Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

Dimension reduction for outlier detection using DOBIN

Sevvandi Kandanaarachchi & Rob J. Hyndman

Dimension reduction for outlier detection using DOBIN

Sevvandi Kandanaarachchi[1], Rob J. Hyndman[2]

[1]School of Science, Mathematical Sciences, RMIT University, Melbourne VIC 3000, Australia.

[2]Department of Econometrics and Business Statistics, Monash University, Clayton VIC 3800, Australia.

Corresponding author Sevvandi Kandanaarachchi

sevvandi.kandanaarachchi@monash.edu

Abstract

This paper introduces DOBIN, a new approach to select a set of basis vectors tailored for outlier detection. DOBIN has a simple mathematical foundation and can be used as a dimension reduction tool for outlier detection tasks. We demonstrate the effectiveness of DOBIN on an extensive data repository, by comparing the performance of outlier detection methods using DOBIN and other bases. We further illustrate the utility of DOBIN as an outlier visualization tool. The R package *dobin* implements this basis construction.

Key words—outlier detection, dimension reduction, outlier visualization, basis vectors

1 Introduction

Outlier detection is used in diverse applications such as the identification of extreme weather events, stock market crashes and fraudulent transactions. A common challenge in many domains is that a data point may be an outlier in the original high dimensional space, but not an outlier in the low dimensional subspace created by a dimension reduction method.

We address this challenge by proposing a set of basis vectors tailored for unsupervised outlier detection which we call DOBIN: a Distance based Outlier BasIs using Neighbors. The acronym DOBIN is inspired from the informal verb ‘dob in’, which the Collins English dictionary defines as ‘to inform against or report, specially to the police'. We emphasize that DOBIN is not an outlier detection method; rather it is a pre-processing step that can be used by any outlier detection method.

To the best of our knowledge no formal dimension reduction techniques exist for outlier detection. It is common to use Principal Component Analysis (PCA) where dimension reduction is used when detecting outliers (e.g., Talagala et al., 2019; Hyndman et al., 2015). However, the goal of PCA is to find a set of basis vectors that explains the variance of the data, such that the highest variance is in the direction of the first vector, and so on. Therefore, the PCA basis may not be suited for detecting outliers. By introducing DOBIN, we aim to bridge this gap in the literature.

We envisage two uses of DOBIN. First, as a basis more conducive to outlier detection, it brings outliers to the forefront using a smaller number of components. In this sense, it is somewhat similar to subspace outlier detection (e.g., Aggarwal and Yu, 2001; Keller et al., 2012), although DOBIN does not detect outliers directly. If outliers arise in the full input space, DOBIN enables the outlier detection algorithm to detect outliers using fewer components. The second use of DOBIN is to assist with visualization of outliers as we will see in Section 4.

The literature on outlier detection has evolved in two fields, with contributions from statisticians and computer scientists. Although the methodologies used by the two groups are somewhat different, both groups have contributed to a rich and diverse literature, benefiting the broader community. Useful summaries of the computer science literature are provided by Goldstein and Uchida (2016) and Zimek et al. (2012); while similar surveys of the statistics outlier literature are given in Rousseeuw and Leroy (2005) and Unwin (2019 *a*). We have included methods from both disciplines in this paper.

Some methods treat outlier identification as a binary rule, with observations classified as outliers or otherwise (e.g., Billor et al., 2000; Wilkinson, 2017; Rousseeuw and Bossche, 2018), while other methods rank all observations in terms of “outlyingness” (e.g., Breunig et al., 2000; Liu et al., 2008), or provide a probability of an observation being an outlier (e.g., Kriegel et al., 2009). This difference impacts the performance evaluation metrics. Metrics such as false negatives and false positives or positive predictive value and negative predictive value are used to evaluate the binary results (Wilkinson, 2017). By contrast, having a ranking of the observations leads to use tools such as area under the Receiver Operator Characteristic (ROC) curve, area under the Precision-Recall (PR) curve or precision at n to evaluate performance (Campos et al., 2016).

In Sections 3 and 5 we investigate the effect of DOBIN on three outlier detection methods which first appeared in the computer science literature: LOF (Breunig et al., 2000), KNN (Ramaswamy et al., 2000) and iForest (Liu et al., 2008). To evaluate their performance, we use area under the ROC curve. We examine the effectiveness of DOBIN by comparing it against four other bases: 2 PCA bases, the original basis and a kurtosis maximizing basis (Tyler et al., 2009). In Section 3 we use synthetic data to conduct experiments and in Section 5 we use real data from our repository of more than 12, 000 datasets (Kandanaarachchi, Muñoz Acosta, Smith-Miles and Hyndman, 2019).

Section 4 examines outlier detection methods developed by the statistical community on some associated datasets. We use O3 plots by Unwin (2019 *a*), which can be used to compare the results of six different outlier detection methods: *HDoutliers* (Wilkinson, 2017), *mvBACON* (Billor et al., 2000), *adjOutlyingness* (Brys et al., 2005), *covMcd* (Rousseeuw and Driessen, 1999), *FastPCs* (Vakili and Schmitt, 2014) and *DetectDeviatingCells* (Rousseeuw and Bossche, 2018). We use O3 plots in conjunction with the first two DOBIN components of each dataset and visually inspect if the outliers identified by the O3 plots are further away from the other points in the DOBIN space. This section highlights the usage of DOBIN as an outlier visualization tool.

We have produced an R package *dobin* (Kandanaarachchi, 2019), which computes the DOBIN basis. In addition, all examples in this paper are available in the supplementary materials at https://github.com/sevvandi/Outlier-Basis. We start the next section with the mathematical framework of DOBIN.

2 Mathematical Framework

Before we begin describing the mathematical notation, we explain our intuition behind this specific basis construction. Our goal is to find a basis that maximizes *k* nearest neighbor (knn) distances. The reason for this choice is that knn distances are a simple and effective method of detecting outliers (Ramaswamy et al., 2000). Indeed, certain outlier detection evaluation studies (Campos et al., 2016; Kandanaarachchi, Muñoz, Hyndman and Smith-Miles, 2019; Goldstein and Uchida, 2016) found that knn and weighted knn distances performed better than most outlier detection methods. Furthermore, researchers have used knn distances as a springboard to develop new outlier detection methods (Wilkinson, 2017). Armed with this knowledge, we find a set of basis vectors such that the first vector is in the direction of the highest knn distances, the second vector corresponds to the second highest knn distances, and so on.

Let $X_{N\times p}$ be a matrix denoting a dataset of *N* observations and *p* attributes. Let us denote the i^{th} row of *X* by $\boldsymbol{x}_i$. The distance between the observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be written as

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top S(\boldsymbol{x}_i - \boldsymbol{x}_j), \quad (1)$$

where *S* is a symmetric positive definite matrix. Thus, matrix *S* in equation (1) is diagonalizable, giving us motivation for considering a diagonal *S* with $S = \text{diag}(s_1, s_2, \ldots, s_p)$. Using this we obtain

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \left\langle \eta, (\boldsymbol{x}_i - \boldsymbol{x}_j)^\circ(\boldsymbol{x}_i - \boldsymbol{x}_j) \right\rangle, \qquad (2)$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product in $\mathbb{R}^p$, $\eta = (s_1, s_2, \ldots, s_p)^\top$, and $\circ$ denotes the element-wise vector product.

2.1 The *Y* space

Next we establish the Y space, which is instrumental in the construction of DOBIN. Let $\boldsymbol{y}_{ij} = (\boldsymbol{x}_i - \boldsymbol{x}_j)\circ(\boldsymbol{x}_i - \boldsymbol{x}_j) = \left((x_{i1}-x_{j1})^2, (x_{i2}-x_{j2})^2, \ldots, (x_{ip}-x_{jp})^2\right)$. Substituting in equation (2) gives

$$\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \langle \eta, \boldsymbol{y}_{ij}\rangle. \qquad (3)$$

By finding the appropriate η of unit length, we can maximize the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. This prompts the question as to which $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ need to be chosen to maximize the distance between them, if outlier detection is the goal. As we use knn distances as a guiding principle, we are interested in the k nearest neighbors of a given point. Consequently, we can choose $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that share a common neighborhood.

If we were to calculate $\boldsymbol{y}_{ij}$ for all pairs, it would be computationally expensive as there are $N(N+1)/2$ pairs to consider. In addition, this would give us pairwise distances between points that we are not interested in, such as points on the boundary of the point cloud, that are opposite each other. Finding η that maximizes the distances between such points is not beneficial for outlier detection, because a large distance between polar-opposite points does not mean that either of those points are outliers. Therefore we do not compute $\boldsymbol{y}_{ij}$ for all pairs of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$; rather we select the pairs that we want to include in our Y space.

For each point $\boldsymbol{x}_i$ we compute $\boldsymbol{y}_{ij}$ arising from a set a nearest neighbors. From Kandanaarachchi, Muñoz, Hyndman and Smith-Miles (2019) we know that nearest neighbors depend on the normalization technique. As a result, we have a choice of two normalization techniques to pre-process the data; Min-Max and Median-IQR. Min-Max normalization scales each column to values between 0 and 1, with the minimum mapped to 0 and the maximum mapped to 1. Median-IQR scales each column to median 0 and IQR 1. The choice of normalization is a parameter with a default value of Min-Max as it was shown that Min-Max achieves better performance for most outlier detection methods (Kandanaarachchi, Muñoz, Hyndman and Smith-Miles, 2019). Let X_1 denote the normalized data using Min-Max or Median-IQR. We compute the set of nearest neighbors from 1 to k for X_1 where k is a parameter. Using the nearest neighbors we construct a list of pairs denoted by T. For example, if the nearest neighbors of $\boldsymbol{x}_i$ are $\{\boldsymbol{x}_2, \boldsymbol{x}_5, \boldsymbol{x}_9\}$, the pairs $(i,2), (i,5), (i,9)$, will be in T.

For each pair in T, we compute $\boldsymbol{y}_{ij}$ as in equation (2) on the normalized space X_1. We now have the initial Y space: a set of $\tilde{M}$ points in $\mathbb{R}^{p+}$. As the list of pairs that give rise to the initial Y space are contained in T, we can denote $Y = \{\boldsymbol{y}_\ell\}_{\ell=1}^{\tilde{M}} = \left\{\left(y_{l1}, \ldots, y_{lp}\right)\right\}_{l=1}^{\tilde{M}}$, where each $\boldsymbol{y}_\ell = \boldsymbol{y}_{i_{(\ell)} j_{(\ell)}}$, i.e. ℓ is the row number in the matrix Y and each $\boldsymbol{y}_\ell$ comes from the ℓ^{th} pair in T.

We are interested in neighboring distances that are relatively large. If we consider the sum of knn distances as a measure of outlyingness, data points anywhere between the top $0-5\%$ distances may be labeled as outliers or marked for further scrutiny. We mimic this procedure in our Y space construction. The Euclidean distance squared between two points in the X_1 space can be written as $\sum_{m=1}^{p} y_{\ell m}$ for the appropriate ℓ. In order to consider only points with relatively large distances, we remove points that contribute to distances below a certain threshold determined by a percentile, i.e. we remove points such that $\sum_{m=1}^{p} y_{\ell m} < Q$ where Q is the q^{th} percentile of $\left\{\sum_{m=1}^{p} y_{\ell m}\right\}_{\ell=1}^{\tilde{M}}$. We remove the associated pairs from T as well. This constitutes the Y space: $Y = \{\boldsymbol{y}_\ell\}_{\ell=1}^{M}$. We illustrate this with a toy example having 6 data points. For simplicity, let us consider only five nearest neighbors and let the q^{th} percentile be 19. The knn distances which do not contribute to the Y space are crossed out in Table 1. The associated pairs of points are removed from the Y space. Thus, by removing these point pairs we are putting a spotlight on point pairs with high knn distances, emulating knn outlier detection method. This example is for illustration purposes only as finding outliers in a dataset of 6 data points with $k = 5$ is not a meaningful exercise.

The main advantage of the Y space is that it makes distance computation between two points in the X space a linear function. This makes maximizing distances between points a much easier task. Algorithm 1 summarizes the construction of the Y space.

Algorithm 1: Construction of the Y space.

input: X, $k \in \mathbb{Z}^+$, $q \in (0,1)$ and choice of normalization.

output: The space Y consisting of all $\{\boldsymbol{y}_\ell\}_{\ell=1}^{M}$ and the associated indices i and j.

1 Normalize X using Min-Max or Median-IQR normalization according to the choice parameter. Let us call this space X_1.

2 Find k nearest neighbors for each point in X_1.

3 Let $T =$ the set of i and j indices as in equation (1), corresponding to the pairs of points in X_1.

4 For all these pairs, compute $\boldsymbol{y}_{ij}$. Let $Y = \{\boldsymbol{y}_1 = \boldsymbol{y}_{i_{(1)}j_{(1)}}, \boldsymbol{y}_2 = \boldsymbol{y}_{i_{(2)}j_{(2)}}, \ldots, \boldsymbol{y}_{\tilde{M}} = \boldsymbol{y}_{i_{(\tilde{M})}j_{(\tilde{M})}}\}$, where $(i_{(1)}, j_{(1)})$ is the first $(i, j)^{\text{th}}$ pair in T. Consequently Y is an $\tilde{M} \times p$ matrix, with the ℓ^{th} row denoted by $\boldsymbol{y}_\ell$.

5 Let Q be the q^{th} percentile of $\left\{\sum_{m=1}^{p} y_{\ell m}\right\}$. The quantity $\sum_{m=1}^{p} y_{\ell m}$ is the distance between points $\boldsymbol{x}_{i_{(\ell)}}$ and $\boldsymbol{x}_{j_{(\ell)}}$ in X_1 space.

6 Remove points for which $\sum_{m=1}^{p} y_{\ell m} < Q$.

7 Remove the associated pairs of (*i*, *j*) from *T*.

8 The remaining points constitute the *Y* space: $Y = \{\boldsymbol{y}_\ell\}_{\ell=1}^{M}$.

Figure 1 illustrates the *X* and *Y* spaces where *X* is a 2 dimensional normal distribution of 100 points with a single outlier depicted in red. We see that the *Y* space consists of a lesser number of points compared to *X*, of which 6 points are contributed by the outlier. This is because in the construction of the *Y* space 5 neighbors are considered for each point with $k_1 = 1$ to $k_2 = 5$, and the points $\boldsymbol{y}_k$ with relatively small distances $\left(\sum_i y_{ki} \leq Q\right)$ are removed. Thus, the effect of outliers is magnified in the *Y* space.

The *Y* space computes relative vectors between *k* nearest neighbors. As such, it is not designed to find the position vector of an outlier. As we discuss later, we find the direction of outlyingness by computing the average of these vectors so that the outliers contribute considerably more to this direction than non-outliers clustered near the origin. Furthermore, the *Y* space is not rotationally invariant; it depends on the input coordinates. Performing PCA before computing the *Y* space gives rotational invariance to the overall algorithm. However, we have not adopted this approach as we loose some interpretability.

We summarize the salient features of the *Y* space below:

1. $Y = \{\boldsymbol{y}_\ell\}_{\ell=1}^{M}$, where $\boldsymbol{y}_\ell \in \mathbb{R}^{p+}$.
2. Each $\boldsymbol{y}_\ell \in Y$ is constructed from two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in *X* with $T(\ell) = (i, j)$, where *T* is a list consisting of the pairs in *X* that are used to construct *Y*. In addition, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are *k*-nearest neighbors for some $k \in \mathbb{N}$.
3. For i, j, ℓ as above, $\boldsymbol{y}_\ell = (\boldsymbol{x}_i - \boldsymbol{x}_j)\circ(\boldsymbol{x}_i - \boldsymbol{x}_j)$, where $\circ$ denotes the element-wise vector product.

4. The Y space contains points $\boldsymbol{y}_\ell$ with relatively high $\sum_{m=1}^{p} y_{\ell m}$, i.e. it contains vectors that contribute to high knn distances in the X space.

2.2 Maximizing the distance between points

Restating equation (3) using $T(\ell) = (i, j)$ we obtain

$$\sum_{(i,j)\in T} \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle. \qquad (4)$$

Thus the total distance squared for relatively high nearest neighbor distances $\sum_{(i,j)\in T} \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2$ can be maximized by finding the appropriate η which maximizes $\sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle$. However, (4) depends on the scale of η, so we normalize η such that $(\eta(=1$. We now state our optimization problem as

$$\max_{\eta} \sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle \qquad \text{subject to } (\eta(^2 = 1. \qquad (5)$$

As

$$\sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle = \left\langle \eta, \sum_{\ell} \boldsymbol{y}_\ell \right\rangle,$$

$$\leq \left| \left\langle \eta, \sum_{\ell} \boldsymbol{y}_\ell \right\rangle \right|,$$

$$\leq (\eta(\left\| \sum_{\ell} \boldsymbol{y}_\ell \right\| \quad \text{using Cauchy-Schwarz inequality,}$$

$$\text{we obtain} \quad \left\langle \eta, \sum_{\ell} \boldsymbol{y}_\ell \right\rangle \leq \left\| \sum_{\ell} \boldsymbol{y}_\ell \right\|,$$

$$\text{giving us} \quad \max_{\eta} \left\langle \eta, \sum_{\ell} \boldsymbol{y}_\ell \right\rangle = \left\| \sum_{\ell} \boldsymbol{y}_\ell \right\|.$$

With the standard inner product in $\mathbb{R}^n$, equality can only occur when $\eta = c \sum_{\ell} \boldsymbol{y}_\ell$. As η is a unit vector we obtain

$$\eta = \frac{\sum_{\ell=1}^{M} \boldsymbol{y}_\ell}{\left\| \sum_{\ell=1}^{M} \boldsymbol{y}_\ell \right\|}. \quad (6)$$

We can also arrive at the same solution using Lagrange Multipliers. For

$$f(\eta) = \sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle \qquad \text{and} \qquad g(\eta) = \|\eta\|^2 - 1 = 0,$$

we solve $\nabla f = \lambda \nabla g$. As $\nabla f = \left(\frac{\partial f}{\partial \eta_1}, \frac{\partial f}{\partial \eta_2}, \cdots, \frac{\partial f}{\partial \eta_p} \right)$ we compute

$$\frac{\partial f}{\partial \eta_j} = \frac{\partial}{\partial \eta_j} \sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle = \frac{\partial}{\partial \eta_j} \sum_{\ell=1}^{M} \sum_{i=1}^{p} \eta_i y_{\ell i} = \sum_{\ell=1}^{M} \sum_{i=1}^{p} \delta_{ij} y_{\ell i} = \sum_{\ell=1}^{M} y_{\ell j},$$

where δ_{ij} is the Kronecker delta function which equals 1 when $i = j$ and zero otherwise. Hence

$$\nabla f = \left(\sum_{\ell=1}^{M} y_{\ell 1}, \sum_{\ell=1}^{M} y_{\ell 2}, \ldots, \sum_{\ell=1}^{M} y_{\ell p} \right) = \sum_{\ell=1}^{M} \boldsymbol{y}_\ell .$$

Similarly, as $g(\eta) = \left(\sum_{j=1}^{p} \eta_j^2 \right) - 1$, we get $\nabla g = \eta$. By substituting $\nabla f = \lambda \nabla g$ and using $\|\eta\| = 1$, we obtain the same solution as in equation (6).

2.3 Constructing a basis

We have computed the vector η that maximizes $\sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle$. To find a basis, we need to find the second, third and subsequent vectors which are orthogonal and in some way maximize the quantity $\sum_{\ell} \langle \eta, \boldsymbol{y}_\ell \rangle$. Let us first rename η as η_1. Taking η_1 as the first basis vector, we take the projection of $\boldsymbol{x}_\ell$ on to η_1 and remove these components from $\boldsymbol{x}_\ell$:

$$\boldsymbol{x}_{\ell_1} = \boldsymbol{x}_\ell - \langle \eta_1, \boldsymbol{x}_\ell \rangle \eta_1 .$$

Thus $\boldsymbol{x}_{\ell_1} \perp \eta_1$. Let $X_1 = \{\boldsymbol{x}_{\ell_1}\}_{\ell=1}^{N}$ be the set of remaining components of $\{\boldsymbol{x}_\ell\}_{\ell=1}^{N}$ after removing the projection on to η_1. We compute the Y space using X_1 and find

$$\eta_2 = \frac{\sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_1}}{\left\| \sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_1} \right\|}.$$

Proceeding in this way we obtain

$$\boldsymbol{x}_{\ell_b} = \boldsymbol{x}_{\ell_{b-1}} - \langle \eta_b, \boldsymbol{x}_{\ell_{b-1}} \rangle \eta_b,$$

and after computing the Y space for $\boldsymbol{x}_{\ell_b}$ we compute

$$\eta_{b+1} = \frac{\sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_b}}{\left\| \sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_b} \right\|},$$

with $\boldsymbol{x}_{\ell_0} = \boldsymbol{x}_\ell$. The set $\Theta = \{\eta_1, \eta_2, \ldots, \eta_p\}$ gives a basis for Y with each η_{i+1} maximizing $\sum_{\ell} \langle \eta, \boldsymbol{y}_{\ell_i} \rangle$. This constitutes the central computation of the basis construction. This process is somewhat similar to Gram Schmidt orthogonalisation (Anton and Rorres, 2013), which computes an orthonormal basis for a given set of basis vectors using orthogonal complements. In our case, we find each basis vector using the orthogonal complement of the previous basis vectors.

We note that each subsequent X_j lies in a lower dimensional subspace compared to the original X space, because X_j constitutes of the remaining vectors after removing the projections on to $\{\eta_1, \eta_2, \ldots, \eta_j\}$. Consequently, it is not optimal to use the original set of coordinates to describe vectors in X_j, even though $X_{N\times p}$ was described using coordinates in $\mathbb{R}^p$. Therefore, for each X_j we change the basis to that of a lower dimensional space, compute the *Y* space and find η_{j+1} using this basis and transform η_{j+1} back to the original basis. We explain this process next.

2.3.1 Same vectors, two bases

Consider scalars $\alpha, \beta \in \mathbb{R}$, vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$, where *V* is a vector space, and two bases *B* and *C* for *V*. Let $\boldsymbol{u}^B$ denote the coordinates of ***u*** using *B* and $\boldsymbol{u}^C$ the coordinates using *C*. Suppose

$$\boldsymbol{u} = \alpha \boldsymbol{v} + \beta \boldsymbol{w}. \quad (7)$$

As illustrated in Figure 2, equation (7) holds true independent of the coordinate system. If we evaluate Equation (7) using basis *B*, then

$$\boldsymbol{u}^B = \alpha \boldsymbol{v}^B + \beta \boldsymbol{w}^B,$$

and we obtain the coordinates of ***u*** in basis *B*. Alternatively we can evaluate equation (7) using basis *C* and obtain

$$\boldsymbol{u}^C = \alpha \boldsymbol{v}^C + \beta \boldsymbol{w}^C,$$

giving coordinates of ***u*** in basis *C*. Both coordinate representations $\boldsymbol{u}^B$ and $\boldsymbol{u}^C$ describe the vector ***u***.

Now we consider the change of basis mentioned in Section 2.3.

2.3.2 Step 1 - Changing coordinates of $\boldsymbol{x}_{\ell_1}$

Let *E* denote the standard Euclidean basis in $\mathbb{R}^p$ that describes the original vectors $\boldsymbol{x}_\ell \in X$. After removing the projection of $\{\boldsymbol{x}_\ell\}_{\ell=1}^N$ on to η_1, the remaining vectors $\{\boldsymbol{x}_{\ell_1}\}_{\ell=1}^N$ of X_1 lie in a subspace having a maximum dimension of $p - 1$. As such, these vectors can be described using $p - 1$ coordinates. In fact, as

$$\left\langle \eta_1, \boldsymbol{x}_{\ell_1} \right\rangle = \eta_1^T \boldsymbol{x}_{\ell_1} = 0,$$

$\boldsymbol{x}_{\ell_1}$ lies in the null space of η_1^T, that is $\mathrm{Null}(\eta_1^T)$. As such, we can use the standard orthonormal basis of $\mathrm{Null}(\eta_1^T)$, denoted by $B_1 = \{\gamma_1, \ldots, \gamma_{p-1}\}$ to describe $\{\boldsymbol{x}_{\ell_1}\}_{\ell=1}^{N}$. Here B_1 is computed using the E basis with $\gamma_i^E \in \mathbb{R}^p$ and $\gamma_i^{B_1} = (0,\ldots,1,\ldots,0) \in \mathbb{R}^{p-1}$. Using the fact that B_1 is an orthonormal basis for $\mathrm{Null}(\eta_1^T)$ and $\boldsymbol{x}_{\ell_1} \in \mathrm{Null}(\eta_1^T)$, we express $\boldsymbol{x}_{\ell_1}$ as a linear combination of γ_i:

$$\begin{aligned} \boldsymbol{x}_{\ell_1} &= \left\langle \boldsymbol{x}_{\ell_1}, \gamma_1 \right\rangle \gamma_1 + \cdots + \left\langle \boldsymbol{x}_{\ell_1}, \gamma_{p-1} \right\rangle \gamma_{p-1}, \\ &= a_1\gamma_1 + a_2\gamma_2 + \cdots + a_{p-1}\gamma_{p-1}, \end{aligned} \qquad (8)$$

where we have used the standard inner product. This is illustrated in Figure 3. The vector equation (8) holds in any coordinate system. This gives us the new coordinates of $\boldsymbol{x}_{\ell_1}$ in B_1 basis

$$\boldsymbol{x}_{\ell_1}^{B1} = \left(\left\langle \boldsymbol{x}_{\ell_1}, \gamma_1 \right\rangle, \ldots, \left\langle \boldsymbol{x}_{\ell_1}, \gamma_{p-1} \right\rangle \right)^T = \left(a_1, \ldots, a_{p-1} \right)^T.$$

Let $X_1^{B_1}$ denote the new coordinates of X_1 in the basis B_1. Then

$$X_1^{B_1} = X_1 B_1,$$

where $\boldsymbol{x}_{\ell_1}$ is a row vector of X_1 and γ_j is a column vector of B_1 making the product $X_1^{B_1}$ an $N \times (p-1)$ matrix. This completes the first transformation we talk about in Section 2.3, i.e. we transform $\boldsymbol{x}_{\ell_1}$ from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$ using $\mathrm{Null}(\eta_1^T)$.

2.3.3 Step 2 - Changing coordinates of η_2 back to $\mathbb{R}^p$

Next we repeat the Y space construction for $X_1^{B_1}$ and find the vector η_2 using B_1 basis:

$$\eta_2^{B_1} = \frac{\sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_1}^{B_1}}{\left\| \sum_{\ell=1}^{M} \boldsymbol{y}_{\ell_1}^{B_1} \right\|},$$

Let $\eta_2^{B_1} = \left(\zeta_1, \ldots, \zeta_{p-1} \right)^T$, then

$$\eta_2 = \zeta_1\gamma_1 + \cdots + \zeta_{p-1}\gamma_{p-1}. \qquad (9)$$

Equation (9) is a vector equation that holds in any coordinate system. If we evaluate equation (9) using the original basis *E*,

$$\begin{aligned} &\eta_2^E = \zeta_1\gamma_1^E + \cdots + \zeta_{p-1}\gamma_{p-1}^E, \\ &\text{giving us} \qquad \eta_2^E = B_1\eta_2^{B_1}, \end{aligned} \qquad (10)$$

as the matrix *B*$_1$ contains the column vectors γ_i^E. This constitutes transforming η_{j+1} back to the original basis *E*.

2.3.4 Step 3 - Repeat Steps 1 and 2

We repeat Steps 1 and 2 for the lower dimensional space $X_1^{B_1}$, which contain $\boldsymbol{x}_{l_1} \in \mathbb{R}^{p-1}$. We remove the projection of $\{\boldsymbol{x}_{l_1}^{B_1}\}_{l_1=1}^N$ on to $\eta_2^{B_1}$ and continue Steps 1 and 2 and find $\eta_3^{B_2}$, where *B*$_2$ is an orthonormal basis for the null space of $\left(\eta_2^{B_1}\right)^T$. When changing the coordinates of $\eta_3^{B_2}$ to η_3^E, we need to change coordinates from *B*$_2$ to *B*$_1$ first and then from *B*$_1$ to *E*. Similar to equation (25)

$$\begin{aligned} &\eta_3^{B_1} = B_2\eta_3^{B_2}, \\ &\text{and} \qquad \eta_3^E = B_1\eta_3^{B_1}, \\ &\text{making} \qquad \eta_3^E = B_1B_2\eta_3^{B_2}, \end{aligned}$$

giving us $\{\eta_1, \eta_2, \eta_3\}$ in standard coordinates. Then we start the Steps 1 and 2 again for $X_2^{B_2}$, which contains $\boldsymbol{x}_{l_2} \in \mathbb{R}^{p-2}$. Continuing this way we obtain all the basis vectors.

2.4 New DOBIN coordinates

Once the basis Θ is computed, we change the basis of the original data as follows:

$$\tilde{X} = X_1\Theta \qquad (11)$$

where *X*$_1$ is the normalized *X* space according to the choice of normalization. We note that the basis Θ is not invariant to rotations; i.e. if one were to start with a different basis, the resulting Θ would be different. This is because when constructing the *Y* space we compute element-wise squares of vector differences. While vector addition is basis invariant, squared element-wise components depend on the basis vectors. Notwithstanding this limitation, we achieve good results for experiments conducted on our extensive collection of more than 12, 000 real world datasets. For almost all of our experiments, we constructed DOBIN using the original basis as input. We only used a different basis if the number of variables were much higher than the number of observations. Furthermore, as we will see in Section 4 it is easier to gain insights about the outlying directions when the axes are original variables.

We conclude this mathematical discussion by summarizing the key steps of DOBIN:

1. Find the Y space for a given dataset X as detailed in Algorithm 1.
2. Construct the basis Θ as outlined in Section 2.3.
3. Transform the original space X using equation (29).

2.5 Parameters of DOBIN

We have two numerical parameters k and q for constructing the Y space with the following default values:

$$k = \min\left(20, \max(\lfloor N / 20 \rfloor, 2)\right),$$
$$q = 0.95,$$

where N is the number of observations in the dataset. The parameter k set to be the minimum of 20 and $N/20$, i.e. 5% of N. As outliers are rare observations we have taken 5% of the dataset as the upper limit for the outlier population. For real world applications such as credit card frauds and intrusion detection, this percentage is much lower than 5% (Goldstein and Uchida, 2016; Kaggle and Machine-Learning-Group, 2018). We set a maximum value of 20 for k as it affects the run time of the algorithm.

The parameter q determines the cut-off quantile needed to obtain a subset of $\boldsymbol{y}$ with large distances. Therefore q needs to be a relatively high value in the interval (0, 1). From equation (5) we maximize the projection of $\boldsymbol{y}$ on to η. If q

is too low, many $\boldsymbol{y}$ with smaller projections may contribute heavily to the sum, making DOBIN vectors less effective.

The other parameter for constructing the Y space is the choice of normalization. While Min-Max normalization is commonly used by the computer science community (Campos et al., 2016), other normalization techniques such as distance from univariate medians and Mahalanobis distance are commonly used by the statistical community (Billor et al., 2000). We give two options for normalization: Min-Max and Median-IQR, and use Min-Max as the default method.

3 Experiments with synthetic data

In this section we do five experiments with synthetic data. For each experiment we compare results for three outlier detection methods, namely LOF (Breunig et al., 2000), KNN (Ramaswamy et al., 2000) and isolation forest (Liu et al., 2008). We choose these methods as they are well recognized and fundamentally different: LOF uses a local density based approach; KNN uses a global distance based approach; and iForest uses a randomized tree based approach. For each outlier detection method we consider the following five coordinate systems:

1. All variables in the dataset;
2. Perform Principal Component Analysis (PCA) on the dataset, and use the first half of the principal components (PCs).
3. Perform PCA, and use the second half of the principal components (Amnarttrakul et al., 2011).
4. Use the eigenvectors of the fourth moment matrix as detailed in Tyler et al. (2009) as a basis. The fourth moment matrix is a measure of the kurtosis of a distribution and as such is associated with its tails. We compute the coordinates of the dataset with respect to this basis, and use the first half of these coordinates. This set of coordinates is denoted by COV4 in the rest of the paper.
5. Perform DOBIN on the dataset, and use the first half of the DOBIN components.

For each outlier detection method, we compare the results using these three sets of coordinates. We use area under the Receiver Operator Characteristic curve (AUC) as our performance measure.

3.1 Experiment 1

The first experiment considers two normal distributions: an inlier normal distribution and an outlier normal distribution, which moves out from the inlier distribution as its mean increases. We consider a dataset of 405 observations in $\mathbb{R}^6$, of which 400 observations in each dimension are normally distributed with mean 0 and standard deviation 1. The remaining 5 observations signify outliers and are normally distributed with mean μ and standard deviation 0.2 in one dimension, and mean 0 and standard deviation 1 in other dimensions. The value of μ is changed from 0 to 4.5 by increments of 0.5. The reason for a smaller standard deviation in the first dimension is to ensure that the outliers stay close to the mean μ and move out of the inlier distribution as μ increases. For each value of μ we consider the five sets of coordinates described above and their performance using the outlier methods LOF, KNN and iForest. We perform 10 repetitions of this experiment. Figure 4(a) shows the results for all five outlier methods using the average performance values of 10 repetitions.

We see that DOBIN and COV4 give similar results and perform better than the other methods. As our outlier distribution contributes to the kurtosis, COV4 performs well in this example. Using 3 DOBIN and COV4 components give better performance for all 3 outlier detection methods for $\mu > 1$. Indeed, we are interested in a set of coordinates that can accentuate the outliers as they move out from the inlier distribution. For values of $\mu \in \{0, 0.5, 1\}$, the outlier distribution lies in the interior of the inlier distribution, making a performance comparison between the coordinates not meaningful.

3.2 Experiment 2

The second experiment considers a bi-modal setting with two inlier normal distributions and one outlier normal distribution which moves into the valley as its mean changes. Somewhat similar to the previous example we consider

805 observations in $\mathbb{R}^6$; 800 inlier observations of which 400 centered at (5, 0, 0, 0, 0, 0) and the other 400 centered at (–5, 0, 0, 0, 0, 0). All inlier observations come from normal distributions with standard deviation 1 in each dimension. The outlier distribution consists of 5 points with mean $(5-\mu, 0, 0, 0, 0, 0)$ and standard deviations 0.2 in the first dimension and 1 in other dimensions. The value of μ is increased from 0 to 4.5 in increments of 0.5. Again, a smaller standard deviation is to ensure that the outliers move into the valley. Figure 4(b) shows the results for all three outlier methods on all five coordinate systems using the average values of 10 iterations.

We see that using 3 DOBIN components gives significantly higher performance for LOF and KNN methods compared to the other coordinates. For iForest, DOBIN and COV4 gives similar performance values. However, iForest does not perform particularly well for this example as the best performance for both DOBIN and COV4 is less than 0.9, indicating that the outliers are not ranked in the top 10%.

3.3 Experiment 3

The third experiment is motivated from Zimek et al. (2012). We consider a uniform distribution in $\mathbb{R}^{20}$ and place a manual outlier at 0.9 in i dimensions where i is changed from 1 to 20. While this is not an outlier in any single dimension, as i increases this point stands out from the rest of the distribution. Indeed, for large i this point lies furthest from the rest. An equivalent outlier is 0.1 in i dimensions. The reason that these points are outliers is because they are at the corners of the hyper-cube and are so far away from other points. On the contrary, 0.5 in i dimensions would be closest to other points. This is because $(0.5, 0.5, \dots)$ has neighbors from all sides and has on average 2^{20} more neighbors compared to $(1, 1, \dots)$ which is a corner on the unit hyper-cube. For practical purposes the point $(0.9, 0.9, \dots)$ is similar to $(1, 1, \dots)$ and has significantly fewer neighbors compared to $(0.5, 0.5, \dots)$. Figure 4(c) shows the results of all three outlier methods on all five coordinate systems.

For LOF and KNN, using 10 DOBIN components gives similar performance results as using all 20 variables. The next best performance for LOF and KNN is achieved by using COV4 coordinates. Again, this is an example where outlyingness overlaps with kurtosis as the point $(0.9, 0.9, \ldots, 0.9)$ contributes to a heavy tail. In contrast, the first and the last 10 PC components give much lower results for these two outlier methods. Again, iForest behaves somewhat differently in this example. The COV4 coordinates are not as competitive for iForest as they were for LOF and KNN. For dimensions greater than 8, DOBIN achieves better results for iForest followed closely by all 20 variables. We also see that iForest results are quite noisy compared with LOF and KNN.

3.4 Experiment 4

For this experiment we consider 805 points in $\mathbb{R}^{14}$. In the first two dimensions the points lie in an annulus as shown in Figure 5(a). In other dimensions all points are normally distributed with mean 0 and standard deviation 1. There are 800 non-outliers and 5 outliers, which are distributed normally with mean $5-(i-1)0.5$ and standard deviation 0.1 in the first dimension and mean 0 and standard deviation 0.1 in the second dimension. With each iteration i increases by 1 making the outlier distribution move inward into the annulus. We repeat this experiment 10 times.

We see that for LOF and KNN, DOBIN outperforms all other coordinates by a fair margin. All coordinates perform poorly for iForest, with most scores below 0.5.

3.5 Experiment 5

This experiment is somewhat similar to the previous experiment. We consider 805 points in $\mathbb{R}^{16}$. In the first two dimensions the points lie on a parabola as shown in Figure 5(b). In other dimensions all points are normally distributed with mean 0 and standard deviation 1. Similar to the previous experiments, there are 800 non-outliers and 5 outliers, which are distributed normally with mean 0 and standard deviation 0.1 in the first dimension and mean $(i-1)0.5$ and standard deviation 0.1 in the second dimension. With each iteration i increases by 1 making the outlier distribution move out of the parabola. We repeat this experiment 10 times.

We see that for LOF and KNN, DOBIN outperforms all other coordinates. For iForest there is no clear winner. Even though DOBIN and COV4 perform better than the other coordinates for the last two iterations, the performance values are quite low.

DOBIN is a top performer for all five experiments. For experiments 2, 4 and 5, DOBIN outperforms all other methods by a fair margin for KNN and LOF. For these three experiments iForest gives poorer performance compared to KNN and LOF. For experiment 1, COV4 gives equally good performance results to DOBIN. For experiment 3, COV4 is a close contender to DOBIN. For both these experiments outlyingness coincides with kurtosis, as the outliers are placed in the tails of the distribution. As such, we may expect COV4 to perform well in such instances. Furthermore, LOF and KNN perform better than iForest for most experiments. This may be attributed to the randomness of iForest and the tree structure property, which acts on one attribute at a time. Principal Components are generally less effective than the full set of coordinates for these three experiments and outlier methods. We note that this is not a failure of PCA, as outlier detection is not its intended goal. Rather, PCA maximizes the variance in each component such that a low dimensional representation of the dataset is a good approximation to the original. These results suggest that DOBIN is preferred over COV4 and PCA as a dimension reduction technique for outlier detection.

4 Results with visualization

In this section we investigate specific examples motivated from Unwin (2019 *a*) and Wilkinson (2017) using O3 plots and the DC1-DC2 space, which is the space spanned by the first two DOBIN vectors. O3 plots (Unwin, 2019 *a*) can be used to compare the results of 6 different outlier detection methods: *HDoutliers*, *mvBACON*, *adjOutlyingness*, *covMcd*, *FastPCs*, and *DetectDeviatingCells*. Besides facilitating an ensemble of outlier methods, O3 plots also highlight outliers in axis parallel subspaces. We use O3 plots as a means of validating the DC1-DC2 space.

4.1 *Election2005* dataset

We start with the *Election2005* dataset (Grimm, 2019), which is used by Unwin (2019 *a*) to illustrate the O3 plot. This dataset includes election results of two German elections and certain attributes of the electorates. We examine the O3 plot and the DC1-DC2 space of this dataset.

Figure 6(a) shows the O3 plot of this dataset using all six outlier methods. The columns on the left indicate the variables, the columns on the right indicate the observations, the rows specify the axis parallel subspaces and the colors depict the number of methods that identify each observation in each subspace as an outlier. From this plot we see that observation *X*84 is identified as an outlier by 5 methods in 5 subspaces, 4 methods in 2 subspaces, 3 methods in 1 subspace and by 1 method in 1 subspace. *X*84 is arguably the most outlying observation in this dataset. The observations *X*83, *X*76, *X*82 are also identified as outliers by 5 methods in the dimension of population density. They are also identified as outliers by multiple methods in different subspaces. The layout of the O3 plot is such that the outlyingness of the observations increase to the right.

Figure 6(b) shows the first 2 DOBIN components of the *Election2005* dataset. This is a projection of the dataset onto a two dimensional subspace spanned by the first 2 DOBIN vectors. Here we see the observation *X*84 far away from the rest of the data with *X*76, *X*83 and *X*82 somewhat detached from the rest. The observations *X*87 an *X*81 are almost at the same position in this 2-dimensional space. The observations *X*221 and *X*21 also appear a bit outside the boundary of other points. A simple KNN algorithm in the DC1-DC2 space gives the top 8 candidates as *X*84, *X*76, *X*83, *X*82, *X*221, *X*81, *X*87, and *X*21. Thus, the DC1-DC2 space accentuates the eight most outlying observations according to the O3 plot, while showing clearly that *X*84 is the most outlying observation.

The first DOBIN vector is of interest to us as the outliers deviate most in this direction. Equation (12) gives the first DOBIN vector, DC1, as a linear combination of the input variables.

$$\text{DC1} = \begin{bmatrix} 0.024 & 0.980 & 0.118 & 0.154 \end{bmatrix} \begin{bmatrix} \text{Area} \\ \text{Population Density} \\ \text{Birthrate} \\ \text{Car Ownership} \end{bmatrix} \qquad (12)$$

From equation (12) we see that *Population Density* is the main variable contributing to outliers in the DC1-DC2 space. The O3 plot supports this observation. If we look carefully at the O3 plot we see that *X*21 to *X*84 gets identified as outliers in all subspaces that contain *Population Density*. This insight gives us a better understanding of the dataset.

4.2 *Diamonds* dataset

The next example is taken from the *O3Outliers* R package (Unwin, 2019 *b*) and uses the *Diamonds* dataset (Wickham, 2016). This dataset contains attributes of diamonds.

Figure 7(a) shows the O3 plot of this dataset using the three outlier methods *HDoutliers*, *FastPCs* and *adjOutlyingness*. We could not use all six methods on this dataset as the other three methods included in the *OutliersO3* R package gave computational errors. Of the three methods used, all methods identified *X*4792, *X*2315 and *X*2208 as outliers in 6 subspaces including the full space. These 3 points and *X*4519 appear in the DC1-DC2 space in the bottom left region. Comparing with the O3 plot, we see that *X*4519 is identified as an outlier by 2 methods in 6 subspaces. In keeping with the guidelines of the previous section, we use half of the DC components to evaluate a KNN algorithm. The top 4 candidates in this 3-dimensional DC space are *X*4792, *X*2315, *X*2208 and *X*4519 in that order.

To understand the first DOBIN vector, we look at its coefficients:

$$\text{DC1} = \begin{bmatrix} 0.155 & 0.497 & 0.343 & 0.158 & 0.193 & 0.739 \end{bmatrix} \begin{bmatrix} \text{Carat} \\ \text{Depth} \\ \text{Table} \\ \text{x} \\ \text{y} \\ \text{z} \end{bmatrix}$$

We see that the variable *z* contributes more to outliers in the DC1-DC2 space as well as in the O3 plot.

4.3 *Airquality* dataset

The *Airquality* dataset discussed in Chambers et al. (1983) has measurements on air quality in New York city from May to September 1973. The O3 plot using all 6 outlier methods and the DC1-DC2 space of this dataset are shown in Figure 8.

The O3 plot in Figure 8(a) shows *X*117 as the only outlier identified in 6 subspaces by 2 methods. The associated DC1-DC2 space shows the observation *X*117 away from the rest at the bottom right. Again, we see agreement between the O3 plot and DC1-DC2 space. The observation *X*117 was identified as an outlier only by a third of the outlier methods. This is corroborated in the DC1-DC2 plot by comparatively less outlyingness of observation *X*117 compared to outliers in the *Diamonds* dataset. Performing a KNN algorithm revealed the observation with the highest KNN distance as *X*117 in the DC1-DC2 space.

The first DOBIN vector has the following coefficients:

$$\mathrm{DC1} = \begin{bmatrix} 0.650 & 0.209 & 0.667 & 0.296 \end{bmatrix} \begin{bmatrix} \mathrm{Ozone} \\ \mathrm{Solar.R} \\ \mathrm{Wind} \\ \mathrm{Temp} \end{bmatrix} \qquad (13)$$

From equation (13) we see that the observation *X*117 is an outlier due to high values of *Ozone* and *Wind*.

4.4 *lesmis* dataset

The *lesmis* dataset (Vialaneix et al., 2019) contains the character coappearance network of characters in the novel *Les Misérables* by Victor Hugo. This dataset is stored as a graph, with vertices corresponding to characters and is shown in Figure 9. The character network in *Les Misérables* is also studied in Wilkinson (2017) in the context of outlier detection. Here we conduct a similar study using the O3 plot and the DC1-DC2 space.

As in Wilkinson (2017) we compute the graph-based features centrality, transitivity, closeness, betweenness, degree, average nearest neighbor degree and page rank for each character in the dataset. This transforms the original graph to a rectangular dataset of 77 observations and 7 variables, with each observation corresponding to a character in the novel. Figure 10(a) shows the O3 plot for the transformed *lesmis* dataset using the three methods *HDoutliers*, *mvBACON* and *covMCD*. The other three outlier methods used in O3 plots gave computational errors and so could not be used. Figure 10(b) shows the DC1-DC2 space of the transformed *lesmis* dataset.

From the O3 plot it is evident that Valjean is the main outlier consistently identified by the three outlier methods in the vast majority of the subspaces. This is confirmed by the DC1-DC2 plot as Valjean appears far away from the rest of the characters in the top right corner. The next most outlying characters in the O3 plot are Myriel, Gavroche, Marius and Fantine. These characters also appear in the DC1-DC2 space somewhat detached from the rest of the characters, although none so remarkably far away as Valjean. A KNN algorithm performed on the first half of the DC components identified Valjean, Myriel, Gavroche, Marius and Fantine as having the highest KNN distances.

As the first DOBIN vector is discriminatory, we look at its coefficients: The first DOBIN vector has the following coefficients:

$$\mathrm{DC1} = \begin{bmatrix} 0.0081 & 0.0001 & 0.0050 & 0.9999 & 0.0050 & 0.0040 & 0.0066 \end{bmatrix} \begin{bmatrix} \text{Centrality} \\ \text{Transitivity} \\ \text{Closeness} \\ \text{Betweenness} \\ \text{Degree} \\ \text{Avg. neighbor Degree} \\ \text{Page Rank} \end{bmatrix}.$$

(14)

From equation (14) we see that *Betweenness* is the main feature that makes Valjean an outlier in the DC1-DC2 space. The property *Betweenness* is defined as the number of shortest paths going through a vertex (Csardi and Nepusz, 2006). Thus Valjean becomes a node in many shortest paths in the character network making him the main outlier.

4.5 Classics from Gutenberg

This example is on text analysis of 22 classics downloaded from the Gutenberg project (*Project Gutenberg*, n.d.) and is similar to the example in Wilkinson (2017). We consider the novels *Alice in Wonderland*, *Anna Karenina*, *Bleak House*, *Emma*, *Frankenstein*, *Gullivers Travels*, *Jude the Obscure*, *Lord Jim*, *Mansfield Park*, *Middlemarch*, *Moby Dick*, *Northanger Abbey*, *Persuasion*, *Pride and Prejudice*, *Sense and Sensibility*, *Silas Marner*, *Sons and Lovers*, *The Life and Opinions of Tristram Shandy*, *Wizard of Oz*, *Ulysses*, *Vanity Fair* and *War and Peace* in our analysis.

We strip each novel into words and obtain the collection of words used and their relative frequencies. This set of words need to be cleaned before computing any useful features. We use the R package *tidytext* (Silge and Robinson, 2016) for this task. As the first ‘cleaning’ step, we eliminate the stop words such as *the*, *a*, *an* and *in* from our collection. We also remove numbers from our word collection. Next we use a stemming procedure to reduce words to their word stem. For example the words *run* and *running* have the same word stem. This process gives us a cleaned set of words for each novel. We compute the tf-idf (term frequency-inverse document frequency) measure on this collection. The tf-idf statistic measures how important a word is to a document when considering a collection of documents. Consequently we end up with a rectangular dataset of 22 observations and 39, 624 variables, where each observation is a novel and each variable is the tf-idf statistic of a word.

However, 22 vectors in $\mathbb{R}^{39,624}$ can only span a subspace of 22 dimensions or less. Therefore, we compute this subspace and the respective coordinates of the observations using PCA. We use all the principal components so that we’ re performing a change of basis and not a projection. As a result we obtain a 22 × 22 matrix, where each row is an observation corresponding to a novel and each column is a linear combination of 39, 624 tf-idf statistics. The resulting O3 plot using *HDoutliers* and the corresponding DC1-DC2 space is illustrated in Figure 11. Again we see the O3 plot and the DC1-DC2 space agreeing on the most outlying novel *Ulysses* confirming the findings of Wilkinson (2017). Performing a KNN algorithm yielded *Ulysses* as the observation with the highest KNN distance.

The novel *Ulysses* stands out in the DC1-DC2 space mainly because of its second DC component. However, we do not explore the coefficients of this vector as each coefficient is a linear combination of 39, 624 variables, making it difficult to gain insights.

This section substantiated DOBIN's ability as an outlier visualization tool. In each example we saw the outliers projected onto a 2D space spanned by the first two DOBIN vectors, and their legitimacy was endorsed by the O3 plots. Furthermore, we gained insights about the nature of outlyingness in data by inspecting DOBIN vectors.

5 Results on a data repository

In this section we work with a data repository of 12, 222 datasets (Kandanaarachchi, Muñoz Acosta, Smith-Miles and Hyndman, 2019), generated from approximately 200 source datasets. Each dataset has labeled outliers and originally was used for classification purposes. The present datasets are modified from its original form by downsampling minority class observations, converting categorical attributes to numerical values, removing missing data and creating several variants from each classification dataset. More details on dataset preparation can be found in Kandanaarachchi, Muñoz, Hyndman and Smith-Miles (2019). This collection also includes datasets from Campos et al. (2016).

A data point that is clearly an outlier in a subspace may be difficult to detect as an outlier in the full space, particularly when other variables drown the subspace outlier (Zimek et al., 2012). For example in Figure 11(a), the novel *Alice in Wonderland* is detected as an outlier in the fourth subspace in the O3 plot, but not in others. Conversely, when testing algorithms for outlier detection, we need to consider data where some variables contribute noise that mask outliers.

Therefore, to make these datasets suitable for testing our outlier detection algorithms, we "fatten" each dataset by adding 20 noise variables distributed normally, i.e. $\sim \mathcal{N}(0,1)$. We use these 'fattened' datasets in our experiment.

Similar to Section 3, for each dataset we compare the results of three outlier methods LOF, KNN and iForest using the area under the ROC curve (AUC), on five different coordinate systems, namely the full set of coordinates, the first half of the DOBIN components, the first half of the Principal Components, the last half of Principal Components, and the first half of COV4 coordinates. That is, for a dataset with *p* variables and *N* observations, we consider $\lfloor \min(N/2, p/2) \rfloor$ DOBIN, COV4 and PC components.

Furthermore, we need to select appropriate tools to present results of 12, 222 datasets across three methods and five coordinate systems. The ability to delve into each dataset and obtain insights is no longer feasible with a large repository. In addition, we want to ascertain whether DOBIN coordinates improve the performance of the three outlier detection methods compared to the other four coordinate systems. Therefore, we investigate each outlier detection method separately.

For each dataset we record the performance of each outlier detection method for each coordinate system. If coordinates cannot be computed or if an outlier detection method fails, we record the value as missing. For 1485 datasets COV4 coordinates could not be computed as it involves computing the inverse of the covariance matrix. For some datasets with more variables than observations, i.e. $p > N$, this inverse may not exist. Nevertheless, it is common to have real world datasets with $p > N$.

We consider the set of datasets that have valid performance values, that is we remove datasets having missing values. For each outlier method, we compare the performance of the five coordinate systems using a Friedman test adjusting for the dataset variants. To adjust for the dataset variants, we randomly sample 20 datasets from each source, so that each dataset source has a fixed number of variants and none of the sources are over represented. If the Friedman test is significant, then we perform a Nemenyi test (Hollander et al., 2013), so that we can rank the coordinate systems according to performance. For convenience we use the shortened names 1/2 DOBIN for using the first half of DOBIN components, First 1/2 PCA for using the first half of Principal Components, Last 1/2 PCA for using the second half of Principal Components, 1/2 COV4 for using the first half of COV4 components and 'All Vars' for using all variables in the remainder of the section.

5.1 Significance tests

Figure 12 gives the results of the five coordinate systems on outlier detection method LOF. A Friedman test gave the *p*-value 2.18×10^{-23}, and Figure 12(a) shows the associated Nemenyi plot for this data. This plot shows the ranks of the coordinate systems, with lower ranks indicating better performance. The blue squares highlight methods which are not significantly different from each other. From Figure 12(a) we see that the best coordinates for LOF are 1/2 DOBIN, followed by First 1/2 PCA and All Vars, with First 1/2 PCA and All Vars not significantly different from each other. The coordinates Last 1/2 PCA and 1/2 COV4 give weaker results compared to others.

Figure 12(b) shows the boxplots of performance differences between DOBIN and the other four coordinate systems. Positive values in AUC difference indicate an improvement in DOBIN over the other coordinates. We see that 1/2 DOBIN – all coordinate systems have positive median AUC differences.

Figure 13 shows the results of the five coordinate systems on the outlier detection method KNN. We obtained the *p*-value 1.47×10^{-113} from the Friedman test. Again, from Figure 13(a) we see that 1/2 DOBIN outperforms First 1/2 PCA, All Vars, 1/2 COV4 and Last 1/2 PCA. First 1/2 PCA and All

Vars are not significantly different, similar to 1/2 COV4 and Last 1/2 PCA. This is corroborated by the boxplots in Figure 13(b).

Similar performances are obtained for iForest, which is illustrated in Figure 14. A Friedman test conducted on the data gave the *p*-value 1.79×10^{-108}. The Nemenyi plot shows that while 1/2 DOBIN performs the best, the top three methods are significantly different from each other, with All Vars outperforming First 1/2 PCA. This is supported from the boxplots in Figure 14(b).

We see that 1/2 DOBIN performs best for all three outlier detection methods and is significantly better than First 1/2 PCA, Last 1/2, 1/2 COV4 and All Vars. This is evidence that DOBIN is better as a dimension reduction tool for outlier detection compared to PCA, COV4 and the original variables.

6 Conclusion

Dimension reduction for outlier detection is a topic that has not received much attention in the literature. One way to achieve this is to construct a basis that empowers outlier detection. In this paper, we present DOBIN: a Distance based Outlier BasIs using Neighbors, which strives to accomplish this goal. We used DOBIN on an extensive data repository of real world datasets and showed that DOBIN basis improves performance of the outlier detection methods LOF, KNN and iForest. Using half of the DOBIN components we outperformed PCA, COV4 and the standard basis as coordinate systems for these three outlier detection methods.

In addition, we conducted a visual analysis of outliers using DOBIN on five datasets from diverse sources, which included a character graph from the novel *Les Misérables* and 22 classics from Gutenberg website. We projected the data onto a 2D space spanned by the first two DOBIN vectors, and saw that outliers identified by many methods in O3 plots were indeed far away from the rest of the data. We gained insights about the outliers in these datasets, and identified the variables that contributed to make them outliers.

Currently DOBIN is sensitive to rotations in the input space. As future work we plan to investigate avenues that can contribute to a rotation invariant version of DOBIN. One possibility, for example, would be to first perform a PCA decomposition followed by DOBIN. We would gain rotation-invariance from such a version, but we would potentially lose some interpretability of the results. The R package *dobin* contains the implementation and is available at https://CRAN.R-project.org/package=dobin for download.

Supplementary Material

R package dobin: This package contains the DOBIN basis construction.

Datasets: Datasets discussed in Section 5 are available at Kandanaarachchi, Muñoz Acosta, Smith-Miles and Hyndman (2019). The character network from the novel *Les Misérables* was taken from the R package *SOMbrero* (Vialaneix et al., 2019). The book vectors dataset used in Section 4.5 is available at the github repository https://github.com/sevvandi/Outlier-Basis.

Scripts: The script `Figures_For_Paper_1.R` contains the R code used to conduct experiments and produce graphs in Section 3. The script `Figures_For_Paper_2.R` contains the R code used in Section 4. And finally, the script `Figures_For_Paper_3.R` contains the R code used to produce the graphs in Section 5. The original computation on the data repository detailed in Section 5 was conducted using the MonARCH HPC Cluster. The results of this computation are available at the github repository https://github.com/sevvandi/Outlier-Basis.

Other R-packages: In addition to the R package *dobin* we have also used the R packages *OutliersO3* (Unwin, 2019 *b*), *HDoutliers* (Fraley, 2018), *isolationForest* (Liu, 2009), *DMwR* (Torgo, 2010), *pROC* (Robin et al., 2011), *ggplot2* (Wickham, 2016), *mbgraphic* (Grimm, 2019), *gridExtra* (Auguie, 2017), *tsutils* (Kourentzes, 2019), *dplyr* (Wickham et al., 2019) and *tidyr* (Wickham and Henry, 2020).

Acknowledgments

Funding was provided by the Australian Research Council through the Linkage Project LP160101885. This research was also supported by the Monash eResearch Centre and eSolutions-Research Support Services through the MonARCH HPC Cluster.

References

Aggarwal, C. C. and Yu, P. S. (2001), ‘Outlier detection for high dimensional data’, *ACM Sigmod Record* **30**(2), 37–46.

Amnarttrakul, R., Thongteeraparp, A. et al. (2011), ‘New statistics for detection of outliers using the last few principal components’, *Science Asia* **37**, 355–359.

Anton, H. and Rorres, C. (2013), *Elementary linear algebra: applications version*, John Wiley & Sons.

Auguie, B. (2017), *gridExtra: Miscellaneous Functions for ”Grid” Graphics*. R package version 2.3.

URL: *https://CRAN.R-project.org/package=gridExtra*

Billor, N., Hadi, A. S. and Velleman, P. F. (2000), ‘Bacon: blocked adaptive computationally efficient outlier nominators’, *Computational statistics & data analysis* **34**(3), 279–298.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. and Sander, J. (2000), ‘LOF: identifying density-based local outliers’, *ACM Sigmod Record* **29**(2), 93–104.

Brys, G., Hubert, M. and Rousseeuw, P. (2005), ‘A robustification of independent component analysis’, *Journal of Chemometrics: A Journal of the Chemometrics Society* **19**(5-7), 364–375.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J., Micenková, B., Schubert, E., Assent, I. and Houle, M. E. (2016), ‘On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study’, *Data Mining and Knowledge Discovery* **30**(4), 891–927.

Chambers, J., Cleveland, W., Kleiner, B. and Tukey, P. (1983), ‘Graphical methods for data analysis’, *Pacific Grove: Wadsworth* pp. 158–62.

Csardi, G. and Nepusz, T. (2006), ‘The igraph software package for complex network research’, *InterJournal* **Complex Systems**, 1695.

URL: *http://igraph.org*

Fraley, C. (2018), *HDoutliers: Leland Wilkinson’s Algorithm for Detecting Multidimensional Outliers*. R package version 1.0.

URL: *https://CRAN.R-project.org/package=HDoutliers*

Goldstein, M. and Uchida, S. (2016), ‘A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data’, *PLOS ONE* **11**(4), 1–31.

Grimm, K. (2019), *mbgraphic: Measure Based Graphic Selection*. R package version 1.0.1.

URL: *https://CRAN.R-project.org/package=mbgraphic*

Hollander, M., Wolfe, D. A. and Chicken, E. (2013), *Nonparametric statistical methods*, Vol. 751, John Wiley & Sons.

Hyndman, R. J., Wang, E. and Laptev, N. (2015), Large-scale unusual time series detection, *in* ‘2015 IEEE international conference on data mining workshop (ICDMW)’, IEEE, pp. 1616–1619.

Kaggle and Machine-Learning-Group (2018), ‘Credit card fraud detection’.

URL: *https://www.kaggle.com/mlg-ulb/creditcardfraud*

Kandanaarachchi, S. (2019), *dobin: Dimension Reduction for Outlier Detection*. R package version 1.0.2.

URL: *https://CRAN.R-project.org/package=dobin*

Kandanaarachchi, S., Muñoz Acosta, M., Smith-Miles, K. and Hyndman, R. (2019), ‘Datasets for outlier detection’.

URL: *https://monash.figshare.com/articles/Datasets_12338_zip/7705127/4*

Kandanaarachchi, S., Muñoz, M. A., Hyndman, R. J. and Smith-Miles, K. (2019), ‘On normalization and algorithm selection for unsupervised outlier detection’, *Data Mining and Knowledge Discovery* pp. 1–46.

Keller, F., Muller, E. and Bohm, K. (2012), Hics: High contrast subspaces for density-based outlier ranking, *in* ‘2012 IEEE 28th international conference on data engineering’, IEEE, pp. 1037–1048.

Kourentzes, N. (2019), *tsutils: Time Series Exploration, Modelling and Forecasting*. R package version 0.9.0.

URL: *https://CRAN.R-project.org/package=tsutils*

Kriegel, H.-P., Kröger, P., Schubert, E. and Zimek, A. (2009), Loop: local outlier probabilities, *in* ‘Proceedings of the 18th ACM conference on Information and knowledge management’, ACM, pp. 1649–1652.

Liu, F. T. (2009), *IsolationForest: Isolation Forest*. R package version 0.0-26/r4.

URL: *https://R-Forge.R-project.org/projects/iforest/*

Liu, F. T., Ting, K. M. and Zhou, Z.-H. (2008), Isolation forest, *in* ‘2008 Eighth IEEE International Conference on Data Mining’, IEEE, pp. 413–422.

Project Gutenberg (n.d.), http://www.gutenberg.org. Accessed: 2019-08-30.

Ramaswamy, S., Rastogi, R. and Shim, K. (2000), ‘Efficient algorithms for mining outliers from large data sets’, *ACM Sigmod Record* **29**(2), 427–438.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C. and Müller, M. (2011), 'proc: an open-source package for r and s+ to analyze and compare roc curves', *BMC Bioinformatics* **12**, 77.

Rousseeuw, P. J. and Bossche, W. V. D. (2018), 'Detecting deviating data cells', *Technometrics* **60**(2), 135–145.

Rousseeuw, P. J. and Driessen, K. V. (1999), 'A fast algorithm for the minimum covariance determinant estimator', *Technometrics* **41**(3), 212–223.

Rousseeuw, P. J. and Leroy, A. M. (2005), *Robust regression and outlier detection*, John wiley & sons.

Silge, J. and Robinson, D. (2016), 'tidytext: Text mining and analysis using tidy data principles in r', *JOSS* **1**(3).

URL: *http://dx.doi.org/10.21105/joss.00037*

Talagala, P. D., Hyndman, R. J., Smith-Miles, K., Kandanaarachchi, S. and Muñoz, M. A. (2019), 'Anomaly detection in streaming nonstationary temporal data', *Journal of Computational and Graphical Statistics* pp. 1–28. just-accepted.

Torgo, L. (2010), *Data Mining with R, learning with case studies*, Chapman and Hall/CRC.

URL: *http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR*

Tyler, D. E., Critchley, F., Dümbgen, L. and Oja, H. (2009), 'Invariant co-ordinate selection', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(3), 549–592.

Unwin, A. (2019 *a*), 'Multivariate outliers and the O3 plot', *Journal of Computational and Graphical Statistics* pp. 1–11.

Unwin, A. (2019 *b*), *OutliersO3: Draws Overview of Outliers (O3) Plots*. R package version 0.6.2.

URL: *https://CRAN.R-project.org/package=OutliersO3*

Vakili, K. and Schmitt, E. (2014), 'Finding multivariate outliers with fastpcs', *Computational Statistics & Data Analysis* **69**, 54–66.

Vialaneix, N., Mariette, J., Olteanu, M., Rossi, F., Bendhaiba, L. and Bolaert, J. (2019), *SOMbrero: SOM Bound to Realize Euclidean and Relational Outputs*. R package version 1.2-4.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.

URL: *http://ggplot2.org*

Wickham, H., François, R., Henry, L. and Müller, K. (2019), *dplyr: A Grammar of Data Manipulation*. R package version 0.8.3.

URL: *https://CRAN.R-project.org/package=dplyr*

Wickham, H. and Henry, L. (2020), *tidyr: Tidy Messy Data*. R package version 1.0.2.

URL: *https://CRAN.R-project.org/package=tidyr*

Wilkinson, L. (2017), 'Visualizing big data outliers through distributed aggregation', *IEEE transactions on visualization and computer graphics* **24**(1), 256–266.

Zimek, A., Schubert, E. and Kriegel, H.-P. (2012), 'A survey on unsupervised outlier detection in high-dimensional numerical data', *Statistical Analysis and Data Mining: The ASA Data Science Journal* **5**(5), 363–387.

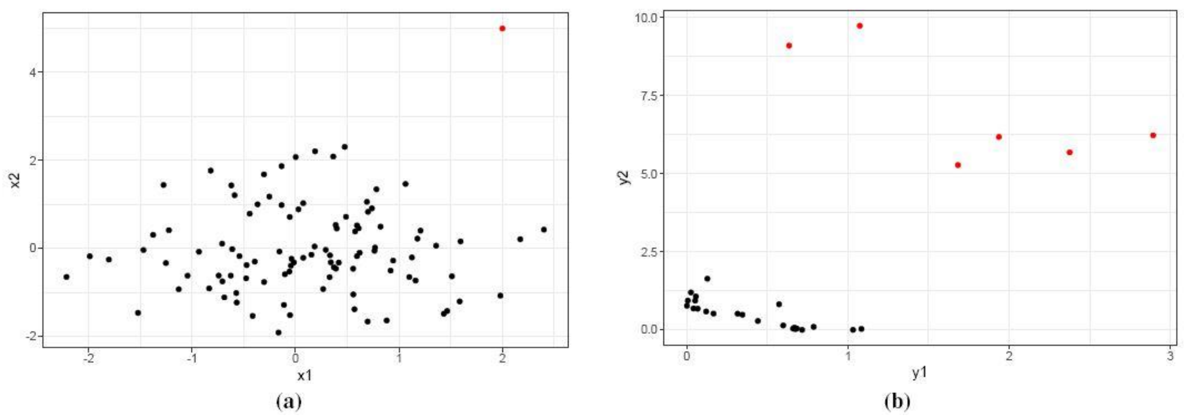


Fig. 1 The X and Y spaces. The X space in (a) and the associated Y space in (b) with $q = 0.95$. The red colored point in the X space gives rise to 6 points, which are colored in red in the Y space.

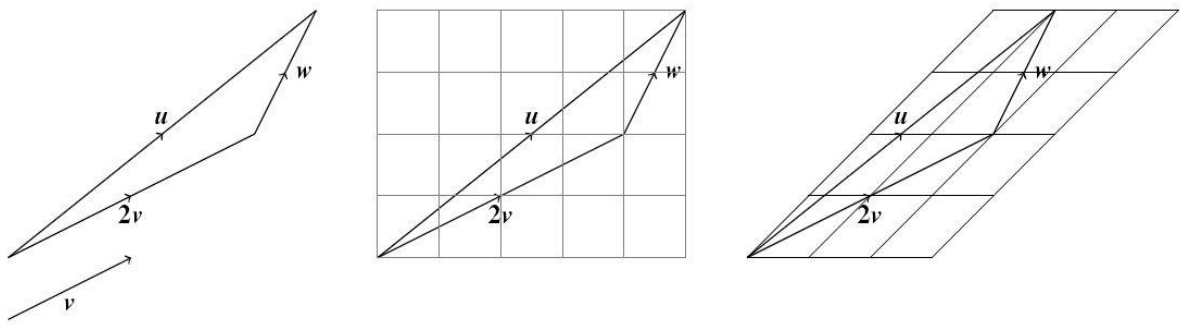


Fig. 2 Here $\boldsymbol{u} = 2\boldsymbol{v} + \boldsymbol{w}$. In the coordinate system in the middle $\boldsymbol{v} = (2,1)^T$, $\boldsymbol{w} = c(1,2)^T$ and $\boldsymbol{u} = (5,4)^T$. In the coordinate system on the right $\boldsymbol{v} = (1,1)^T$, $\boldsymbol{w} = c(-1,2)^T$ and $\boldsymbol{u} = (1,4)^T$. Regardless of the coordinate system the vector equation $\boldsymbol{u} = 2\boldsymbol{v} + \boldsymbol{w}$ holds true.

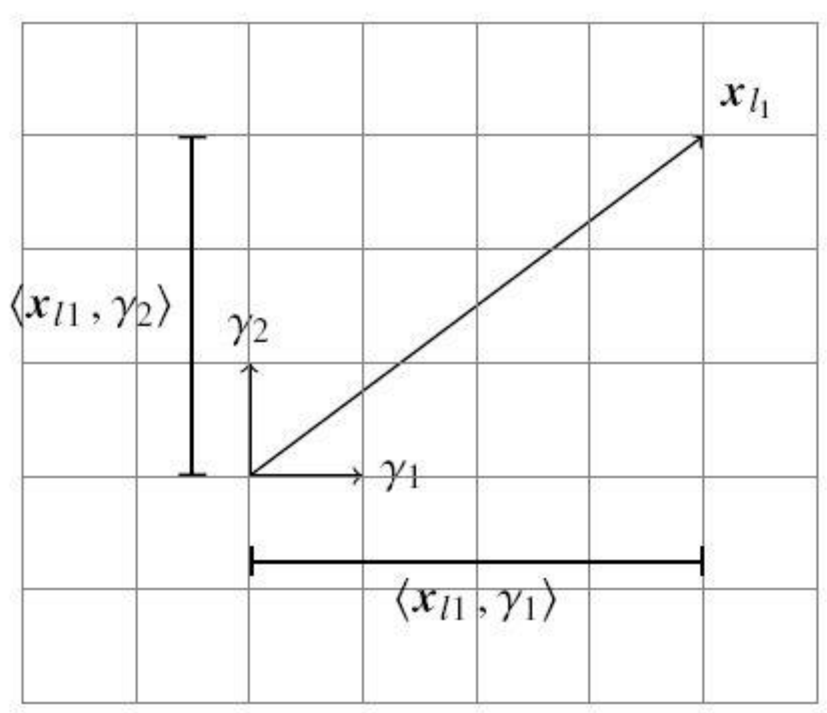

$\boldsymbol{x}_{l_1}$
$\langle \boldsymbol{x}_{l1}, \gamma_2 \rangle$
γ_2
γ_1
$\langle \boldsymbol{x}_{l1}, \gamma_1 \rangle$

Fig. 3 The coordinates of $\boldsymbol{x}_{l_1}$ in the orthonormal basis $\{\gamma_1, \gamma_2\}$ are $\big(\langle \boldsymbol{x}_{l1}, \gamma_1 \rangle, \langle \boldsymbol{x}_{l1}, \gamma_2 \rangle\big)$, i.e. $\boldsymbol{x}_{l1} = \langle \boldsymbol{x}_{l1}, \gamma_1 \rangle \gamma_1 + \langle \boldsymbol{x}_{l1}, \gamma_2 \rangle \gamma_2$.

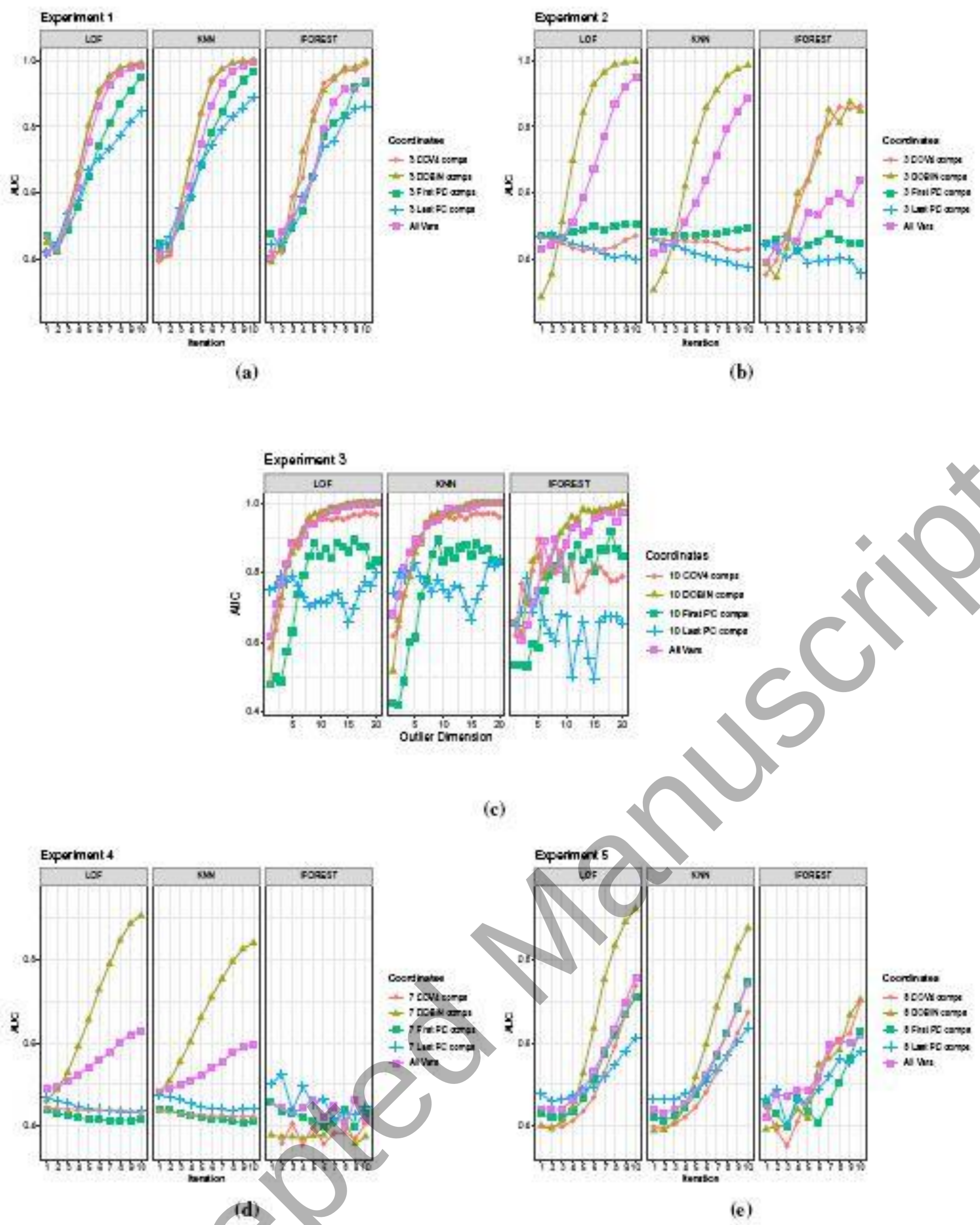


Fig. 4 Results of experiments 1 - 5 for LOF, KNN and iForest.

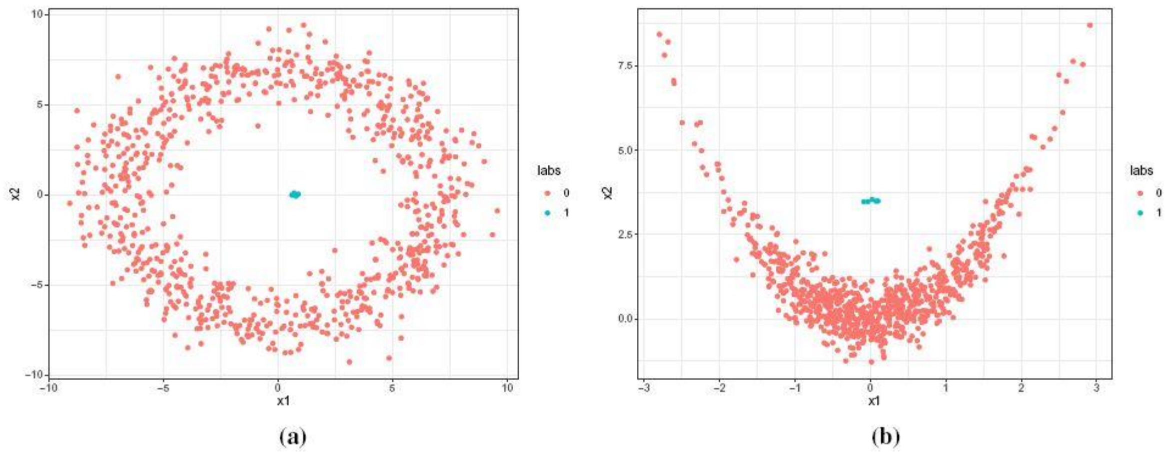
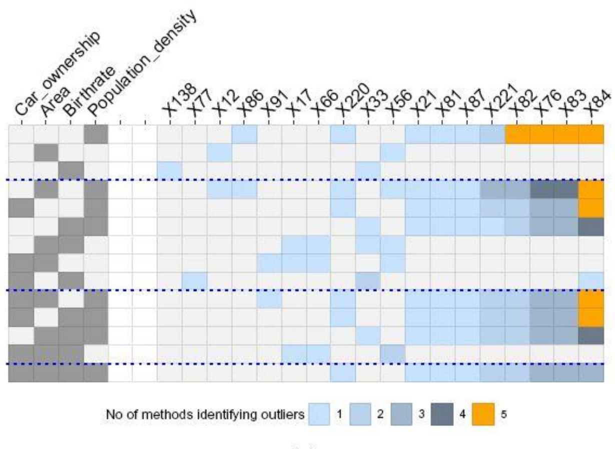


Fig. 5 Figures 5(a) and 5(b) show data for experiments 4 and 5 in the first two dimensions with inliers in orange and outliers in blue. The other dimensions are normally distributed with mean 0 and standard deviation 1.



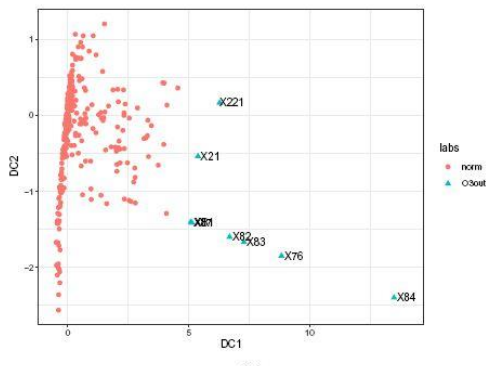
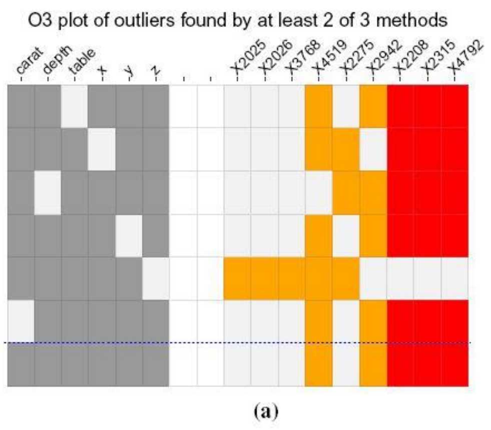


Fig. 6 O3plot of the *Election2005* dataset in Figure 6(a) and the first 2 DOBIN components in Figure 6(b)



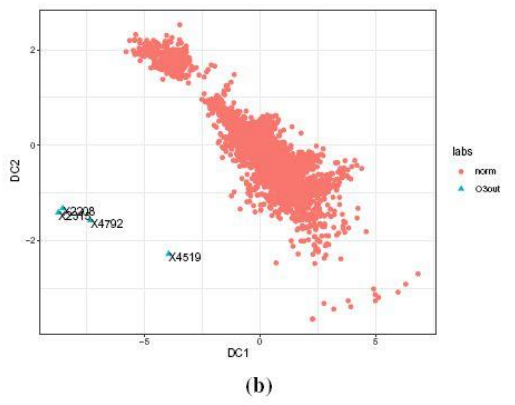
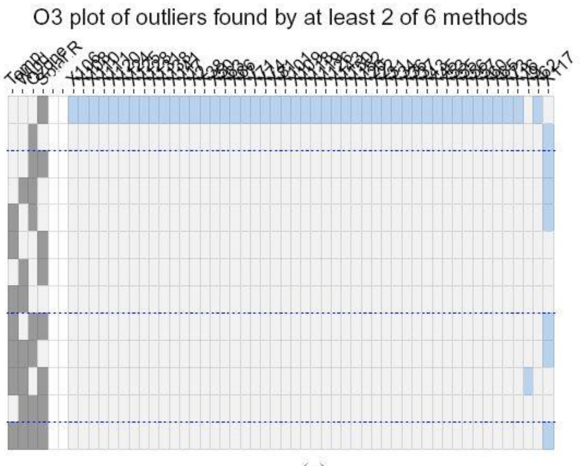
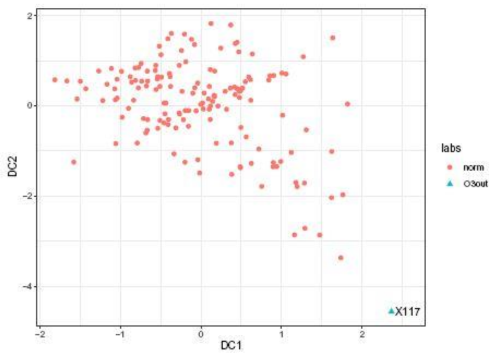


Fig. 7 O3plot of the *Diamonds* dataset in Figure 6(a) and the first 2 DOBIN components in Figure 6(b).



(a)



(b)

Fig. 8 O3plot of the *Airquality* dataset in Figure 8(a) and the first 2 DOBIN components in Figure 8(b).

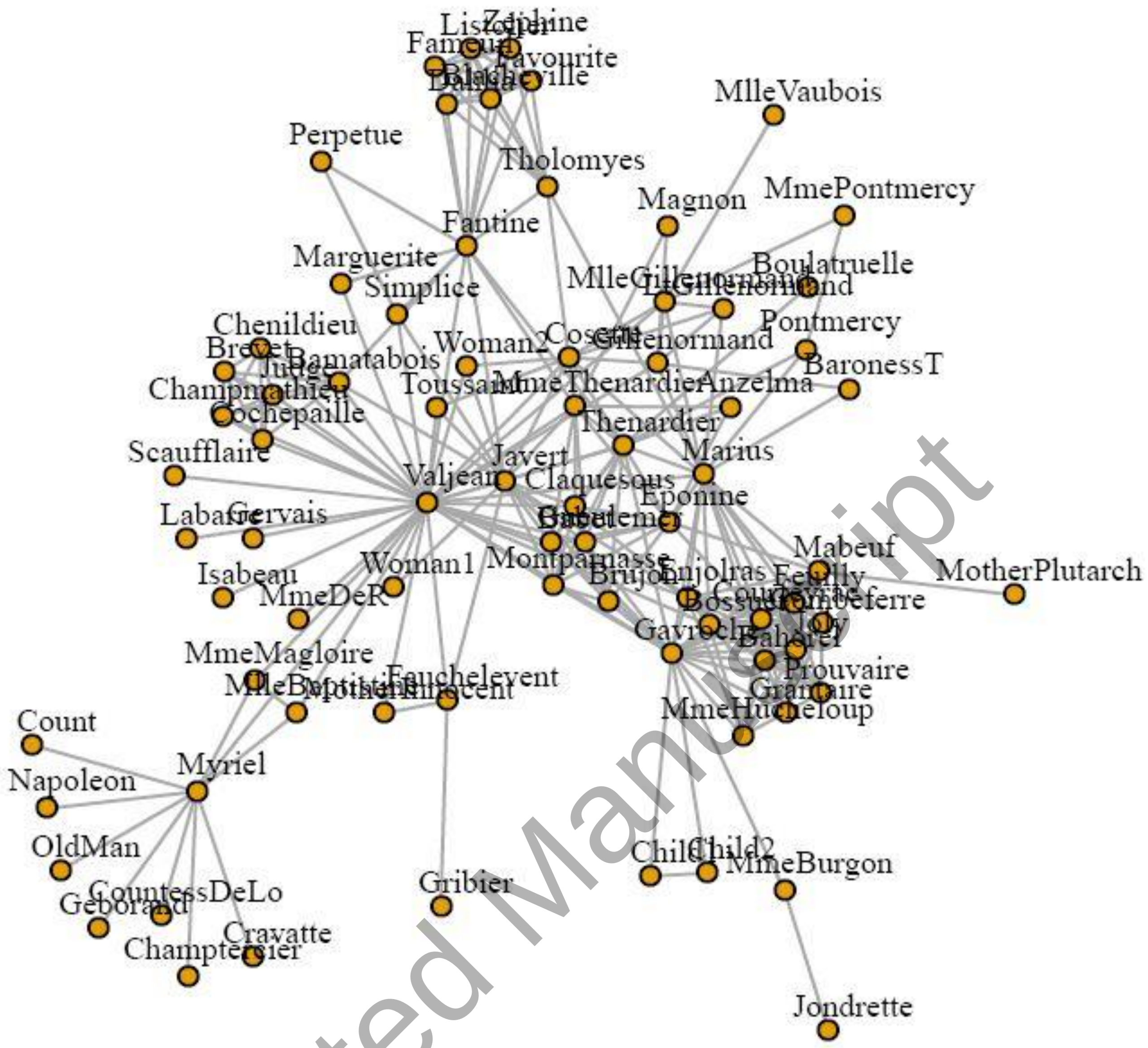
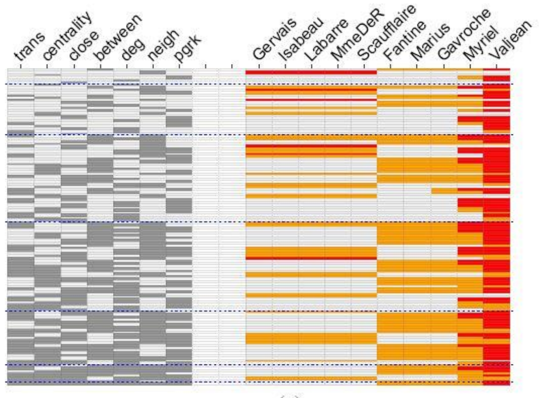
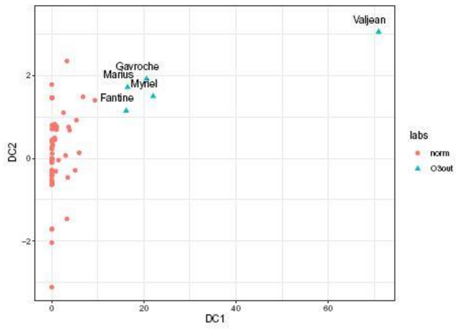


Fig. 9 The *lesmis* dataset.



(a)



(b)

Fig. 10 O3plot of the transformed *lesmis* dataset in Figure 10(a) and the first 2 DOBIN components in Figure 10(b).

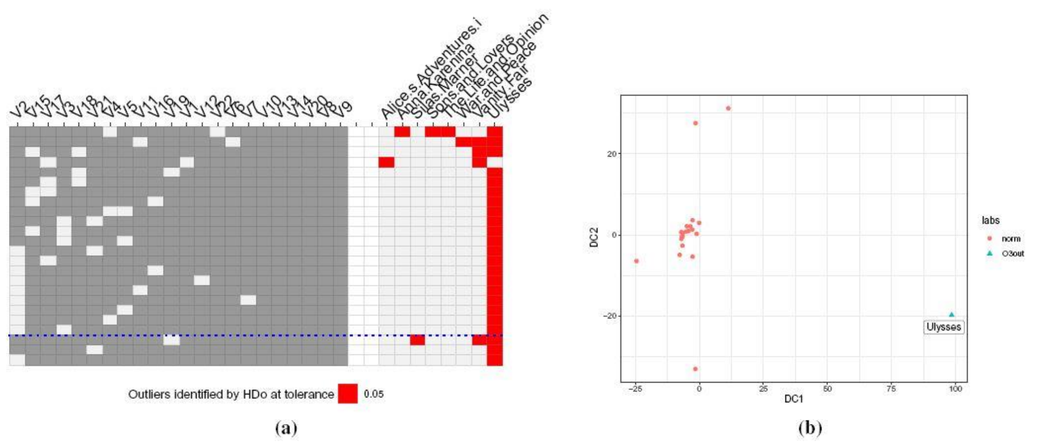


Fig. 11 O3plot of 22 Gutenberg classics in Figure 11(a) and the first 2 DOBIN components in Figure 11(b).

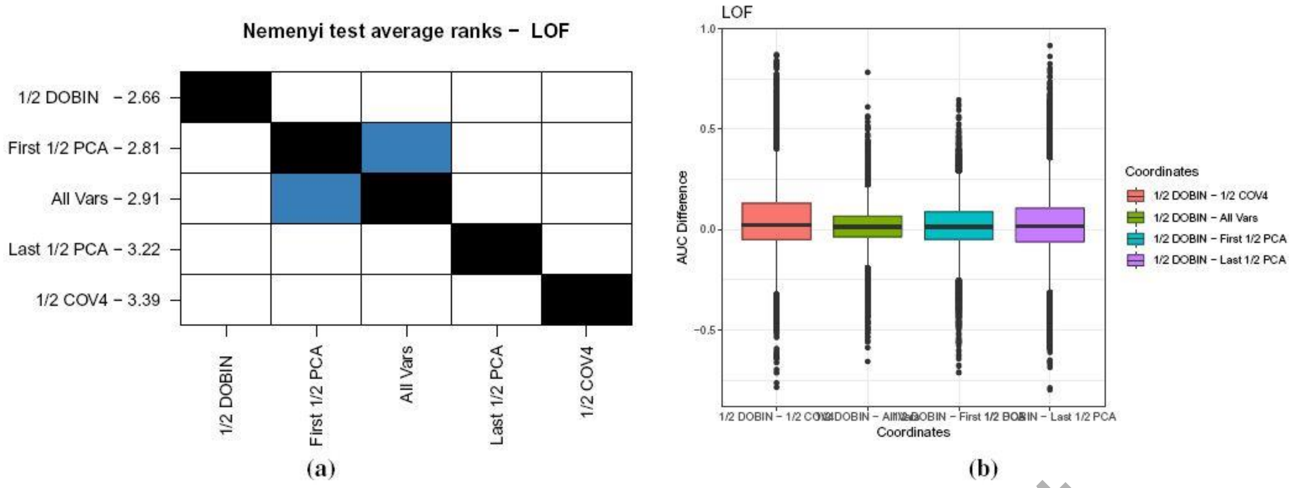


Fig. 12 Nemenyi plot of LOF results and the boxplots of performance differences in Figures 12(a) and 12(b).

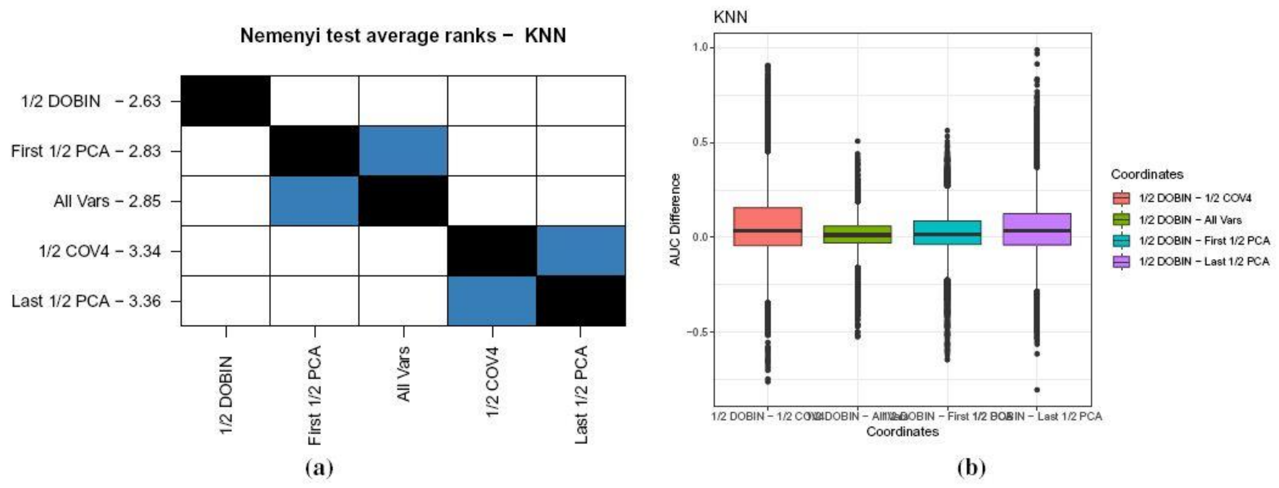


Fig. 13 Nemenyi plot of KNN results and the boxplots of performance differences in Figures 13(a) and 13(b).

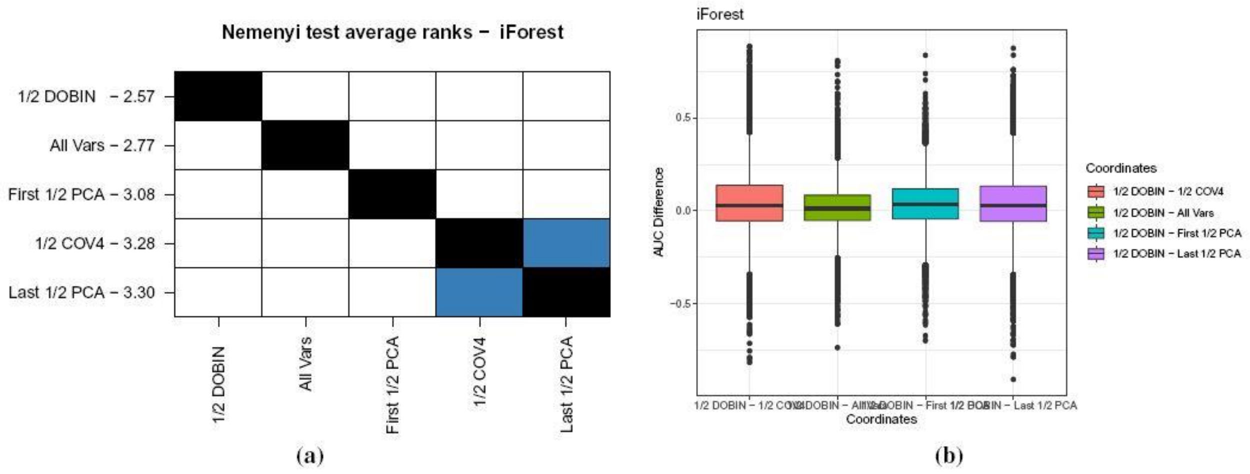


Fig. 14 Nemenyi plot of iForest results and the boxplots of performance differences in Figures 14(a) and 14(b).

Table 1 Removing pairs of points for the *Y* space construction

Point index	knn distance squared for				
	k = 1	*k* = 2	*k* = 3	*k* = 4	*k* = 5
1	20	25	28	32	37
2	~~5~~	~~7~~	~~9~~	~~11~~	20
3	~~2~~	~~3~~	~~4~~	~~6~~	~~8~~
4	~~2~~	~~7~~	~~10~~	~~13~~	~~23~~
5	~~4~~	~~5~~	~~7~~	~~9~~	~~11~~
6	~~3~~	~~4~~	~~8~~	~~10~~	~~13~~