \neq F_2. \tag{1}$$

A two-sample test, therefore, can be used to check whether the underlying distributions of two datasets coincide. Analogously, the k-sample problem is defined as

$$H_0 : F_1 = F_2 = \cdots = F_k \text{ vs. } H_1 : \exists i \neq j \in \{1, \ldots, k\} : F_i \neq F_j,$$

for k distributions $F_1, \ldots, F_k$.

Many different methods are proposed in the literature for quantifying the similarity of two or more datasets, and most of these define a two- or k-sample test. In this package, a subset of these methods are implemented, which were selected as relevant from a literature review

(Stolte, Kappenberg, Rahnenführer, and Bommert 2024). For more details on the methods and their selection, see the 'Details' vignette. In the following, the basic steps for using the **DataSimilarity** package are explained using real-world example datasets with different characteristics with regard to the scale level, number of datasets, and presence of a target variable in each dataset. Moreover, it is demonstrated how a method from the package can be chosen. For that, both theoretical properties of the method (Stolte *et al.* 2024) and their empirical performance in simulation studies (Stolte, Rahnenführer, and Bommert 2026a,b) are considered.

2. Workflow

In the following, the typical workflow for working with the package is explained.

There are two different use cases with different workflows.

a) We already know which method to apply to our dataset comparison at hand.

b) We have two (or more) datasets that we want to compare, but we do not have a specific method in mind.

In case a), the workflow for using the package would be to find the corresponding function for the method and apply it to the data. The full list of methods can also be found in the 'Details' vignette as well as in the `method.table` dataset. Some examples for working with the package are given in Section 3. There, we differentiate six cases with regard to the applicability of the selected methods. These are summarized in Table 1. In Section 3, one data example per case is shown and one corresponding applicable method is explained and demonstrated.

Scenario no.	No.datasets	Scale level	Target variable
1	$k = 2$	Numeric	No
2	$k \geqslant 2$	Numeric	No
3	$k = 2$	Numeric	Yes
4	$k = 2$	Categorical	No
5	$k \geqslant 2$	Categorical	No
6	$k = 2$	Categorical	Yes

Table 1: Overview of considered cases for applicability of the dataset similarity methods. If present, the target variable included in each dataset has to be a categorical variable.

In case b), the package can also be used as a tool for finding an appropriate method. This depends on the dataset characteristics. For a quick overview of which methods are applicable in which cases, see also the help page `?DataSimilarity`. Here, we distinguish between numeric and categorical data, the number of datasets (two or more than two), and whether or not the datasets include a target variable. We demonstrate how to find and apply a method for different types of datasets in the following. The general workflow for case b) is summarized in Table 2. The proposed COMPARE procedure (**C**haracteristics of datasets, **O**bjective of comparison, **M**ethod requirements, **P**roduction of method list, **A**ssessment of computational complexity, **R**anking of methods, **E**stimation and testing) for finding the best-suited method

	Step	Description
1.	Characteristics of datasets	Evaluate number of datasets, scale level of variables, presence of target variable, dimensions of datasets
2.	Objective of comparison	Is the focus mainly on a test decision, or is the (dis)similarity value itself of interest and should be interpretable?
3.	Method requirements	Are additional properties of the method regarding invariances, metric properties, or consistency of a test of relevance?
4.	Production of method list	Call `findSimilarityMethod()` with the criteria identified in steps 1.–3.
5.	Assessment of computational complexity	Are the identified methods feasible with regard to their runtime and memory consumption?
6.	Ranking of methods	Compare the remaining methods regarding their performance in simulation studies and select the best
7.	Estimation and testing	Apply the chosen method(s)

Table 2: Step-by-step procedure for choosing the best method for a given application.

is described in more detail in the following. It can be seen as a step-by-step guide for first specifying all requirements that can then be passed to `findSimilarityMethod()`. This returns a list of eligible methods. The remaining steps then consist of finding the best method from this list and applying it to the datasets. In Section 4, each step is described and the full procedure is demonstrated for one application example of comparing strategies for creating synthetic datasets.

3. Examples for case a): We already know which method to apply

In the following, we present examples how to apply specific methods for six special data situations with different characteristics. For method descriptions, see the Details vignette. The methods for the examples here are chosen for demonstration mainly based on their interpretability and simplicity. For guidance on choosing methods for a given application, refer to the following section. For using any of the methods, we first need to attach the package.

```
R> library("DataSimilarity")
```

3.1. Cross-Match test for exactly two numeric datasets without target variables

The dataset `dhfr` (Sutherland and Weaver 2004) from the **caret** package (Kuhn and Max 2008) is a binary classification dataset (regarding Dihydrofolate Reductase inhibition) consisting of 325 compounds of which 203 are labeled as 'active' and 122 as 'inactive'. The variables are 228 molecular descriptors. As the active and inactive compounds should differ in their descriptors, we divide the dataset according to the first variable that indicates the activity status.

```
R> data(dhfr, package = "caret")
R> act <- dhfr[dhfr$Y == "active", -1]
R> inact <- dhfr[dhfr$Y == "inactive", -1]
```

For finding an appropriate method, we can use the function `findSimilarityMethod()`. We specify that we have two numeric datasets. As two datasets is already the default, we only need to specify `Numeric = TRUE`:

```
R> findSimilarityMethod(Numeric = TRUE)
```

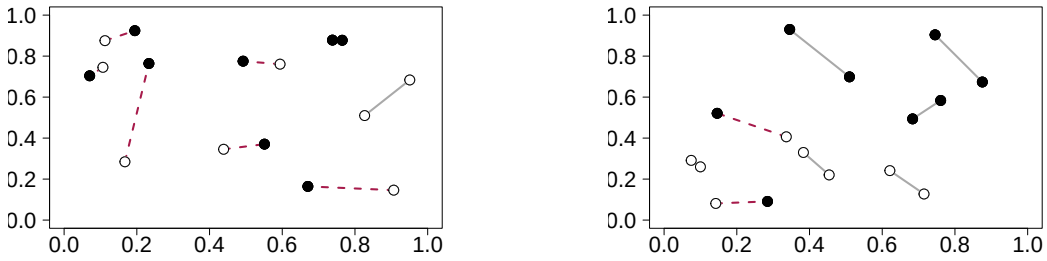
```
 [1] "Bahr"           "BallDivergence" "BF"
 [4] "BG"             "BG2"            "BMG"
 [7] "BQS"            "C2ST"           "CCS"
[10] "CF"             "Cramer"         "DiProPerm"
[13] "DISCOB"         "DISCOF"         "DS"
[16] "Energy"         "engineerMetric" "FR"
[19] "FStest"         "GGRL"           "GPK"
[22] "HMN"            "Jeffreys"       "KMD"
[25] "LHZ"            "MMCM"           "MMD"
[28] "MW"             "NKT"            "OTDD"
[31] "Petrie"         "RISE"           "RItest"
[34] "Rosenbaum"      "SC"             "SH"
[37] "Wasserstein"    "YMRZL"          "ZC"
```

We can also get more information if we set `only.names = FALSE`. We could use this additional information for choosing a method but here we assume that we already have a method in mind. We apply the Rosenbaum cross-match test here to check whether the active and inactive compounds differ. One example method for this case is the Rosenbaum (2005) cross-match test. It is a graph-based method. Most graph-based methods work by constructing a similarity graph on the pooled sample and counting the edges that connect points from different samples. Here, the optimal non-bipartite matching is used, i.e., a graph where pairs of two observations in the pooled sample are connected such that the sum over the edge lengths (= Euclidean distances of connected observations) is minimized. The optimal non-bipartite matching for two example data situations is shown in Figure 1. In case of an odd number of observations, a ghost observation is introduced that has the highest distance to all other observations. The observation that is matched with that ghost observation is discarded from further analysis.

The test statistic of the cross-match test is given by the standardized cross-match count

$$\frac{\text{CMC} - \mathsf{E}_{H_0}(\text{CMC})}{\sqrt{\mathsf{VAR}_{H_0}(\text{CMC})}},$$

where CMC denotes the cross-match count and E_{H_0} and VAR_{H_0} its expectation and variance, respectively, under $H_0 : F_1 = F_2$. The cross-match count is the number of edges connecting points stemming from different datasets. The exact distribution of the test statistic under H_0 is known. For small samples, it can be used for computing an exact p value. For large samples, the asymptotic standard normal distribution of the test statistic can be used. The idea of the



(a) Datasets drawn from the same distribution. (b) Datasets drawn from different distributions.

Figure 1: Optimal non-bipartite matching for example datasets. Dataset 1 is indicated by white points and Dataset 2 by black points. Edges connecting points from different datasets are indicated by red and dashed lines. Edges connecting points from the same sample are indicated by grey and solid lines.

test is that for similar datasets, the number of edges connecting points from different samples is expected to be higher than in datasets that differ. This is illustrated in Figure 1a compared to Figure 1b. In case of data drawn from different datasets, fewer edges connect points from different datasets, indicated by the lower number of red edges in Figure 1b.

Since the combined sample size here is smaller than 340, we can apply the exact test. We can either use the `DataSimilarity()` function and specify the `method` argument accordingly:

```
R> DataSimilarity(act, inact, method = "Rosenbaum", exact = TRUE)

        Exact cross-match test

data:  act and inact
z = -9.4098, p-value < 2.2e-16
alternative hypothesis: The distributions of act and inact are unequal.
sample estimates:
edge.count
        20
```

Alternatively, we can use the `Rosenbaum()` function directly:

```
R> Rosenbaum(act, inact, exact = TRUE)

        Exact cross-match test

data:  act and inact
z = -9.4098, p-value < 2.2e-16
alternative hypothesis: The distributions of act and inact are unequal.
sample estimates:
edge.count
        20
```

The output of the Rosenbaum test is an object of class '`htest`'. The output of the other methods is also in this format. The statistic value can be accessed by saving the result and accessing the `statistic` element of the saved result:

```
R> res.Rosenbaum <- Rosenbaum(act, inact, exact = TRUE)
R> res.Rosenbaum$statistic

        z
-9.409805
```

The p value can be accessed analogously as follows:

```
R> res.Rosenbaum$p.value

[1] 3.56166e-22
```

This holds for almost all other functions in this package. Additionally, the output might include more information specific to the method, which is then described on the respective help page. For the Rosenbaum test, for example, the unstandardized cross-match count is also returned and can be accessed via

```
R> res.Rosenbaum$estimate

edge.count
        20
```

The cross-match count is equal to 20. At most, there could be 122 cross-matches if each observation from the 'inactive' dataset was connected to an observation in the 'active' dataset. Therefore, the cross-match count of 20 can be considered a rather small value. This is also reflected by the z score of -9.41. Consequently, we see that the hypothesis of equal distributions can be rejected with a p value smaller than $2.2 \cdot 10^{-16}$.

We obtain a warning that informs us that a ghost value was introduced when calculating the optimal non-bipartite matching, due to the odd pooled sample size. This means that an artificial point was added to the sample that has the highest distance to all other points in the sample, such that the optimal non-bipartite matching, which needs an even sample size, could be calculated. The ghost value and the point with which it was matched are then discarded from the subsequent calculations.

3.2. MMCM for more than two numeric datasets without target variables

The well-known `iris` dataset (Fisher 1936) included in the **datasets** package that comes with base R (R Core Team 2024) includes measurements of sepal and petals of 50 flowers each of three iris species. We compare the datasets for the three species Iris setosa, versicolor, and virginica, which are known to differ in their sepal and petal measurements.

```
R> data("iris")
R> setosa <- iris[iris$Species == "setosa", -5]
R> versicolor <- iris[iris$Species == "versicolor", -5]
R> virginica <- iris[iris$Species == "virginica", -5]
```

For finding an appropriate method, we can use the function `findSimilarityMethod()` again and specify that we have more than two numeric datasets using the `Numeric` and in addition the `Multiple.samples` options:

```
R> findSimilarityMethod(Numeric = TRUE, Multiple.Samples = TRUE)

 [1] "BallDivergence" "C2ST"           "DISCOB"
 [4] "DISCOF"         "Energy"         "FStest"
 [7] "KMD"            "MMCM"           "MW"
[10] "Petrie"         "RItest"         "SC"
```

For comparing the three datasets, we could, for example, use the Mukherjee, Agarwal, Zhang, and Bhattacharya (2022) Mahalanobis multisample cross-match (MMCM) test, which is a generalization of the cross-match test for multiple samples. The method of Mukherjee *et al.* (2022) is an extension of the Rosenbaum (2005) cross-match test for multiple samples. The cross-match counts $A = (a_{12}, a_{13}, \ldots, a_{ik}, a_{23}, \ldots, a_{2k}, \ldots, a_{k-1,k})^\top$ for all pairs of datasets are calculated using the optimal non-bipartite matching on the pooled sample. The test statistic then is the Mahalanobis distance of the observed cross-counts under the null hypothesis $H_0 : F_1 = F_2 = \cdots = F_k$

$$\text{MMCM} = (A - \mathsf{E}_{H_0}(A))^\top \mathsf{COV}_{H_0}^{-1}(A)(A - \mathsf{E}_{H_0}(A)).$$

The expectation and covariance matrix of the cross-count vector A under H_0 can be calculated analytically and depend only on the sample sizes $n_i, i = 1, \ldots, k$. Small values of the multisample Mahalanobis cross-match (MMCM) statistic indicate similarity. However, as there is no known upperbound, it is hard to interpret the MMCM value. The MMCM statistic follows a $\chi^2_{\binom{k}{2}}$ distribution asymptotically under the null, which can be used for testing.
For applying the test, we can again either use the `DataSimilarity()` function or the `MMCM()` function directly

```
R> DataSimilarity(setosa, versicolor, virginica, method = "MMCM")

        Approximative MMCM test

data:  setosa, versicolor, virginica
chisq = 129.78, df = 3, p-value < 2.2e-16
alternative hypothesis: At least one pair of distributions are unequal.

R> MMCM(setosa, versicolor, virginica)

        Approximative MMCM test

data:  setosa, versicolor, virginica
chisq = 129.78, df = 3, p-value < 2.2e-16
alternative hypothesis: At least one pair of distributions are unequal.
```

The MMCM statistic value on its own is hard to interpret. However, the test rejects the null hypothesis of equal distributions with $p < 2.2 \cdot 10^{-16}$. Therefore, we can conclude that the observed MMCM value presents an extreme value when assuming the null. Thus, the datasets are dissimilar.

3.3. NKT for exactly two numeric datasets with target variables

The `segmentationData` dataset (Hill, LaPan, Li, and Haney 2007) in the **caret** package (Kuhn and Max 2008) includes cell body segmentation data. The dataset contains 119 imaging measurements of 2019 cells to predict the segmentation that is divided into the two classes `PS` for 'poorly segmented' and `WS` for 'well segmented'. Moreover, there is a division into 1009 observations used for training and 1010 observations used as a test set. We compare this training and test set. Ideally, the distributions of the training and test set should be equal in this predictive modelling setting.

```
R> data(segmentationData, package = "caret")
R> test <- segmentationData[segmentationData$Case == "Test", -(1:2)]
R> train <- segmentationData[segmentationData$Case == "Train", -(1:2)]
```

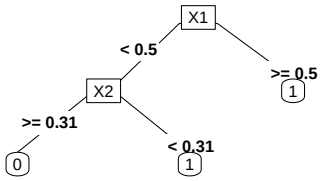
The following methods would be appropriate to use:

```
R> findSimilarityMethod(Numeric = TRUE, Target.Inclusion = TRUE)
```

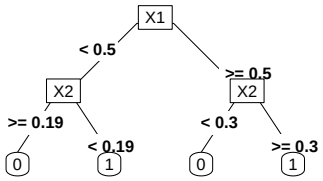
```
[1] "GGRL" "NKT"  "OTDD"
```

Setting `Target.Inclusion = TRUE` selects only the methods that can handle datasets that include a target variable. For demonstration, we choose the method of Ntoutsi, Kalousis, and Theodoridis (2008) and use all three proposed similarity measures NTO1, NTO2, and NTO3. Ntoutsi *et al.* (2008) propose measuring dataset similarity based on probability density estimates derived from decision trees. For this, it is assumed that in addition to both covariate datasets $X^{(1)}$ and $X^{(2)}$, categorical target variables $Y^{(1)}$ and $Y^{(2)}$ are given. On each dataset $X^{(j)}$, a classification tree is constructed with $Y^{(j)}$ as the target variable, $j = 1, 2$. The splits defined by the decision trees induce a partition of the feature space $\mathcal{X}$ such that each leaf node corresponds to one segment in the partition. Figure 2 demonstrates the procedure for two example datasets. First, trees are fit to each dataset (Figure 2a and 2b). Then, the sample space is divided into segments based on the splits performed in each tree (Figure 2c and 2d). These partitions are intersected (Figure 2e) and based on the joint partition, the probability densities $P_D(\mathcal{X})$ and $P_D(Y^{(j)}, \mathcal{X})$ are estimated for $D \in \{X^{(1)}, X^{(2)}, Z\}$.

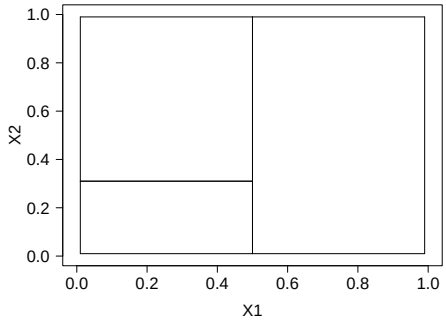
Let n_r denote the number of segments in the joint partition and n_c the number of classes in $X^{(1)}$ and $X^{(2)}$. $\hat{P}_D(\mathcal{X}) \in \mathbb{R}^{n_r}$ uses the proportion of observations in D that fall into each segment of the joint partition. This means that for each of the n_r segments of the partition, the number of observations from dataset D that fall into that segment is counted and divided by the total number of observations in D. For the estimation of the joint density $P_D(Y, \mathcal{X})$, the proportion of observations that fall into each segment of the joint partition and belong to each class is determined, $\hat{P}_D(Y, \mathcal{X}) \in \mathbb{R}^{n_r \times n_c}$. Here, for each of the n_r segments of the partition and each of the n_c classes, the number of observations in D where the corresponding target variable has the respective class value and that fall into the respective segment is counted and divided by the total number of observations in D. The conditional density $P_D(Y|\mathcal{X})$ is estimated by calculating the proportion of observations belonging to each class separately for each segment, $\hat{P}_D(Y|\mathcal{X}) \in \mathbb{R}^{n_r \times n_c}$. Here, for each of the n_r segments of the partition and each of the n_c classes, the number of observations in D where the corresponding target variable has the respective class value and that fall into the respective segment is counted and divided by the total number of observations in D that fall into the respective segment.



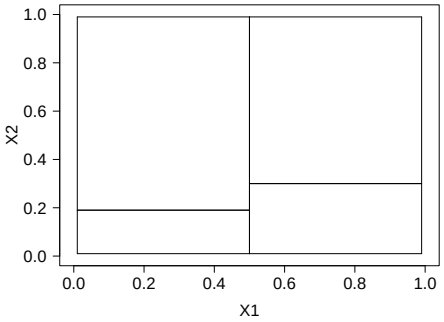
(a) Fitted Tree for Dataset 1.



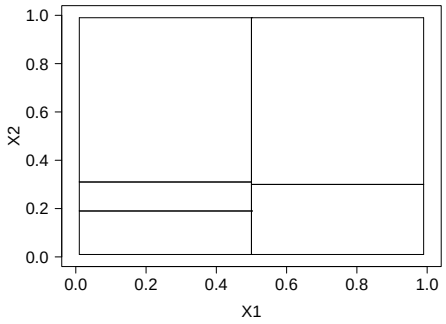
(b) Fitted Tree for Dataset 2.



(c) Partition of sample space derived from fitted tree for Dataset 1.



(d) Partition of sample space derived from fitted tree for Dataset 2.



(e) Intersected partition (greatest common refinement, GCR) from fitted trees for Datasets 1 and 2. Each dataset includes two covariates and a binary target variable.

Figure 2: Partitioning of sample space by fitting trees to two example datasets.

Then, Ntoutsi *et al.* (2008) consider the similarity index

$$s(p, q) = \sum_i \sqrt{p_i \cdot q_i}$$

for vectors p and q, where $(n_r \times n_c)$-matrices are interpreted as $(n_r \cdot n_c)$-dimensional vectors. For the conditional distribution, the similarity vector $S(Y|\mathcal{X}) \in \mathbb{R}^{n_r}$ is computed with $S(Y|\mathcal{X})_i = s(\hat{P}_{X^{(1)}}(Y|\mathcal{X})_{i\bullet}, \hat{P}_{X^{(2)}}(Y|\mathcal{X})_{i\bullet})$ and index $i\bullet$ denoting the i-th row. Based on this, three similarity measures for datasets are proposed:

1. NTO1 $= s(\hat{P}_{X^{(1)}}(\mathcal{X}), \hat{P}_{X^{(2)}}(\mathcal{X}))$
2. NTO2 $= s(\hat{P}_{X^{(1)}}(Y, \mathcal{X}), \hat{P}_{X^{(2)}}(Y, \mathcal{X}))$
3. NTO3 $= S(Y|\mathcal{X})^{\top} \hat{P}_Z(\mathcal{X})$.

All three measures have values in the interval $[0, 1]$, where high values correspond to high similarity.

For applying the method to our data, the `target1` and `target2` arguments have to be specified as the column names of the target variable in the first and second supplied datasets, respectively. Here, the target variable is named `"Class"` in both cases. Again, we can use either the `DataSimilarity()` function or `NKT()`.

```
R> DataSimilarity(train, test, method = "NKT", target1 = "Class",
+                 target2 = "Class", tune = FALSE)

        Data similarity according to Ntoutsi et al. (2008), version 1

data:  train and test
s = 0.96931
alternative hypothesis: The distributions of train and test are unequal.

R> NKT(train, test, target1 = "Class", target2 = "Class", tune = FALSE)

        Data similarity according to Ntoutsi et al. (2008), version 1

data:  train and test
s = 0.96931
alternative hypothesis: The distributions of train and test are unequal.

R> DataSimilarity(train, test, method = "NKT", target1 = "Class",
+                 target2 = "Class", tune = FALSE, version = 2)

        Data similarity according to Ntoutsi et al. (2008), version 2

data:  train and test
s = 0.92444
alternative hypothesis: The distributions of train and test are unequal.

R> NKT(train, test, target1 = "Class", target2 = "Class", tune = FALSE,
+      version = 2)

        Data similarity according to Ntoutsi et al. (2008), version 2

data:  train and test
s = 0.92444
alternative hypothesis: The distributions of train and test are unequal.
```

```
R> DataSimilarity(train, test, method = "NKT", target1 = "Class",
+                 target2 = "Class", tune = FALSE, version = 3)

        Data similarity according to Ntoutsi et al. (2008), version 3

data:  train and test
s = 0.96648
alternative hypothesis: The distributions of train and test are unequal.

R> NKT(train, test, target1 = "Class", target2 = "Class", tune = FALSE,
+      version = 3)

        Data similarity according to Ntoutsi et al. (2008), version 3

data:  train and test
s = 0.96648
alternative hypothesis: The distributions of train and test are unequal.
```

We observe high similarity between the training and test datasets with all three methods, reflected by the similarity values `s` that are all close to the maximal value 1. For the method of Ntoutsi *et al.* (2008), no test is proposed and therefore, no p value is calculated.

3.4. HMN for exactly two categorical datasets without target variables

The `banque` dataset from the **ade4** package (Dray and Dufour 2007) consists of bank survey data of 810 customers. All variables are categorical and contain socio-economic information of the customers. We divide the data into bank card owners and non-bank card owners and compare these two groups. In total, 243 out of the 810 customers own a bank card. Bank card owners and non-bank card owners might differ in their socio-economic characteristics.

```
R> data(banque , package = "ade4")
R> card <- banque[banque$cableue == "oui", -7]
R> no.card <- banque[banque$cableue == "non", -7]
```

We again apply the `findSimilarityMethod()` function to find appropriate methods for comparing two categorical datasets. Again, two samples are the default. Therefore, we only have to specify `Categorical = TRUE`.

```
R> findSimilarityMethod(Categorical = TRUE)

 [1] "C2ST"       "CCS_cat"    "CF_cat"     "CMDistance" "FR_cat"
 [6] "GGRL_cat"   "HMN"        "MMCM"       "MMD"        "OTDD"
[11] "Petrie"     "YMRZL"      "ZC_cat"
```

For demonstration, we use the random forest test of Hediger, Michel, and Näf (2022) to compare these two groups. Hediger *et al.* (2022) provide a two-sample test based on random forests that is applicable for both numeric and categorical data. For this test, a pooled dataset

is created where each observation is labeled according to its original dataset membership, and a random forest is trained to distinguish between the dataset labels. The idea is that if the datasets are generated from the same distribution, the classification error of the random forest should be close to the chance level, otherwise, the classifier should be able to distinguish between the two distributions and hence the classification error should be lower than the chance level. One advantage of using random forests as the classifier is that it requires almost no tuning. An asymptotic test is proposed. For this, the pooled dataset has to be split into a training set on which the random forest is trained and a test set on which its classification error is evaluated. In the implementation, both datasets are split in half to create a training and a test dataset. Alternatively, an out-of-bag (OOB) based permutation test can be performed that does not require data splitting. OOB statistics can be used to increase the sample efficiency compared to the test based on a holdout sample. Both the OOB-based test and the asymptotic version of the test are implemented. The test statistic is either the mean of the per-class OOB or test classification errors or the overall OOB or test classification error, respectively. In the asymptotic case, a binomial test is performed in case of the overall classification error, or a Z test is performed in case of the mean per-class classification error. Otherwise, a permutation test is performed. The variable importance measures of the random forest can provide additional insights into sources of distributional differences.

For easier interpretation, we here look at the overall out-of-bag (OOB) prediction error instead of the per-class OOB prediction error and perform a permutation test with 1000 permutations. For reproducibility, we set a seed before applying the method. Alternatively, we could supply the seed via the `seed` argument for setting the seed within the function.

```
R> set.seed(1234)
R> DataSimilarity(card, no.card, method = "HMN", n.perm = 1000,
+                 statistic = "OverallOOB")

        Permutation OverallOOB random forest based two-sample test

data:  card and no.card
p.hat = 0.1605, p-value = 0.000999
alternative hypothesis: The distributions of card and no.card are unequal.

R> set.seed(1234)
R> HMN(card, no.card, n.perm = 1000, statistic = "OverallOOB")

        Permutation OverallOOB random forest based two-sample test

data:  card and no.card
p.hat = 0.1605, p-value = 0.000999
alternative hypothesis: The distributions of card and no.card are unequal.
```

The overall OOB prediction error is 0.161, which is considerably smaller than the naive prediction error of 243/810 = 0.3. Therefore, the random forest can distinguish between the datasets, so we can conclude that the datasets differ. This is also reflected by the p value of 9.990e-04.

3.5. C2ST for more than two categorical datasets without target variables

We consider the `banque` dataset from the **ade4** package (Dray and Dufour 2007) again. This time, we split it by the nine socio-professional categories given by 'csp', which are again expected to differ with regard to the other socio-economic characteristics.

```
R> data(banque, package = "ade4")
R> agric <- banque[banque$csp == "agric", -1]
R> artis <- banque[banque$csp == "artis", -1]
R> cadsu <- banque[banque$csp == "cadsu", -1]
R> inter <- banque[banque$csp == "inter", -1]
R> emplo <- banque[banque$csp == "emplo", -1]
R> ouvri <- banque[banque$csp == "ouvri", -1]
R> retra <- banque[banque$csp == "retra", -1]
R> inact <- banque[banque$csp == "inact", -1]
R> etudi <- banque[banque$csp == "etudi", -1]
```

To compare these datasets, we now need a method that can handle multiple datasets at once:

```
R> findSimilarityMethod(Categorical = TRUE, Multiple.Samples = TRUE)
```

```
[1] "C2ST"   "MMCM"   "Petrie"
```

We apply the classifier two-sample test (C2ST). The general idea of this method by Lopez-Paz and Oquab (2017) is to use a classifier to determine which of two or more datasets a sample belongs to. The *classifier two-sample test (C2ST)* uses the classification accuracy of this classifier as its test statistic.
The C2ST consists of five steps:

1. Construct the dataset consisting of the samples from all datasets, labeled with their membership to each of the datasets.
2. Assign the observations of the dataset constructed in 1. randomly to a training and test set.
3. Train a classifier that predicts for an observation to which dataset $X^{(j)}, j = 1, \ldots, k$ it belongs.
4. Calculate the C2ST statistic, which is the accuracy on the test set. The accuracy should be close to the chance level for $F_1 = \cdots = F_k$, and it should be greater than the chance level for $\exists i \neq j \in \{1, \ldots, k\} : F_i \neq F_j$ since in the latter case the classifier should identify distributional differences between the samples.
5. Calculate a p value using a binomial test for comparing the accuracy to the chance level.

Maximizing the power of a C2ST is a trade-off between using a large training set to optimize the classifier and a large test set to better evaluate the performance of the classifier.
The test statistic is interpretable as the percentage of samples that are correctly classified on the unseen test data. The above-mentioned test of Hediger *et al.* (2022) can be seen

as a special case of the general framework proposed by Lopez-Paz and Oquab (2017). One difference in the implementation of the tests is that for the C2ST, categorical data is dummy-encoded, while for the test of Hediger *et al.* (2022) the categorical variables are passed to `ranger::ranger()` directly. Moreover, the use of OOB predictions and feature importance is specific to the random forest-based test and cannot be used for all of the available classifiers for the C2ST. Further, the C2ST uses the accuracy as its test statistic while the test of Hediger *et al.* (2022) uses the classification error, i.e., 1 − accuracy.

First, we use the default K-NN classifier. Categorical variables are dummy-coded. Again, we can use either `DataSimilarity()` or `C2ST()`:

```
R> DataSimilarity(agric, artis, cadsu, inter, emplo, ouvri, retra, inact,
+                 etudi, method = "C2ST")

        Approximative Classifier Two-Sample Test using knn

data:  agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi
p.hat = 0.33333, size = 567.00000, prob = 0.22593, p-value =
1.078e-07
alternative hypothesis: At least one pair of distributions are unequal.

R> C2ST(agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi)

        Approximative Classifier Two-Sample Test using knn

data:  agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi
p.hat = 0.33333, size = 567.00000, prob = 0.22593, p-value =
1.078e-07
alternative hypothesis: At least one pair of distributions are unequal.
```

The accuracy of the K-NN classifier is 0.333. It is larger than the naive accuracy for always predicting the largest class, which is given by `prob = 0.226` in the output. The classifier seems to be able to distinguish between the datasets, and we can therefore regard them as dissimilar. Moreover, the null hypothesis of equal distributions can be rejected with a p value of 1.078e-07.

For demonstration, we additionally perform the C2ST with a neural net classifier.

```
R> DataSimilarity(agric, artis, cadsu, inter, emplo, ouvri, retra, inact,
+                 etudi, method = "C2ST", classifier = "nnet",
+                 train.args = list(trace = FALSE))

        Approximative Classifier Two-Sample Test using nnet

data:  agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi
p.hat = 0.16667, size = 567.00000, prob = 0.22593, p-value =
0.05269
alternative hypothesis: At least one pair of distributions are unequal.
```

```
R> C2ST(agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi,
+       classifier = "nnet", train.args = list(trace = FALSE))

        Approximative Classifier Two-Sample Test using nnet

data:  agric, artis, cadsu, inter, emplo, ouvri, retra, inact, etudi
p.hat = 0.27778, size = 567.00000, prob = 0.22593, p-value =
2.425e-05
alternative hypothesis: At least one pair of distributions are unequal.
```

The results are very similar to using K-NN.

3.6. OTDD for exactly two categorical datasets with target variables

We consider the `banque` dataset from the **ade4** package (Dray and Dufour 2007) again. In this case, we interpret the savings bank amount (`eparliv`) variable as the target variable, which is again supplied via the `target1` and `target2` arguments. It is divided into the three categories '> 20000', '> 0 and < 20000', and 'nulle'. We divide the data into the socio-professional categories as before, and now need a method for two categorical datasets that include a target variable.

```
R> findSimilarityMethod(Categorical = TRUE, Target.Inclusion = TRUE)

[1] "GGRL_cat" "OTDD"
```

We use the optimal transport dataset distance (OTDD) to compare the resulting datasets for craftsmen, shopkeepers, company directors ('artis'), to that of higher intellectual professions ('cadsu'), and to that of manual workers ('ouvri'). Alvarez-Melis and Fusi (2020a) define this distance based on optimal transport between datasets that include a target (class) variable Y. The *optimal transport dataset distance* (OTDD) is defined as

$$d_{\text{OT}}(X^{(1)}, X^{(2)}) = \min_{\pi \in \Pi(F_1, F_2)} \int_{\mathcal{Z} \times \mathcal{Z}} d_{\mathcal{Z}}(z, z')^q \, \mathrm{d}\pi(z, z')$$

where $X^{(1)}, X^{(2)}$ denote the two datasets,

$$\Pi(F_1, F_2) := \{\pi_{1,2} \in \mathcal{P}(\mathcal{Z} \times \mathcal{Z}) | \pi_1 = F_2, \pi_2 = F_2\}$$

is the set of joint distributions over the product space $\mathcal{Z} \times \mathcal{Z}$ over the sample space of the pooled sample with marginal distributions F_2 and F_2, and

$$d_{\mathcal{Z}}(z, z') := (d_{\mathcal{X}}(x, x')^q + W_{q'}^{q'}(\alpha_y, \alpha_{y'}))^{1/q}.$$

defines a distance of two points $z^\top = (x^\top, y)$, and $z'^\top = (x'^\top, y')$ in the pooled sample. $d_{\mathcal{X}}$ defines a distance on the covariate space, e.g., the Euclidean distance, and $W_{q'}(\alpha_y, \alpha_{y'})$ is the q'-Wasserstein distance of the distribution of the subset of covariate data with corresponding response value y and the distribution of the subset of covariate data with corresponding response value y'. The powers q and q' have to be chosen in advance to calculate the OTDD.

The optimal transport problem can intuitively be motivated by imagining each probability density as a pile of dirt. Then, the cost function corresponds to the cost for transporting the dirt from one point to another, which is proportional to the distance between the two points. The optimal transport then corresponds to the lowest cost required for moving one pile of dirt fully to the shape and location of the other. Therefore, distributions can be regarded as more similar if the optimal transport between them is lower. For an intuitive explanation and visualization of the OTDD, also refer to Alvarez-Melis and Fusi (2020b). As all variables are categorical, we use the Hamming distance instead of the default Euclidean distance. We can either use `DataSimilarity()` or `OTDD()`.

```
R> DataSimilarity(artis, cadsu, method = "OTDD", target1 = "eparliv",
+                 target2 = "eparliv", feature.cost = hammingDist)

        Optimal Transport Dataset Distance

data:  artis and cadsu
OTDD = 44.166
alternative hypothesis: Distributions of artis and cadsu are unequal

R> OTDD(artis, cadsu, target1 = "eparliv", target2 = "eparliv",
+       feature.cost = hammingDist)

        Optimal Transport Dataset Distance

data:  artis and cadsu
OTDD = 44.166
alternative hypothesis: Distributions of artis and cadsu are unequal
```

We obtain a dataset distance of 44.166 between craftsmen/shopkeepers/company directors and executives/higher intellectual professions. For the OTDD, low values correspond to high similarity, and the minimum value is 0. The observed value is clearly larger than zero, so the datasets are not exactly similar. How dissimilar they are is however hard to interpret from the observed OTDD value on its own. For the OTDD, no test is proposed and therefore, no p value is calculated.

```
R> DataSimilarity(artis, ouvri, method = "OTDD", target1 = "eparliv",
+                 target2 = "eparliv", feature.cost = hammingDist)

        Optimal Transport Dataset Distance

data:  artis and ouvri
OTDD = 49.427
alternative hypothesis: Distributions of artis and ouvri are unequal

R> OTDD(artis, ouvri, target1 = "eparliv", target2 = "eparliv",
+       feature.cost = hammingDist)
```

```
        Optimal Transport Dataset Distance

data:  artis and ouvri
OTDD = 49.427
alternative hypothesis: Distributions of artis and ouvri are unequal
```

We obtain a dataset distance of 49.427 between craftsmen/shopkeepers/company directors and manual workers. Again, this value on its own is hard to interpret. However, we can compare the values and conclude that the data of craftsmen/shopkeepers/company directors is more similar to that of executives/higher intellectual professions than to that of manual workers.

4. Examples for case b): We are looking for a method to apply

In the following, the individual steps of the COMPARE procedure for finding a suitable dataset similarity method are first described and than applied for the application of comparing two strategies of creating a synthetic dataset.

4.1. COMPARE steps

Characteristics of datasets The following characteristics of the datasets for which similarity is assessed should be considered:

- How many datasets are there to compare simultaneously?
- What are the scale levels of the variables in the datasets?
- Do the datasets include target variables that we wish to treat differently from the remaining covariates?
- Do the sample sizes of the datasets differ?
- Does any of the datasets include fewer observations than variables?

If we have answers to all of these questions, we can proceed with the next step. The help pages of the methods also give a first overview for which datasets each method is applicable.

Objective of comparison We should define the objective of comparing the datasets. The main decision here is whether we want to perform a test or are only interested in measuring (dis)similarity. In the case where only or mainly the (dis)similarity is of interest, the interpretability of the statistic values might be desirable, especially boundedness.

Method requirements Next, we should should consider whether any theoretical properties (invariances to shift/scale/rotation, metric properties, consistency of test) are relevant in the given application. For example, invariances might be desirable if data will be transformed, metric properties are useful in applications relying on distances between datasets, like clustering the datasets, and consistency of the test might be desirable if the test result is of interest.

Production of method list After assessing the requirements, we can call the function `findSimilarityMethod()`. The arguments of that function correspond to the theoretical criteria of Stolte *et al.* (2024). For a full list, refer to the help pages in this package or the cited article. The most relevant criteria for the method choice are already covered by the first three steps in this procedure, so we can now call `findSimilarityMethod()` using the answers from above, which returns a list of methods that fulfill our requirements.

Assessment of computational complexity Another aspect to consider is the computational complexity of the methods. Different methods require different computational resources, both with regard to runtime and to memory consumption. These requirements also depend on the dataset characteristics, especially on the dimensions and number of datasets, but also, for example, in the case of categorical data, for some methods, on the number of categories. Limits for the computational resources, especially for runtime, will also depend on the application. In applications where two or more concrete datasets are compared once, higher runtimes will likely be considered more tolerable than for running a simulation study where each method is applied at least hundreds of times. Theoretical complexity is given in the corresponding column of `method.table` and will also be returned by `findSimilarityMethod()`, if `only.names` is set to `FALSE`. In Stolte *et al.* (2026a) and Stolte *et al.* (2026b), the methods were additionally benchmarked regarding empirical runtime and memory requirements. The results of those benchmarks can be summarized as follows. For categorical data, the CM distance shows the lowest resource consumption for low numbers of variables and classes, but can no longer be computed if the number of classes or variables becomes too high, since an enumeration of the whole sample space is required for calculation. The edge count test (FR, CF, CCS, ZC) and the other graph-based methods, MMCM and Petrie, show the next lowest resource consumption. The classifier-based methods C2ST and HMN need considerably more runtime and memory. The two methods, GGRL and OTDD, that can also handle a target variable appropriately, show by far the highest resource consumption. For numeric data, the methods with the lowest resource consumption are the Engineer metric, Wasserstein distance, interpoint distance-based methods like Energy, DISCO, Bahr, and BF, as well as graph-based (SC, KMD) and kernel-based methods (MMD, GPK) consume the fewest resources. The highest resource consumption is observed for BMG, which requires computing the shortest Hamilton path. Again, the methods that can handle a target variable (NKT, GGRL, OTDD) or use a classifier (C2ST, YMRZL), but also Jeffreys divergence, DS, SH, and BG2. Note that BG becomes unfeasible for high numbers of variables. For more details and the complete ranking, see Sections 3.2.1 and 3.3.1 of Stolte *et al.* (2026b). It should be noted that the benchmark was performed with zero permutations for all methods. The runtimes of the permutation and Bootstrap tests will increase considerably with increasing the number of permutations. If the sample size is large and the runtime is limited, methods that offer asymptotic tests might be preferred. The information on which tests are available can be found in the method descriptions in the Details vignette as well as on the corresponding help pages. Runtime and memory requirements can, of course, additionally be evaluated in practice by simply applying the methods to the given data or by performing a small benchmark tailored to the data characteristics at hand.

Ranking of methods Now that we reduced the set of methods to those that satisfy all practical requirements, we can choose from the remaining methods by considering their per-

formance in simulation studies like those provided in Stolte *et al.* (2026a) and Stolte *et al.* (2026b). There, overall method rankings are provided, but also more detailed results and even decision rules taking into account specific dataset characteristics. The overall results for categorical datasets can be summarized as follows. The edge count tests (FR, CCS, CF, ZC) often showed high performance for all considered alternatives, especially the FR using the 5-MST as the similarity graph and the union of graphs. The CM distance showed even better performance in some scenarios, but was infeasible for higher numbers of categories or variables. The HMN performed okay for equal sample sizes but worse for unequal sample sizes. The C2ST performance depended on the scenario. Out of the methods that treat the target variable separately, the OTDD outperformed the GGRL, but both performed poorly compared to all other methods.

For two numeric datasets, the BF method performed best overall. The combination of BF (using FracB as the kernel function), ZC (using the 5-MST as the similarity graph and $\kappa = 1$), DS, and GPK was able to cover 90% of the considered scenarios with good performance in the two-sample case. For the multi-sample case, the DISCO F-test (with $\alpha = 0.5$) performed best overall. Combined with SC (using the 5-MST and the statistic S), it was again able to cover 90% of the considered scenarios with good performance for the multi-sample test.

For more details, we refer to the original studies in Stolte *et al.* (2026a) and Stolte *et al.* (2026b), as a comprehensive discussion of the results for all methods is out of scope for this vignette.

Estimation and testing Lastly, we can apply the chosen method(s) to our datasets to estimate their (dis)similarity and potentially test whether the underlying distributions of the datasets coincide using the `DataSimilarity()` function and specifying the chosen method via the `method` argument. The interpretation of the results depend on the method. More details on this can be found on the corresponding help pages and in the literature that is referenced there.

4.2. Illustration

In the following, we illustrate the steps for choosing and applying a method as described in the previous subsection to a real-world example dataset in the setting of creating a realistic synthetic dataset. First, we start by attaching the **DataSimilarity** package.

```
R> library("DataSimilarity")
```

The `banque` dataset from the **ade4** package (Dray and Dufour 2007) consists of bank survey data of 810 customers. All variables are categorical and contain socio-economic information of the customers. Details on the variables can be found on the help page of the dataset `?banque`.

```
R> data("banque", package = "ade4")
R> head(banque)
```

```
    csp duree oppo  age sexe interdit cableue assurvi soldevu eparlog
1 ouvri   d48  non ai75  hom      non     non     non      n1     nul
2 cadsu   d24  non ai35  hom      non     non     non      n1     nul
3 cadsu   d48  non ai75  hom      non     non     non      p1     nul
```

```
4 inact   d24  non ai45  fem      oui     non     non      p2     nul
5 retra  d812  non ai75  hom      non     non     non      p4     nul
6 inact  dp12  non ai45  fem      non     non     non      p1     nul
  eparliv credhab credcon versesp retresp remiche preltre prelfin
1     nul     non     nul     non     fai     nul     nul     nul
2     nul     non     nul     non     fai     nul     nul     nul
3     nul     non     nul     non     fai     nul     nul     nul
4     nul     non     nul     oui     fai     fai     nul     nul
5     nul     non     nul     non     fai     nul     nul     nul
6     nul     non     nul     non     fai     nul     nul     nul
  viredeb virecre porttit
1     nul     nul     nul
2     nul     nul     nul
3     nul     nul     nul
4     nul     nul     moy
5     fai     nul     for
6     nul     nul     nul
```

R> summary(banque)

```
      csp       duree      oppo       age       sexe     interdit
 ouvri  :183   dm2 : 91   non:752   ai25: 90   hom:558   non:752
 emplo  :151   d24 :132   oui: 58   ai35:156   fem:252   oui: 58
 cadsu  :103   d48 :207             ai45:212
 inter  :102   d812:144             ai55:174
 inact  : 85   dp12:236             ai75:178
 etudi  : 57
 (Other):129
 cableue   assurvi   soldevu  eparlog   eparliv   credhab   credcon
 non:567   non:674   p4: 95   for: 64   for: 44   non:718   nul:685
 oui:243   oui:136   p3: 69   fai: 44   fai:144   oui: 92   fai: 68
                     p2:145   nul:702   nul:622             for: 57
                     p1:271
                     n1:162
                     n2: 68

 versesp   retresp   remiche   preltre   prelfin   viredeb
 oui: 87   fai:701   for: 71   nul:728   nul:707   nul:485
 non:723   moy: 76   moy: 70   fai: 47   fai: 47   fai:244
           for: 33   fai:220   moy: 35   moy: 56   moy: 41
                     nul:449                       for: 40



 virecre   porttit
 for:118   nul:630
 moy:114   fai: 69
```

```
 fai:174   moy: 61
 nul:404   for: 50
```

To ensure that the ordinal variables are handled correctly later on, we first transform them to '`ordered`' variables. The nominal variables are kept as '`factor`'s.

```
R> ord.vars <- c("duree", "age", "soldevu", "eparlog", "eparliv",
+                "credcon", "retresp", "remiche", "preltre", "prelfin",
+                "viredeb", "virecre", "porttit")
R> banque[, ord.vars] <- lapply(banque[, ord.vars], ordered)
R> str(banque)

'data.frame':        810 obs. of  21 variables:
 $ csp     : Factor w/ 9 levels "agric","artis",..: 6 3 3 8 7 8 8 8 6 1 ...
 $ duree   : Ord.factor w/ 5 levels "dm2"<"d24"<"d48"<..: 3 2 3 2 4 5 3 5 2 3 ...
 $ oppo    : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ age     : Ord.factor w/ 5 levels "ai25"<"ai35"<..: 5 2 5 3 5 3 5 5 2 4 ...
 $ sexe    : Factor w/ 2 levels "hom","fem": 1 1 1 2 1 2 2 2 1 1 ...
 $ interdit: Factor w/ 2 levels "non","oui": 1 1 1 2 1 1 1 1 1 1 ...
 $ cableue : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ assurvi : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ soldevu : Ord.factor w/ 6 levels "p4"<"p3"<"p2"<..: 5 5 4 3 1 4 4 1 4 4 ...
 $ eparlog : Ord.factor w/ 3 levels "for"<"fai"<"nul": 3 3 3 3 3 3 3 3 3 3 ...
 $ eparliv : Ord.factor w/ 3 levels "for"<"fai"<"nul": 3 3 3 3 3 3 3 3 3 3 ...
 $ credhab : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ credcon : Ord.factor w/ 3 levels "nul"<"fai"<"for": 1 1 1 1 1 1 1 1 1 1 ...
 $ versesp : Factor w/ 2 levels "oui","non": 2 2 2 1 2 2 2 2 2 2 ...
 $ retresp : Ord.factor w/ 3 levels "fai"<"moy"<"for": 1 1 1 1 1 1 1 1 1 1 ...
 $ remiche : Ord.factor w/ 4 levels "for"<"moy"<"fai"<..: 4 4 4 3 4 4 4 1 4 3 ...
 $ preltre : Ord.factor w/ 3 levels "nul"<"fai"<"moy": 1 1 1 1 1 1 1 1 1 1 ...
 $ prelfin : Ord.factor w/ 3 levels "nul"<"fai"<"moy": 1 1 1 1 1 1 1 1 1 1 ...
 $ viredeb : Ord.factor w/ 4 levels "nul"<"fai"<"moy"<..: 1 1 1 1 2 1 1 1 1 1 ...
 $ virecre : Ord.factor w/ 4 levels "for"<"moy"<"fai"<..: 4 4 4 4 4 4 4 3 4 4 ...
 $ porttit : Ord.factor w/ 4 levels "nul"<"fai"<"moy"<..: 1 1 1 3 4 1 1 4 1 1 ...
```

Assume that we want to create a synthetic dataset that is similar to this real dataset. We might consider different strategies and evaluate these with regard to the similarity of the synthetic dataset to the original one. First, we create the synthetic datasets. For this, we use two different approaches. The first one is to naively sample data for each variable using the observed frequencies as the class distributions.

```
R> naiveSynth <- function(v) {
+    tab <- prop.table(table(v))
+    n <- length(v)
+    new.v <- sample(x = names(tab), size = n, prob = tab, replace = TRUE)
+    if(is.ordered(v)) {
+      new.v <- ordered(new.v, levels = levels(v))
```

```
+    } else {
+      new.v <- factor(new.v, levels = levels(v))
+    }
+    return(new.v)
+  }
R> set.seed(2026)
R> synth.n <- as.data.frame(lapply(banque, naiveSynth))
R> head(synth.n)

    csp duree oppo  age sexe interdit cableue assurvi soldevu eparlog
1 inact  dp12  non ai75  fem      non     non     oui      p1     nul
2 inter   d24  non ai75  hom      non     non     non      p1     nul
3 ouvri   dm2  non ai75  hom      non     non     non      n1     for
4 emplo  d812  non ai55  hom      non     non     non      n2     nul
5 inter   dm2  oui ai45  hom      non     non     non      p1     nul
6 ouvri   d24  non ai55  hom      non     oui     non      p1     nul
  eparliv credhab credcon versesp retresp remiche preltre prelfin
1     nul     non     nul     non     fai     for     nul     moy
2     nul     non     nul     non     fai     nul     nul     nul
3     nul     non     nul     non     fai     nul     nul     nul
4     fai     non     nul     oui     fai     nul     nul     nul
5     nul     non     nul     non     fai     nul     fai     nul
6     nul     non     nul     non     fai     fai     nul     nul
  viredeb virecre porttit
1     nul     moy     nul
2     fai     fai     nul
3     fai     for     nul
4     fai     fai     nul
5     nul     for     nul
6     fai     for     nul
```

As a second approach, we use the **synthpop** package (Nowok, Raab, and Dibben 2016) to synthesize our data. This uses more sophisticated methods for synthesizing the data that for example also consider correlations between the variables.

```
R> library("synthpop")
R> synth.p <- syn(banque, seed = 2026, print.flag = FALSE)
R> head(synth.p$syn)

    csp duree oppo  age sexe interdit cableue assurvi soldevu eparlog
1 emplo   d48  non ai35  hom      non     oui     non      p1     nul
2 emplo   d24  non ai35  fem      non     non     non      p1     nul
3 inact   d48  non ai75  fem      non     oui     non      p3     nul
4 ouvri  dp12  oui ai45  hom      non     non     non      p1     fai
5 artis  d812  non ai55  hom      non     non     oui      p1     nul
6 inter   dm2  non ai75  hom      non     non     non      p1     nul
  eparliv credhab credcon versesp retresp remiche preltre prelfin
```

```
1     nul     non     nul     non     fai     fai     nul     nul
2     fai     non     nul     non     fai     nul     nul     nul
3     nul     non     nul     non     fai     fai     nul     nul
4     nul     non     nul     non     fai     nul     nul     nul
5     nul     non     nul     oui     for     fai     nul     nul
6     nul     non     nul     non     fai     nul     moy     nul
  viredeb virecre porttit
1     fai     nul     nul
2     nul     fai     nul
3     nul     nul     nul
4     for     moy     nul
5     fai     fai     nul
6     fai     nul     nul
```

Now, we have two synthetic datasets and want to evaluate how closely they capture the real data. For this, we would like to use a dataset similarity method. To find a method for our application, we follow the steps described in the previous section.

Characteristics of datasets For the pairwise comparison of our two synthetic datasets to the original one, our two datasets each consist of 21 categorical variables. The datasets do not contain any target variables. The sample sizes are equal as we always created our synthetic data to match the original sample size of 810. The sample size is also larger than the of variables.

Objective of comparison We are interested in comparing the similarity of the synthetic datasets with the original dataset. Interpretability of the resulting similarity values would therefore be very helpful in this application. Upper or lower bounds of the values would provide additional interpretability of the values themselves but are not essential for comparing the synthesizing strategies.

Method requirements The invariances to transformations do not apply to categorical data and are therefore not relevant in this context. The metric properties or consistency of corresponding tests are also not essential in this case.

Production of method list Now that we know all of our requirements, we can call the `findSimilarityMethod()` function to find appropriate methods. We specify all the criteria that we identified above, which are the applicability to categorical variables and interpretability of the resulting values.

```
R> findSimilarityMethod(Categorical = TRUE, Interpretable.units = TRUE)
```

```
[1] "C2ST"   "FR_cat" "HMN"    "Petrie" "YMRZL"
```

This gives us a list of five methods that would fulfill our requirements.

Assessment of computational complexity If we only want to make the two comparisons of the generated datasets from above, runtime and memory are no limiting factors. However, if we wanted to compare the two strategies for data synthesis more generally, we would want to generate more datasets with each strategy and compare each against the original `banque` dataset. Then, runtime might become a relevant factor. According to the benchmarks performed in Stolte *et al.* (2026a), the graph-based methods `FR_cat()` or `Petrie()` might be preferable to the other classifier-based methods `C2ST()`, `HMN()`, and `YMRZL()` in that case due to lower runtimes. Depending on the runtime limits and the desired number of dataset comparisons, these might even become the only feasible options. For this illustration we will, however, not exclude any of the methods due to their runtime.

Ranking of methods In the performance comparisons of the simulation study in Stolte *et al.* (2026a), `FR_cat()` outperformed the remaining methods in many of the considered scenarios. Therefore, we choose to use it here.

The Friedman-Rafsky (FR) test (Friedman and Rafsky 1979) (also called original edge-count test) is a graph-based test. A similarity graph (in the original proposed form a minimum spanning tree, MST) is calculated on the pooled sample created by pooling the two datasets using an appropriate distance measure. Then, the edges connecting points from different datasets are counted. This between-sample edge-count R_{12} can also be standardized

$$T_{\text{FR}} = \frac{R_{12} - \mathsf{E}_{H_0}(R_{12})}{\sqrt{\mathsf{VAR}_{H_0}(R_{12})}}$$

for testing. The expectation and variance under the null can be calculated analytically. For small samples, a permutation test is proposed. For large samples, the asymptotic standard normal distribution of the test statistic can be used. The idea of the test is that for similar datasets, the number of edges connecting points from different samples is expected to be higher than in datasets that differ. In categorical data, ties in the distance matrix are expected. In that case, the MST is no longer uniquely defined. This can be solved by calculating all optimal MST solutions and either calculating the test statistic as defined above on the union of these graphs, or on each graph individually and averaging the test statistic values over all optimal MST solutions (Zhang and Chen 2022).

Estimation and testing We use the FR test with the recommended parameter settings from the simulations, which is using the union and using the 5-MST instead of the regular MST. The K-MST is defined as follows. First, the regular MST is calculated. Then, the edges used by that MST are removed from the edge set of the complete graph and a new MST is calculated using only the remaining edges. For the third MST, the edges used by the first and second MST are removed and it is calculated again only using the remaining edges. This is repeated until the Kth MST was calculated. Then, the K-MST is the union of all the resulting MSTs. This results in denser graphs which have shown better performance in practice (Zhu and Chen 2024).

As a distance function, we use the Gower distance `daisy()` from the **cluster** package to combine appropriate distances for the nominal and ordinal variables present in the dataset. Like all other methods, the test can be applied using the unified interface of the `DataSimilarity()` function. The chosen graph and aggregation type are already the default settings in the implementations. The aggregation type could be changed using the `agg.type` argument and the

graph type can be specified using the `graph.type` and `K` arguments. We only have to specify the distance function using the `dist.fun` argument. We first compare the original dataset and the naively generated one.

```
R> DataSimilarity(banque, synth.n, method = "FR_cat",
+                 dist.fun = function(x) cluster::daisy(x, metric = "gower"))

        Approximative original edge-count test for categorical data
        using union

data:  banque and synth.n
z = -22.964, p-value < 2.2e-16
alternative hypothesis: The distributions of banque and synth.n are unequal.
```

The standardized test statistic value is very small considering the asymptotic normal distribution of the test statistic. This means that the observed edge count is considerably lower than what we would expect for data from the same distribution, which indicates notable differences between the datasets. Consequently, the test rejects the null hypothesis that the datasets follow the same distribution. Therefore, we can conclude that our naive approach generated a dataset that is dissimilar from the original one. Next, we compare the original dataset and the one generated using **synthpop** to see whether this more sophisticated approach lead to a synthetic dataset closer to the original one.

```
R> DataSimilarity(banque, synth.p$syn, method = "FR_cat",
+                 dist.fun = function(x) cluster::daisy(x, metric = "gower"))

        Approximative original edge-count test for categorical data
        using union

data:  banque and synth.p$syn
z = 0.33514, p-value = 0.6312
alternative hypothesis: The distributions of banque and synth.p$syn are unequal.
```

The standardized test statistic value is positive and close to zero. Therefore, the edge count is close to what would be expected when assuming that the datasets are generated from the same distribution. This is also reflected by the test not rejecting the null of equal distributions. So, as expected, the synthetic dataset generated by **synthpop** is clearly more similar to the original dataset than the one generated with the naive approach. In a real application, we would ideally synthesize the dataset repeatedly with both approaches and compare the resulting similarity values of all repetitions but this would go beyond the scope of this illustration. Moreover, we would inspect the generated datasets in more detail, checking for example their marginal distributions, or correlation structures, looking for implausible observations, or depending on the application also taking into account other aspects like data privacy concerns.

Acknowledgments

This work has been supported (in part) by the Research Training Group "Biostatistical Methods for High-Dimensional Data in Toxicology" (RTG 2624, Project P1) funded by the

Deutsche Forschungsgemeinschaft (DFG, German Research Foundation - Project Number 427806116).
We would like to thank Nabarun Deb and Bodhisattva Sen for allowing us to use their R implementation of their test for the package. Moreover, we would like to thank David Alvarez-Melis, whose Python implementation of the OTDD was the basis for our R implementation.

References

Alvarez-Melis D, Fusi N (2020a). "Geometric Dataset Distances via Optimal Transport." In *Advances in Neural Information Processing Systems*, volume 33, pp. 21428–21439. Curran Associates, Inc.

Alvarez-Melis D, Fusi N (2020b). "Measuring dataset similarity using optimal transport." URL https://www.microsoft.com/en-us/research/blog/measuring-dataset-similarity-using-optimal-transport/.

Dray S, Dufour AB (2007). "The ade4 Package: Implementing the Duality Diagram for Ecologists." *Journal of Statistical Software*, **22**(4), 1–20. doi:10.18637/jss.v022.i04.

Fisher RA (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics*, **7**(2), 179–188. ISSN 2050-1439. doi:10.1111/j.1469-1809.1936.tb02137.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

Friedman JH, Rafsky LC (1979). "Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests." *The Annals of Statistics*, **7**(4), 697–717. ISSN 0090-5364.

Hediger S, Michel L, Näf J (2022). "On the Use of Random Forest for Two-Sample Testing." *Computational Statistics & Data Analysis*, **170**, 107435. ISSN 0167-9473. doi:10.1016/j.csda.2022.107435. URL https://www.sciencedirect.com/science/article/pii/S0167947322000159.

Hill AA, LaPan P, Li Y, Haney S (2007). "Impact of Image Segmentation on High-Content Screening Data Quality for SK-BR-3 Cells." *BMC Bioinformatics*, **8**(1), 340. ISSN 1471-2105. doi:10.1186/1471-2105-8-340. URL https://doi.org/10.1186/1471-2105-8-340.

Kuhn, Max (2008). "Building Predictive Models in R Using the caret Package." *Journal of Statistical Software*, **28**(5), 1–26. doi:10.18637/jss.v028.i05. URL https://www.jstatsoft.org/index.php/jss/article/view/v028i05.

Lopez-Paz D, Oquab M (2017). "Revisiting Classifier Two-Sample Tests." In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=SJkXfE5xx.

Mukherjee S, Agarwal D, Zhang NR, Bhattacharya BB (2022). "Distribution-Free Multisample Tests Based on Optimal Matchings With Applications to Single Cell Genomics." *Journal of the American Statistical Association*, **117**(538), 627–638. ISSN 0162-1459. doi:10.1080/01621459.2020.1791131.

Nowok B, Raab GM, Dibben C (2016). "synthpop: Bespoke Creation of Synthetic Data in R." *Journal of Statistical Software*, **74**(11), 1–26. doi:10.18637/jss.v074.i11.

Ntoutsi I, Kalousis A, Theodoridis Y (2008). "A General Framework for Estimating Similarity of Datasets and Decision Trees: Exploring Semantic Similarity of Decision Trees." In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, pp. 810–821. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-654-2. doi:10.1137/1.9781611972788.73.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rosenbaum PR (2005). "An Exact Distribution-Free Test Comparing Two Multivariate Distributions Based on Adjacency." *Journal of the Royal Statistical Society B*, **67**(4), 515–530. ISSN 1369-7412.

Stolte M, Kappenberg F, Rahnenführer J, Bommert A (2024). "Methods for Quantifying Dataset Similarity: A Review, Taxonomy and Comparison." *Statistics Surveys*, **18**, 163–298. ISSN 1935-7516. doi:10.1214/24-SS149.

Stolte M, Rahnenführer J, Bommert A (2026a). "An Empirical Comparison of Methods for Quantifying the Similarity of Categorical Datasets." doi:10.48550/arXiv.2604.11458. URL https://arxiv.org/abs/2604.11458.

Stolte M, Rahnenführer J, Bommert A (2026b). "An Empirical Comparison of Methods for Quantifying the Similarity of Numeric Datasets." doi:10.48550/arXiv.2604.12327. URL https://arxiv.org/abs/2604.12327.

Sutherland JJ, Weaver DF (2004). "Three-Dimensional Quantitative Structure-Activity and Structure-Selectivity Relationships of Dihydrofolate Reductase Inhibitors." *Journal of Computer-Aided Molecular Design*, **18**(5), 309–331. ISSN 0920-654X. doi:10.1023/b:jcam.0000047814.85293.da.

Zhang J, Chen H (2022). "Graph-Based Two-Sample Tests for Data with Repeated Observations." *Statistica Sinica*, **32**(1), 391–415. ISSN 1017-0405. Publisher: Institute of Statistical Science, Academia Sinica, URL https://www.jstor.org/stable/27108529.

Zhu Y, Chen H (2024). "Limiting distributions of graph-based test statistics on sparse and dense graphs." *Bernoulli*, **30**(1), 770–796. ISSN 1350-7265. doi:10.3150/23-BEJ1616. Publisher: Bernoulli Society for Mathematical Statistics and Probability, URL https://projecteuclid.org/journals/bernoulli/volume-30/issue-1/Limiting-distributions-of-graph-based-test-statistics-on-sparse-and/10.3150/23-BEJ1616.full.

Affiliation:

Marieke Stolte, Luca Sauer, Jörg Rahnenführer, Andrea Bommert
Department of Statistics
TU Dortmund University

Vogelpothsweg 87
44227 Dortmund, Germany
and
Marieke Stolte
Institute for Medical Information Processing, Biometry, and Epidemiology (IBE)
LMU Medizin
Ludwig-Maximilians-Universität München
Marchioninistrasse 15
81377 Munich, Germany
E-mail: marieke.stolte@ibe.med.uni-muenchen.de