Estimating Tukey Depth Using Incremental Quantile Estimators

Abstract

Measures of distance or how data points are positioned relative to each other are fundamental in pattern recognition. The concept of depth measures how deep an arbitrary point is positioned in a dataset, and is an interesting concept in this regard. However, while this concept has received a lot of attention in the statistical literature, its application within pattern recognition is still limited.

To increase the applicability of the depth concept in pattern recognition, we address the well-known computational challenges associated with the depth concept, by suggesting to estimate depth using *incremental quantile estimators*. The suggested algorithm can not only estimate depth when the dataset is known in advance, but can also *track* depth for dynamically varying data streams by using *recursive updates*. The tracking ability of the algorithm was demonstrated based on a real-life application associated with detecting changes in human activity from real-time accelerometer observations. Given the flexibility of the suggested approach, it can detect virtually *any* kind of changes in the distributional patterns of the observations, and thus outperforms detection approaches based on the Mahalanobis distance.

Keywords: data stream, incremental quantile estimator, distributional patterns, real-time analytics, Tukey depth

1. Introduction

Measures of distance or how data points are positioned relative to each other, are fundamental in pattern recognition. For example in anomaly detection (Erfani et al., 2016) e.g using auto encoders (Zavrtanik et al., 2021; Chang et al., 2021), clustering (Huang et al., 2021; Ma et al., 2021) or classification (Rastin et al., 2021; Iwana and Uchida, 2020).

To measure distance or how data points are positioned relative to each other, *data depth* is an interesting concept. Data depth measures how deep an arbitrary point is position in a dataset. While the concept has received a lot of attention in the statistical literature (Mosler, 2013), the application within pattern recognition is still limited. There are however some notable exceptions. For example Kim et al. (2018); Hubert et al. (2017); Jörnsten (2004) applied the concept for classification and clustering. Depth has also been applied to a wide range of disciplines such as economy (Kim et al., 2018; Kosiorowski and Zawadzki, 2014; Hubert et al., 2017), health and biology (Williams et al., 2008; Hubert et al., 2015), ecology (Cerdeira et al., 2018) and hydrology (Chebana and Ouarda, 2011) to name a few.

The earliest and most popular depth measure is Tukey depth (Tukey, 1975). The Tukey depth of a point is defined as the minimum probability mass carried by any closed halfspace containing the point. However, the computation of Tukey depth for higher dimensions or for even moderate amounts of data is computationally demanding which limits its applicability (Liu et al., 2019).

The main aim of this paper is to address these aforementioned

computational issues, and thus increase the applicability of the depth concept within pattern recognition. Our approach takes advantage of the following result from Kong and Mizera (2012) according to which the authors defined halfspaces such that a specific portion of the data points are on one side of the halfspace. They further showed that contours with a specific Tukey depth can be estimated from the intersection of such halfspaces over different directions. Such contours can again be used to estimate the depth of any point. In order to apply this result to estimate depth in dimension p, the positions of $O(c^{p-1})$, $c > 1$ halfspaces must be estimated requiring estimators that are both memory and computationally efficient. In this paper, we therefore suggest to use *incremental quantile estimators* to estimate the positions of the halfspaces (Hammer et al., 2019, 2021). These estimators only need to store a single value in memory, i.e. $O(1)$, and only need to perform a single operation per observation resulting in a computational complexity of $O(n)$ for n observations. As opposed to this, traditional quantile estimators have a memory requirement of $O(n)$ and a $O(n \log n)$ computational complexity. However, the computational efficiency comes with a price and traditional estimators provide more precise estimates based on the same observations.

The second aim is to recursively update and even track Tukey depth contours of streams of multivariate data in *real time*. A remarkable advantage with incremental quantile estimators is that they not only can estimate quantiles when the data is known in advance, but can recursively update and even track quantiles of data streams. Thus, by using incremental quantile estimators to estimate halfspaces, Tukey depth contours

can be tracked in real time. We are not aware of any other method that can efficiently compute Tukey depth in real-time.

Finally, the real-world applicability of computing Tukey depth in real-time settings is demonstrated where the developed methods are used to detect changes in human activity in real-time from accelerometer observations. Due the flexibility of the suggested approach, it can detect virtually *any* kind of changes in the distributional patterns of the accelerometer observation, and outperforms popular approaches based on Mahalanobis distance.

The main contributions of the paper are as follows:

- We present a new, simple and computationally efficient method to compute Tukey depth.
- The method can even be used to compute Tukey depth in real- time, and is to the best of our knowledge the first method with this ability.
- The method is applied to detect changes in human activity in real-time which demonstrates its usefulness and applicability in real-world scenarios.

The paper is organized as follows. In Section 2, the concept of depth is introduced including some theoretical fundamentals to compute Tukey depth. Section 3 provides an efficient procedure to estimate Tukey depth. Section 4 presents performance metrics that will be used to evaluate the algorithm and Sections 5 and 6 provide synthetic and real-life data experiments.

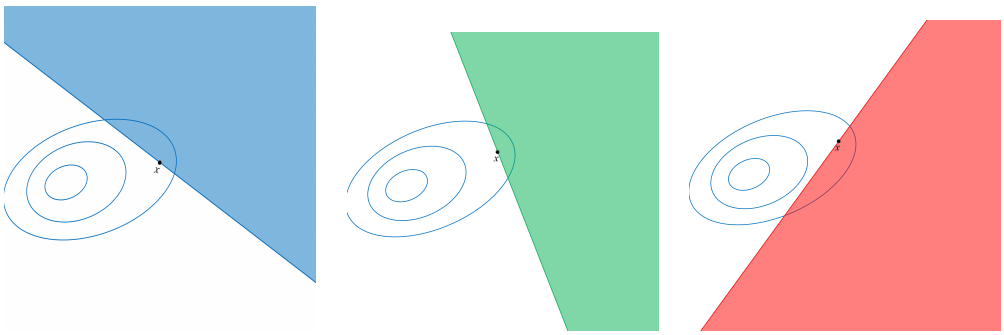

Figure 1: Examples of three halfspaces (blue, green, red) containing the point x. The blue contours represent some probability distribution P.

2. The Concept of Depth

Let $X = (X_1, \ldots, X_p)^T$ represent a p-dimensional stochastic vector with probability distribution P. Let $D(x, P)$ denote the *depth function* of a point x with respect to the probability distribution P. A high (low) value of the depth function refers to a central (outlying) point of the probability distribution. A general depth function is defined by satisfying the natural requirements of affine invariance, maximality at center, monotonicity relative to deepest point and vanishing at infinity (Zuo and Serfling, 2000).

The most used depth function is Tukey depth.

Definition 1 (Tukey depth)**.** *Let $\mathcal{U}$ refer to the set of all vectors with unit length. Tukey depth is the minimum probability mass carried by any closed halfspace containing the point*

$$D(x, P) = \inf_{u \in \mathcal{U}} P\left(u^T X \leq u^T x\right) \tag{1}$$

Figure 1 shows three halfspaces containing the point x. The probability mass carried by the red and blue halfspaces are larger than for the green

halfspace. Thus the probability mass carried by the green halfspace is closer to the Tukey depth, which is the minimum probability mass over all half spaces containing x. Intuitively, this is a reasonable and general measure for the centrality of x with respect to P.

Define α-depth region, directional quantile and directional quantile halfspace.

Definition 2 (α-depth region)**.** *The α-depth region with respect to Tukey depth, $D(\alpha)$, is defined as the set of points whose depth is at least α*

$$D(\alpha) = \{x \in \mathbb{R}^p : D(x, P) \geq \alpha\} \tag{2}$$

The boundary of $D(\alpha)$ is known as the α-depth contour.

The α-depth regions are closed, convex, and nested for increasing α.

Definition 3 (Directional quantile)**.** *For any unit directional vector $u \in \mathcal{U}$, define the directional quantile as*

$$Q(\alpha, u^T X) = F^{-1}_{u^T X}(\alpha) \tag{3}$$

where $F^{-1}_{u^T X}(x)$ refers to the inverse of the univariate cumulative distribution function of the projection of X on u.

Definition 4 (Directional quantile halfspace)**.** *The directional quantile halfspace is defined as*

$$H(\alpha, u) = \left\{x \in \mathbb{R}^p : u^T x \geq Q(\alpha, u^T X)\right\} \tag{4}$$

which is bounded away from the origin at distance $Q(\alpha, u^T X)$ by the hyperplane with normal vector u.

Consequently $P(X \in H(\alpha, u)) = 1 - \alpha$ for any $u \in \mathcal{U}$.

The estimation procedures in this paper builds on the following theorem from Kong and Mizera (2012).

Theorem 1. *The α-depth region in* (2) *equals the directional quantile envelope*

$$D(\alpha) = \bigcap_{u \in \mathcal{U}} H(\alpha, u) \tag{5}$$

Tukey depth may not be defined for depths above some threshold and the intersection becomes empty.

3. Efficient Estimation of Tukey Depth

Given a multivariate dataset, in this section we suggest a simple procedure to estimate whether an arbitrary point is within or outside an α-depth region. The procedure uses Theorem 1, and consists of three parts.

1. Unit length directional vectors. The generation of uniformly distributed directional vectors is simple: Let $Z_1, \ldots, Z_p$ be independent standard normally distributed stochastic variables and define $Z = (Z_1, \ldots, Z_p)^T$. Then $U = Z/\|Z\|_2$ will be uniformly distributed on the unit sphere, where $\|\cdot\|_2$ refers to the Euclidean norm. This is the procedure used in most of the experiments in this paper. However, intuitively, using directional vectors that are more equidistantly spread on the unit sphere would be more efficient. We thus also considered the following approach according to which we generate many uniformly distributed directional vectors, N_u, and secondly filter out directional vectors that are closer than some threshold. The approach is however computationally demanding, $O(N_u^2 p^2)$. There are other algorithms

to generate fairly equidistantly spread direction vectors, see e.g spiral algorithms (Saff and Kuijlaars, 1997). We have not evaluated the potential of these algorithms.

2. Directional quantiles estimates. The next part is to estimate directional quantiles for each directional vector generated above. As pointed out in the introduction, we use incremental quantile estimators. A prominent example is the DUMIQE algorithm which recursively updates directional quantile estimates as follows for every observation $u_i^T x_{j-1}$ (Yazidi and Hammer, 2017)

$$\begin{aligned} \widehat{Q}(\alpha, u_i^T X_j) &\leftarrow (1+\lambda\alpha)\widehat{Q}(\alpha, u_i^T X_{j-1}), && \text{if } u_i^T x_j > \widehat{Q}(\alpha, u_i^T X_{j-1}) \\ \widehat{Q}(\alpha, u_i^T X_j) &\leftarrow (1-\lambda(1-\alpha))\widehat{Q}(\alpha, u_i^T X_{j-1}), && \text{if } u_i^T x_j < \widehat{Q}(\alpha, u_i^T X_{j-1}) \end{aligned} \tag{6}$$

The update is quite intuitive. If the sample $u_i^T x_j$ is above (respectively below) the current estimate, increase (respectively reduce) the corresponding directional quantile estimate. The tuning parameter $\lambda > 0$ controls the update size. If the data is known beforehand or it comes in the form of a stationary data stream, it makes sense to let the value of λ be reduced with time. For non-stationary data streams a constant value of λ is more suitable to gradually forget old and outdated data (Yazidi and Hammer, 2017). The procedure is detailed in Algorithm 1.

3. Compute if a point w is within the α-depth region. The directional quantile estimates from Algorithm 1 can be used to compute if w is within all the directional quantile halfspaces and thus, according to Theorem 1, being within the α-depth region. The procedure is detailed in Algorithm 2. The condition in line 2 is based on Equation (4) and checks if w is outside of the estimated directional quantile halfspace.

Algorithm 1 Estimating directional quantiles.

Input:

$u_1, \ldots, u_{n_u}$ // Unit length directional vectors

$x_1, x_2, \ldots, x_n$ // Dataset

α, λ

$\widehat{Q}(\alpha, u_i^T X_0)$ // Initial value

Method:

for $j \in 1, 2, \ldots, n$ **do**
  for $i \in 1, 2, \ldots n_u$ **do**
    if $u_i^T x_j > \widehat{Q}(\alpha, u_i^T X_{j-1})$ **then**
      $\widehat{Q}(\alpha, u_i^T X_j) \leftarrow (1 + \lambda\alpha)\widehat{Q}(\alpha, u_i^T X_{j-1})$
    else
      $\widehat{Q}(\alpha, u_i^T X_j) \leftarrow (1 - \lambda(1 - \alpha))\widehat{Q}(\alpha, u_i^T X_{j-1})$
    end if
  end for
end for

Convergence. The procedure consists of two approximations 1) the finite number of directional vectors and 2) the estimates of the true directional quantiles. To ensure convergence, the directional vector selection procedure must cover the unit sphere when the number of directional vectors goes to infinity and, secondly, the directional quantile estimates must converge to the true directional quantiles, when the number of observations goes to infinity. By using the simple procedure above to select uniformly distributed directional vectors, the first requirement is satisfied. Further, Yazidi and Hammer (2017) and Hammer et al. (2021) prove the second requirement.

Algorithm 2 Compute if a point w is within the α-depth region.

Input:

$\widehat{Q}(\alpha, u_i^T X_n), i = 1, \ldots, n_u$ // Dir. quantile estimates from Algorithm 1.

w, $i = 1$

InAlphaDepthRegion = True // True (False) if w is within (outside) the α-depth region

Method:

```
while InAlphaDepthRegion and i ≤ n_u do
    if u_i^T w < Q̂(α, u_i^T X_n) then
        InAlphaDepthRegion = False
    end if
    i ← i + 1
end while
Print("Point w in α-depth region?", InAlphaDepthRegion)
```

4. Performance Metrics

We suggest to measure error along lines $l_i, i = 1, \ldots, n_v$ going through the center of the true distribution and outward in uniformly distributed directions $v_i, i = 1, \ldots, n_v$ (Figure 2). This approach scales well with dimension p. We suggest two error measures:

Depth error: Let $\widetilde{w}_{i,k}$ denote the point of intercept between the line, l_i, and the envelope and compute the true depth at this point, $D(\widetilde{w}_{i,k}, P)$. The error is computed using mean absolute depth error (MADE) over all the

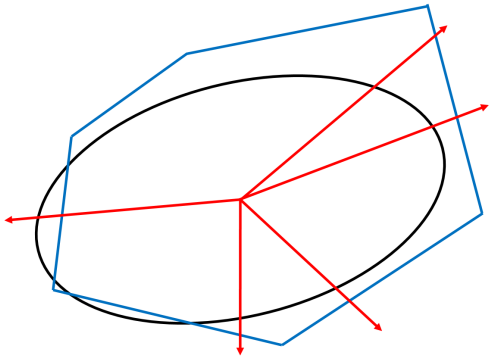

Figure 2: Figure illustrating the approach to measure α-depth contour estimation error. The black and blue curves show the true α-depth regions and the envelope estimate. The lines with directions $v_i, i = 1, \ldots, n_v$ are shown in red.

lines $l_i, i = 1, \ldots, n_v$

$$\text{MADE}_k = \frac{1}{n_v} \sum_{i=1}^{n_v} |\alpha_k - D(\widetilde{w}_{i,k}, P)|$$

and again average over envelopes

$$\text{MADE} = \frac{1}{K} \sum_{k=1}^{K} \text{MADE}_k \tag{7}$$

To compute MADE for higher dimensions, the true depth must be computed for a large set of points $\widetilde{w}_{i,k}$. For non-elliptic distributions this is computationally demanding and was limited to $p \leq 6$ in the experiments. For elliptic distributions, and in particular multivariate normal distributions, the true depth of any point can be computed analytically and thus MADE was computed up to dimension $p = 10$ in the experiments. Details are given in supplementary material S.1. Obviously, if we knew that the observations were multivariate normally distributed, other depth measures such as Maha-

lanobis depth would be more natural, but the computations are only used to evaluate the performance of the algorithm for high dimensions.

Euclidean distance: Along each line, l_i, compute the point of intercept between the line and the true α-depth contour of depth α_k, denoted $w_{i,k}$. Compute the error as the average Euclidean distance (ED)

$$\mathrm{ED}_k = \frac{1}{n_v} \sum_{i=1}^{n_v} \|w_{i,k} - \widetilde{w}_{i,k}\|_2$$

where $\widetilde{w}_{i,k}$ still refers to the intercept between line l_i and the envelope. Further take the average over envelopes

$$\mathrm{ED} = \frac{1}{K} \sum_{k=1}^{K} \mathrm{ED}_k \tag{8}$$

5. Synthetic Experiments

In this section, the performance of the algorithms in Section 3 are evaluated in several synthetic experiments. The experiments focus on streaming data, except in Section 5.2. In Section 6, the algorithms are demonstrated in a real-life data example.

All computations were run on a Dell PowerEdge R815 server with 64 1.8 GHz AMD CPU processors and Linux Ubuntu operating system version 16.04. The experiments were implemented in R (R Core Team, 2021), but with the most computer intensive parts in C++ integrated using Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013).

5.1. Synthetic Experiments - Static Data Stream

Figures 3 show results of estimating the $\alpha = 0.1$ depth contour for a multivariate normally distributed data stream with parameters

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 1 & 0.82 \\ 0.82 & 1 \end{bmatrix} \tag{9}$$

Directional quantiles were estimated using DUMIQE with decreasing values of the tuning parameter, $\lambda_n = 1/n$. We see that a fairly good estimate is achieved with 200 observations and that the error is minimal with 2000 observations. A similar visualization for the highly non-elliptical and heavy tailed lognormal distribution is shown in Figure 7 in supplementary material S.2. Due to the flexibility of the depth concept, the method performs equally well for such a distribution. Further, in supplementary material S.2, a few examples of joint estimation of multiple α-depth regions using the ShiftQ algorithm are shown. The results show that multiple depth regions can efficiently be estimated for both Gaussian and non-Gaussian distributions.

Considered now joint estimation of α-depth regions for $\alpha = 0.05, 0.2$ and 0.4 and for $p > 2$. Table 1 shows results for standard multivariate normally distributed observation. More detailed results are given in Figures 12 and 13 in supplementary material S.2. CPU time refers to the computational time needed per α-depth region to obtain estimates with a given precision using a single CPU core. The number of directional vectors (and thus CPU time) increases with p and estimation precision. The algorithm performs very well. For example, for dimension $p = 10$, MADE less than 0.02 is obtained in about 1.5 seconds. MADE < 0.01 could be reached in shorter CPU time than what is shown in Table 1 using a higher number of directional vectors,

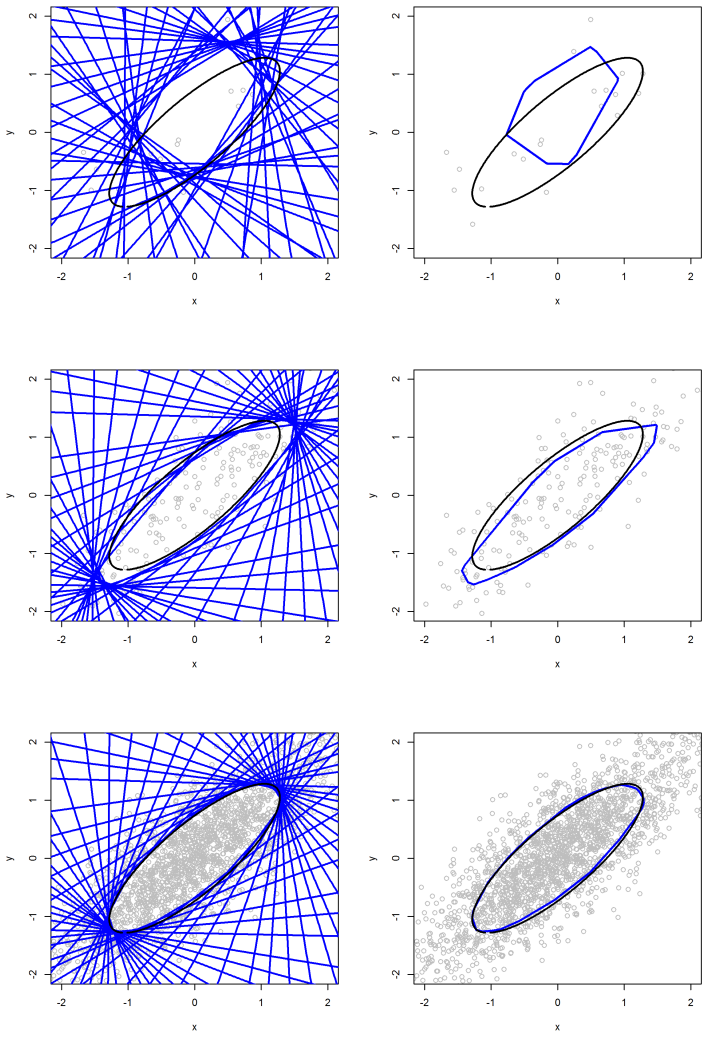

Figure 3: Multivariate normal distribution case. Estimation of α-depth region for $\alpha = 0.1$ using $n_u = 50$ directional vectors. The rows from top to bottom show the estimates for 20, 200 and 2000 observations. The left and right column show all the half planes and the resulting envelopes in blue, respectively. The black curves show the true α-depth contour.

	MADE < 0.05		MADE < 0.02		MADE < 0.01	
	CPU time	n_u	CPU time	n_u	CPU time	n_u
$p = 2$	0.00013	8	0.00174	12	0.00942	18
$p = 3$	0.00023	12	0.00429	27	0.04488	40
$p = 4$	0.00023	16	0.00958	81	0.11631	122
$p = 5$	0.00043	20	0.02991	153	0.35146	345
$p = 6$	0.00054	24	0.07419	274	0.90636	1386
$p = 8$	0.00334	72	0.34695	1228	9.83845	9324
$p = 10$	0.01361	90	1.54246	3450	104.45206	88412

Table 1: Multivariate standard normal distribution case: The second and third columns show the CPU time (in seconds) and the number of directional vectors used to obtain MADE less than 0.05. The other columns show the same to obtain MADE less than 0.02 and 0.01, respectively.

but this is not explored.

Now, assume that $X = (X_1, \ldots, X_p)^T$ is a multivariate normally distributed variable with zero expectation vector and strong dependencies

$$\mathrm{Cov}(X_i, X_j) = \exp(-0.2|i - j|),\ i, j = 1, \ldots, p \tag{10}$$

The results are shown in Table 2. More detailed results are given in Figures 14 and 15 in supplementary material S.2. By comparing Tables 1 and 2, we see that the number of directional vectors and CPU time needed increase when the variables of X are dependent.

Let X still represent the multivariate normally distributed variable with covariances (10). Table 3 shows results for the multivariate lognormal distribution $Y = \exp(X)$. More detailed results are given in Figures 16 and 17

	MADE < 0.05		MADE < 0.02		MADE < 0.01	
	CPU time	n_u	CPU time	n_u	CPU time	n_u
$p = 2$	0.00034	18	0.00622	40	0.03734	40
$p = 3$	0.00095	27	0.03003	135	1.38903	135
$p = 4$	0.00238	54	0.13145	274	7.47343	616
$p = 5$	0.01275	102	0.43698	777	8.47334	3936
$p = 6$	0.03603	183	1.81652	3118	45.88106	15786
$p = 8$	0.17285	819	23.21460	20979	988.12085	358438
$p = 10$	0.68053	2300	245.91893	198927	-	-

Table 2: Multivariate normal distribution case: The second and third columns show the CPU time (in seconds) and the number of directional vectors used to obtain a mean absolute depth error (MADE) less than 0.05, respectively.

in supplementary material S.2. Tables 2 and 3 show that a specific level of

	MADE < 0.05		MADE < 0.02		MADE < 0.01	
	CPU time	n_u	CPU time	n_u	CPU time	n_u
$p = 2$	0.00013	8	0.00957	27	0.11169	40
$p = 3$	0.00024	27	0.01533	135	0.56418	202
$p = 4$	0.00021	24	0.03214	274	1.64312	924
$p = 5$	0.00043	45	0.14592	1166	6.16044	3936
$p = 6$	0.00053	54	0.27431	2079	9.30407	15786

Table 3: Multivariate lognormal distribution case: The second and third columns shows the CPU time (in seconds) and the number of directional vectors used to obtain a mean absolute depth error (MADE) less than 0.05, respectively. The fourth and fifth and the sixth and seventh columns show the same to obtain MADE less than 0.02 and 0.01, respectively.

MADE is reached faster for the lognormal distribution than for the multivariate distribution documenting that the procedure efficiently can characterize non-Gaussian distributions.

To the best of our knowledge, the algorithm by Liu et al. (2019) is the most efficient algorithm in the literature to estimate Tukey α-depth regions. The authors focus on estimating exact trimmed α-depth regions resulting in complex combinatorial algorithms and the computation burden explodes with the number of samples. In comparison, the computational complexity of our algorithm increases linearly with the number of samples. The authors can document estimation results up to dimension $p = 9$, but only when the number of samples are restricted to less than 80. The algorithm by Liu et al. (2019) is not constructed to handle streaming data.

5.2. Synthetic Experiments - Offline Setting

In this section, we compare the performance of the incremental quantile estimator, DUMIQE, with state-of-the-art offline quantile estimators to estimate α-depth regions when data is known in advance. State-of-the-art offline quantile estimators are based on using weighted averages of consecutive order statistics

$$Q(\alpha) = (1 - \delta)y[j] + \delta y[j + 1]$$

where $\frac{j-m}{N} \leq \alpha < \frac{j-m+1}{N}$, $y[j]$ is the jth order statistic of the sample, m a constant and N the sample size. We use $m = \frac{\alpha+1}{3}$ and $\delta = N\alpha + m - j$ and define $\alpha[k] = \frac{k-1/3}{N+1/3}$. The sample quantiles can be read from a linear interpolation between the points $(\alpha[k], y[k]), k = 1, \ldots, N$. The resulting quantile estimates are approximately median-unbiased regardless of the distribution

of the data. This is the method referred to as Type 8 in the `quantile` function in R and is the one recommended by Hyndman and Fan (1996).

We consider the multivariate normal distribution case with covariance matrix as given in (10), sample sizes $N = 500$, 2000, 10^4 and $5 \cdot 10^4$ and dimensions $p = 2$ and $p = 3$. For $p = 2$ and $p = 3$, we used 1500 and 7500 directional vectors, respectively, which were sufficiently many to obtain satisfactory performance.

		$p = 2$			$p = 3$		
N	Method	MADE	ED	CPU	MADE	ED	CPU
500	Offline	14.9	43.1	0.291	16.9	40.5	1.634
	DUMIQE	25.1	63.9	0.045	34.9	69.7	0.288
2000	Offline	7.0	20.7	1.421	7.2	18.2	9.489
	DUMIQE	10.6	28.5	0.182	12.2	26.5	1.154
10^4	Offline	3.0	9.0	8.761	3.0	7.7	55.12
	DUMIQE	4.4	12.1	0.908	4.6	10.6	5.769
$5 \cdot 10^4$	Offline	1.3	4.0	52.32	1.3	3.5	326.0
	DUMIQE	1.8	5.4	4.542	2.0	4.7	28.84

Table 4: Offline experiment: Comparison of the DUMIQE estimator and the estimator recommended in Hyndman and Fan (1996) to estimate α-depth contours for $\alpha = 0.05$, 0.2 and 0.4. MADE, ED and CPU refers to the error measures in (7) and (8) (multiplied by 10^3) and CPU time used (in seconds), respectively. N refers to the sample size.

The results are shown in Table 4. We see that the estimation errors using DUMIQE are about 1.5 time that of the offline estimator. If fewer directional vectors were used, the differences in estimation error were substantially reduced. Further, the computational time of the offline estimator is about ten

times larger than the DUMIQE estimator. In other words, if computational time or memory usage are not an issue, the offline estimator combined with a large amount of directional vectors will give the most precise estimates from the samples. Otherwise, incremental quantile estimators are preferable even for offline settings.

5.3. Synthetic Experiments - Dynamically Changing Data Streams

In this section, we consider the problem of tracking α-depth regions of dynamically varying data streams. Figure 4 illustrates the problem. In each panel, the expectation vector of the data stream distribution moved from the bottom left to the upper right. At the same time the correlation, changed from strongly positive, 0.8, to strongly negative, -0.8. For the 10^3 samples case (first row), the algorithm was able to track the α-depth regions satisfactory. With 10^4 samples, the estimates improve significantly and with 10^5 observations, the estimates are very close to the true contours. With 10^4 and 10^5 samples, 50 directional vectors give better and smoother estimates than 10 directional vectors.

Evaluation for $p > 2$ is given below. Due to the computational burden of evaluating estimation error of non-elliptic distributions, the analysis was restricted to Gaussian distributions. Let $X_n = (X_{n,1}, \ldots, X_{n,p})^T$ be multivariate normally distributed with

$$\mu_{n,i} = E(X_{n,i}) = \sin\left(\frac{2\pi}{T}n + \psi_i\right), \quad i = 1, \ldots, p \tag{11}$$

where $\psi_i,\ i = 1, \ldots, p$ are independent uniformly distributed variables on the interval $[0, 2\pi]$ ensuring that the marginal expectations are out of phase.

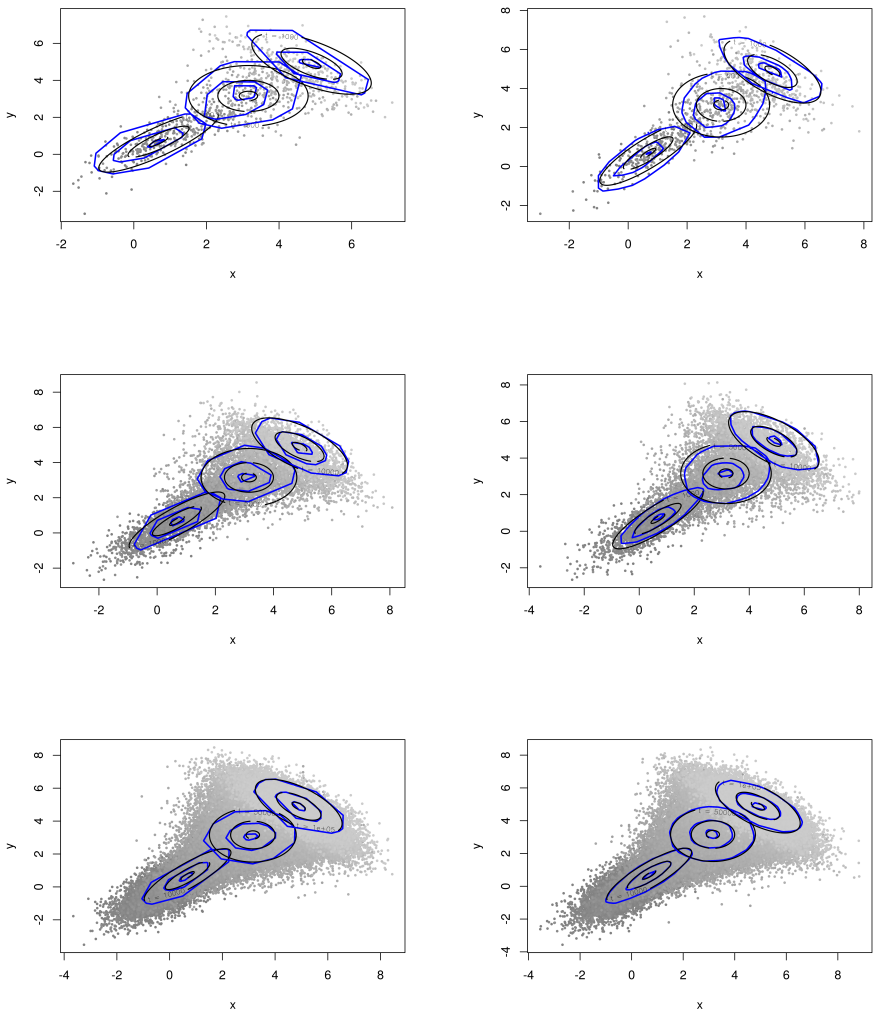

Figure 4: Tracking of α-depth contours for $\alpha = 0.05$, 0.2 and 0.4: In each panel the gray dots are outcomes from the data stream. The first observations from the data stream are shown in dark gray and the dots become lighter gray as time progresses. The left and right column show cases with $n_u = 10$ and 50 directional vectors, respectively. The rows from top to bottom show cases with a total for 10^3, 10^4 and 10^5 observations, respectively.

Covariance between $X_{n,i}$ and $X_{n,j}$ is

$$\mathrm{Cov}(X_{n,i}, X_{n,j}) = \left(0.4 \sin\left(\frac{2\pi}{T} n + \psi\right) + 0.4\right)^{|i-j|} \tag{12}$$

where ψ is uniformly distributed on the interval $[0, 2\pi]$.

Tables 5 to 6 show results tracking α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for periods $T = 10^3$ and $T = 10^4$ under optimal choices of the tuning parameter[1]. More detailed results are given in Figures 18 and 19 in supplementary material S.3. For $T = 10^3$, MADE is around 0.05 and the estimation error does not decrease with increasing number of directional vectors which may seem surprising. The reason is that if the quantile estimates are poor, the intersections of the resulting halfspaces do not necessarily become better by adding more halfspaces. For $T = 10^4$ MADE is between 0.02 and 0.03. The optimal number of halfspaces increases with dimension, but not dramatically.

The algorithm is computationally very efficient. For dimension $p = 5$ the algorithm can optimally process 10^4 to 10^5 observations from a data stream every second on a single CPU processor.

By using more equidistant directional vectors, we expect reduction in the tracking error. Consider the dynamic case above except that the directional vectors are chosen more equidistantly. Directional vectors were generated using the filtering procedure in Section 3 with $N_u = 10n_u$.

The results are shown in Table 7 and more detailed results are given

[1]In a practical situation, the history of the data stream can be used to estimate (or track) optimal values of the tuning parameters. We are currently working on such procedures.

	$p = 2$		$p = 3$		$p = 4$		$p = 5$	
n_u	MADE	Freq	MADE	Freq	MADE	Freq	MADE	Freq
5	0.0559	972.7	–	–	–	–	–	–
10	0.0475	478.9	0.0577	486.7	–	–	–	–
25	**0.0445**	189.2	**0.0457**	189.7	0.0510	184.7	–	–
50	0.0474	95.2	0.0467	95.1	**0.0492**	93.3	**0.0521**	92.4
100	0.0504	47.9	0.0502	47.4	0.0514	46.8	0.0523	46.4
200	–	–	0.0536	23.4	0.0546	22.6	0.0541	22.7
500	–	–	–	–	0.0590	9.1	0.0576	8.9
1000	–	–	–	–	–	–	0.0604	4.5

Table 5: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12) with a period $T = 10^3$. The columns 'Freq' refer to how many times per millisecond the algorithm can update an α-depth region when running on a single 1.8 GHz CPU processor.

in Figure 21 in supplementary material S.3. By comparing Tables 5 and 6 with 7, we see that for $T = 10^3$ and $T = 10^4$, minimum MADE is reduced from 0.045 to 0.040 and from 0.0226 to 0.0216, respectively. However, more importantly, by using equidistant vectors, the best results are obtained using fewer directional vectors. For both $T = 10^3$ and $T = 10^4$, the optimal number of vectors are reduced from 25 to 10. Finally, we observe significant improvement if only five directional vectors were used. Using equidistant directional vectors adds an additional computational cost in the initialization of the algorithm, but results into gained peak performance and fewer directional vectors, and thus less computation time and memory are needed during tracking.

	$p = 2$		$p = 3$		$p = 4$		$p = 5$	
n_u	MADE	Freq	MADE	Freq	MADE	Freq	MADE	Freq
5	0.0439	976.3	–	–	–	–	–	–
10	0.0305	480.1	0.0499	484.3	–	–	–	–
25	**0.0226**	189.2	0.0318	188.5	0.0429	184.7	–	–
50	0.0227	95.4	0.0277	94.3	0.0342	93.6	0.0395	91.1
100	0.0236	48.0	**0.0275**	47.3	0.0306	47.0	0.0337	46.1
200	–	–	0.0289	23.3	**0.0299**	22.7	0.0312	22.6
500	–	–	–	–	0.0307	9.1	**0.0309**	9.0
1000	–	–	–	–	–	–	0.0316	4.5

Table 6: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12) with a period $T = 10^4$. The columns 'Freq' refer to how many times per millisecond the algorithm can update an α-depth region when running on a single 1.8 GHz CPU processor.

n_u	$T = 10^3$	$T = 10^4$
5	0.0465	0.0303
10	**0.0400**	**0.0216**
25	0.0440	0.0217
50	0.0477	0.0226
100	0.0505	0.0236

Table 7: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12) using fairly equidistant directional vectors. Tracking error is measured using MADE. The left and right columns show results for $T = 10^3$ and 10^4, respectively. Dimension is $p = 2$.

6. Real-life Data Examples

In this section, we use the algorithm on a real-life dataset related to activity change detection. A second real-life data example related to real-time event detection using Twitter data is given in supplementary material S.4.

We demonstrate how the algorithm can be used to detect outliers and events and perform classifications in dynamic settings. For example, related to event detection, by characterizing a data stream distribution with multiple depth contours, in practice *any* change in the data stream distribution can be detected. Not only changes in common properties such as expectation and covariance structure, but also changes in shape such as a change from an elliptic to a non-elliptic distribution.

6.1. Activity Change Detection

Activity recognition is a highly active field of research where sensory information is used to automatically detect and identify activities of users. Activity recognition can help for example detect sedentary lifestyle and prompt the user to perform healthy exercises.

We consider an accelerometer dataset from the WISDM (Wireless Sensor Data Mining) project (Kwapisz et al., 2011). Accelerations in x, y and z directions were observed, with a frequency of 20 observations per second, while users were performing the activities walking, jogging, walking up a stairway and walking down a stairway. A total of 36 users were observed and the dataset contains a total of 989 875 observations.

Current research focuses on supervised approaches where historic and

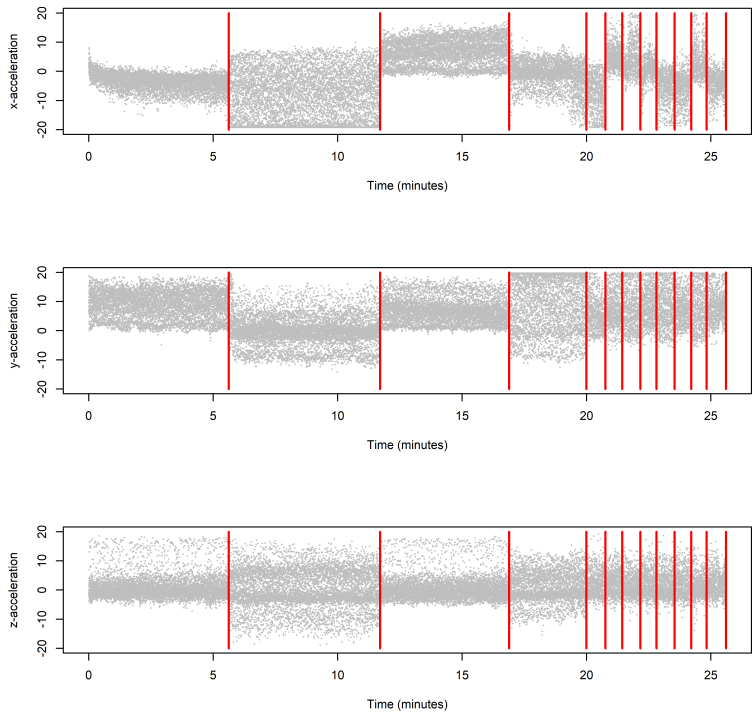

Figure 5: The gray dots show accelerometer observations for an arbitrary user. The red lines show when the user changes activity.

annotated activity observations are used to train an activity classification model. E.g. Kwapisz et al. (2011) trained models such as decision trees and neural networks. However, such an approach is highly sensitive to any temporal changes in the data, e.g. if the user switches to an activity that is not part of the training material as a consequence for example of becoming fitter, sick etc. In this example we rather take an unsupervised approach and the goal is to detect whenever the user changes activity. Since we receive 20 accelerometer observations per second, it is important that the streaming approach is computationally efficient.

Figure 5 shows in gray x, y and z acceleration for an arbitrary user. The red lines show when the user changed activity. Acceleration distributions are fairly stationary within an activity, but with some gradual and abrupt changes. The users changed activities in many cases as often as every 30 seconds making this a challenging tracking and change detection problem. Figure 6 shows scatter plots for two arbitrary sessions with minimal temporal trend. The simultaneous acceleration distributions vary substantially between sessions and are often far from being elliptical. Further, even though the distributions are different, the mean and covariances are often quite similar making the change detection task based on elliptic distributions (Mahalanobis distance) challenging. We thus suggest the following depth based change detection procedure:

1. Track α-depth contours of the simultaneous acceleration distribution by tracking n_u directional quantiles using the DUMIQE algorithm with tuning parameter λ.
2. Compute the Euclidean distance between the current α-depth contours and the contours h seconds back in time using Equation (8), denoted ED_t at time t.
3. Track the expectation and standard deviation of ED_t distribution using exponential moving average

$$
\begin{aligned}
E(\mathrm{ED}_t) &= (1-\delta)E(\mathrm{ED}_{t-1}) + \delta \mathrm{ED}_t \\
E(\mathrm{ED}_t^2) &= (1-\delta)E(\mathrm{ED}_{t-1}^2) + \delta \mathrm{ED}_t^2 \\
SD(\mathrm{ED}_t) &= \sqrt{E(\mathrm{ED}_t)^2 - E(\mathrm{ED}_t^2)}
\end{aligned}
$$

4. When the user changes activity, we expect ED_t to rapidly increase. A

new activity is detected when ED_t is more than η standard deviations higher then $E(\mathrm{ED}_t)$, i.e. $\mathrm{ED}_t \geq E(\mathrm{ED}_t) + \eta\, SD(\mathrm{ED}_t)$.

5. When a new activity is detected, restart the tracking of the α-depth contours and go back to step 1.

This approach is elegant since by virtue of measuring difference in depth contours, it can detect virtually *any* kind of change in the shape of the simultaneous acceleration distribution, for example a change from an elliptic to a non-elliptic distribution. Given the properties of the observations in this application, this flexibility is important.

We compare the approach against an identical approach except that in the first part of the algorithm the mean and covariance structure (and not depth contours) were tracked using multivariate exponentially weighted moving average (MEWMA) (Lowry et al., 1992).

We measured the performance of the depth and the MEWMA approaches for a wide range of values for the tuning parameters. As several sessions lasted for only 30 seconds, it was thus important for the tracking algorithms to rapidly adapt to a session before a new change of activity took place. In the first step of the procedures, we thus chose decreasing values of the tuning parameters, but with a minimum value to take into account the dynamic changes in accelerations within a session, $\lambda_t = \max\{1/t, \lambda_{\min}\}$, and tried the values 0.1, 0.05 and 0.01 for ${\lambda_{\min}}^2$. This performed better than using constant values of the tuning parameter. We further tried the values 0.1,

[2]For MEWMA, λ_t refers to the moving average tuning parameter and $\lambda_t = 1/t$ is thus equivalent to the sample mean.

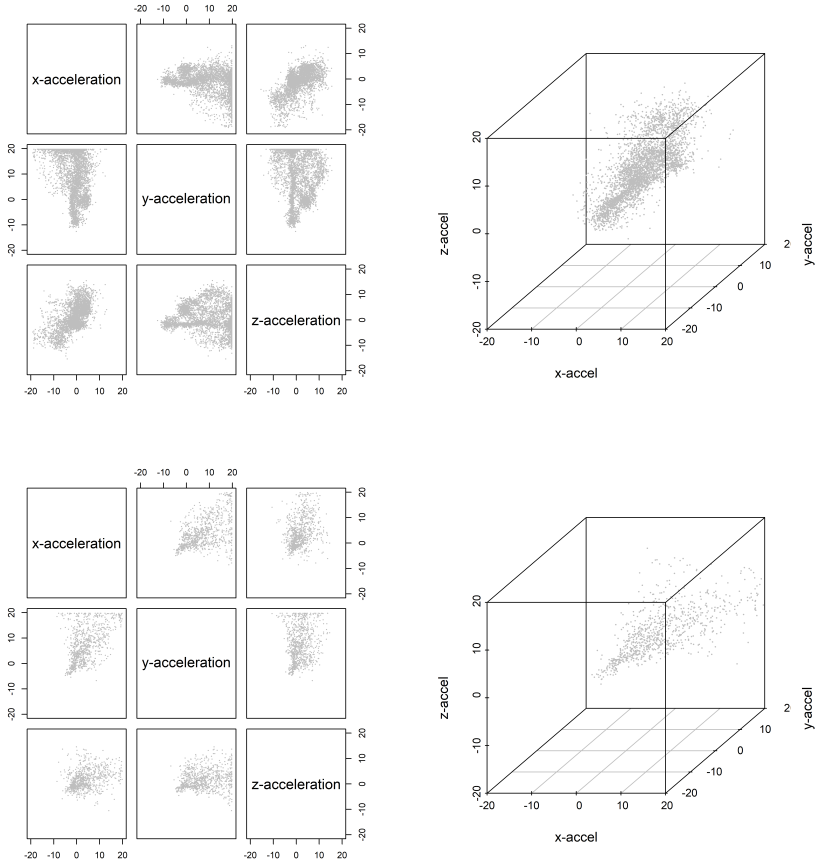

Figure 6: The first and the second row show scatterplot of accelerometer observations for two activity sessions.

0.05 and 0.01 for δ, 2.5, 5 and 10 seconds for h and 2, 5 and 8 for η. Further, for the depth approach we used three depth contours with α equal to 0.2, 0.05 and 0.01 and tried $n_u = 20$ or 50 directional vectors. We ran the two change detection approaches for the whole dataset for all the combinations of the parameters. This resulted in a total of 162 and 81 experiments for the depth and the MEWMA approaches, respectively.

Precision, recall and the F1 score were used to measure performance (Sokolova and Lapalme, 2009). If the approach detects more than one change between two true changes, we characterize the first change as a correct detection and the others as false detections and define

$$\text{Precision} = \frac{\text{No. of correct detections}}{\text{No. of detections}}$$

$$\text{Recall} = \frac{\text{No. of correct detections}}{\text{No. true changes}}$$

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Tables 8 and 9 show the top ten results with respect to the F1 score. The depth approach outperforms the MEWMA with respect to the F1 score and in addition detects the true changes more rapidly. The performance of the depth approach does not seem to be particularly sensitive to the number of directional quantiles used.

7. Closing Remarks

In this paper we have presented a computationally and memory efficient procedure to estimate and track Tukey α-depth contours using incremental quantile estimators. The algorithms use the results by Kong and Mizera (2012) according to which the α-depth region equals the directional quantile envelope. We further demonstrated how incremental quantile estimators can be used to efficiently estimate the directional quantile envelope. By using incremental quantile estimators, we are able to recursively estimate and track α-depth contours, and to the best of our knowledge, it is the first method in the literature with this ability. However, as shown in Section 5.2, if the

$\lambda_{\min}$	$\delta_{\min}$	h	η	n_u	Precision	Recall	F1 score	Det. delay (sec)
0.01	0.01	100	8	20	0.796	0.497	**0.612**	**1.325**
0.01	0.01	100	8	50	0.781	0.486	**0.599**	**1.150**
0.01	0.05	200	8	20	0.532	0.682	**0.597**	**1.415**
0.01	0.05	50	8	50	0.613	0.564	**0.587**	**1.876**
0.01	0.01	200	8	20	0.735	0.488	**0.587**	**1.202**
0.01	0.05	200	8	50	0.510	0.673	**0.580**	**1.482**
0.01	0.01	200	8	50	0.719	0.480	**0.575**	**1.219**
0.01	0.05	100	8	20	0.538	0.618	**0.575**	**1.010**
0.05	0.10	200	8	20	0.539	0.616	**0.575**	**1.903**
0.01	0.05	50	8	20	0.613	0.532	**0.570**	**1.894**

Table 8: Change detection example. Results for the depth approach.

amount of data is limited, it is better to estimate α-depth contours using traditional offline quantile estimators.

The algorithms estimated Tukey depth contours equally well for both elliptic (Gaussian) and non-elliptic distributions. The performance, however, depends on the degree of curvature for the true depth contours being closely related to the degree of dependency between variables. For static data streams, the algorithm estimated a depth contour of dimension $p = 10$ with a mean absolute error in Tukey depth less than 0.01 in 1.2 and 125 minutes for independent and strongly dependent variables, respectively, on a single CPU processor. For dynamically changing data streams, even for dimensions as high as $p = 5$, the algorithm was able to process tens of thousands of observations per second and track depth contours with high precision. We have

$\lambda_{\min}$	$\delta_{\min}$	h	η	Precision	Recall	F1 score	Det. delay (sec)
0.01	0.01	200	8	0.454	0.697	**0.550**	**1.553**
0.05	0.01	200	8	0.447	0.697	**0.545**	**1.691**
0.05	0.01	50	8	0.438	0.699	**0.539**	**2.249**
0.01	0.01	50	8	0.421	0.711	**0.529**	**1.736**
0.01	0.01	100	8	0.398	0.737	**0.517**	**1.293**
0.05	0.01	100	8	0.388	0.711	**0.502**	**1.747**
0.05	0.05	200	8	0.353	0.760	**0.483**	**1.525**
0.05	0.05	50	8	0.336	0.818	**0.476**	**1.832**
0.05	0.10	200	8	0.336	0.803	**0.474**	**1.522**
0.01	0.05	200	8	0.330	0.777	**0.463**	**1.281**

Table 9: Change detection example. Results for the MEWMA approach.

not found any studies that have been able to estimate depth contours of such a high dimension and for such a large of amount of data, which documents the efficiency of the algorithm.

The real-life data examples demonstrate that the procedure is useful to track and detect changes in complex distributional patterns.

To estimate α-depth contours, the number of directional vectors, n_u, and values of tuning parameters in the incremental quantile tracking algorithms must be chosen. We are currently working on procedures that use information from the history of the data stream to recursively update such values. Tukey depth is best suited to account for convex features of the distribution of interest. However, there exist other modified depth measures that better account for non-convex features (Chernozhukov et al., 2017). In the future,

we plan to extend the method in this paper in order to also be applied to these depth measures.

Alkhamees, N., Fasli, M., 2016. Event detection from social network streams using frequent pattern mining with dynamic support values. In: Big Data (Big Data), 2016 IEEE International Conference on. IEEE, pp. 1670–1679.

Atefeh, F., Khreich, W., 2015. A survey of techniques for event detection in twitter. Computational Intelligence 31 (1), 132–164.

Cerdeira, J. O., Monteiro-Henriques, T., Martins, M. J., Silva, P. C., Alagador, D., Franco, A. M., Campagnolo, M. L., Arsénio, P., Aguiar, F. C., Cabeza, M., 2018. Revisiting niche fundamentals with tukey depth. Methods in Ecology and Evolution 9 (12), 2349–2361.

Chang, Y., Tu, Z., Xie, W., Luo, B., Zhang, S., Sui, H., Yuan, J., 2021. Video anomaly detection with spatio-temporal dissociation. Pattern Recognition, 108213.

Chebana, F., Ouarda, T. B., 2011. Depth-based multivariate descriptive statistics with hydrological applications. Journal of Geophysical Research: Atmospheres 116 (D10).

Chernozhukov, V., Galichon, A., Hallin, M., Henry, M., et al., 2017. Monge–kantorovich depth, quantiles, ranks and signs. Annals of Statistics 45 (1), 223–256.

Eddelbuettel, D., 2013. Seamless R and C++ Integration with Rcpp. Springer, New York, iSBN 978-1-4614-6867-7.

Eddelbuettel, D., François, R., 2011. Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40 (8), 1–18.
URL http://www.jstatsoft.org/v40/i08/

Erfani, S. M., Rajasegarar, S., Karunasekera, S., Leckie, C., 2016. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. Pattern Recognition 58, 121–134.

Hammer, H. L., Yazidi, A., Rue, H., 2019. A new quantile tracking algorithm using a generalized exponentially weighted average of observations. Applied Intelligence 49 (4), 1406–1420.

Hammer, H. L., Yazidi, A., Rue, H., 2021. Joint tracking of multiple quantiles through conditional quantiles. Information Sciences 563, 40–58.

Hasan, M., Orgun, M. A., Schwitter, R., 2017. A survey on real-time event detection from the twitter data stream. Journal of Information Science, 0165551517698564.

Huang, S., Kang, Z., Xu, Z., Liu, Q., 2021. Robust deep k-means: An effective and simple method for data clustering. Pattern Recognition 117, 107996.

Hubert, M., Rousseeuw, P., Segaert, P., 2017. Multivariate and functional classification using depth and distance. Advances in Data Analysis and Classification 11 (3), 445–466.

Hubert, M., Rousseeuw, P. J., Segaert, P., 2015. Multivariate functional outlier detection. Statistical Methods & Applications 24 (2), 177–202.

Hyndman, R. J., Fan, Y., 1996. Sample quantiles in statistical packages. The American Statistician 50 (4), 361–365.

Iwana, B. K., Uchida, S., 2020. Time series classification using local distance-based features in multi-modal fusion networks. Pattern Recognition 97, 107024.

Jörnsten, R., 2004. Clustering and classification based on the l1 data depth. Journal of Multivariate Analysis 90 (1), 67–89.

Kim, S., Mun, B. M., Bae, S. J., 2018. Data depth based support vector machines for predicting corporate bankruptcy. Applied Intelligence 48 (3), 791–804.

Kong, L., Mizera, I., 2012. Quantile tomography: using quantiles with multivariate data. Statistica Sinica, 1589–1610.

Kosiorowski, D., Zawadzki, Z., 2014. Depthproc an r package for robust exploration of multidimensional economic phenomena. arXiv preprint arXiv:1408.4542.

Kwapisz, J. R., Weiss, G. M., Moore, S. A., 2011. Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter 12 (2), 74–82.

Liu, X., Mosler, K., Mozharovskyi, P., 2019. Fast computation of tukey trimmed regions and median in dimension $p > 2$. Journal of Computational and Graphical Statistics, 1–31.

Lowry, C. A., Woodall, W. H., Champ, C. W., Rigdon, S. E., 1992. A multivariate exponentially weighted moving average control chart. Technometrics 34 (1), 46–53.

Ma, J., Zhang, Y., Zhang, L., 2021. Discriminative subspace matrix factorization for multiview data clustering. Pattern Recognition 111, 107676.

Massé, J.-C., 2004. Asymptotics for the tukey depth process, with an application to a multivariate trimmed mean. Bernoulli, 397–419.

Mosler, K., 2013. Depth statistics. In: Robustness and complex data structures. Springer, pp. 17–34.

R Core Team, 2021. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
URL `https://www.R-project.org/`

Rastin, N., Jahromi, M. Z., Taheri, M., 2021. A generalized weighted distance k-nearest neighbor for multi-label problems. Pattern Recognition 114, 107526.

Saff, E. B., Kuijlaars, A. B., 1997. Distributing many points on a sphere. The mathematical intelligencer 19 (1), 5–11.

Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information Processing & Management 45 (4), 427–437.

Tukey, J. W., 1975. Mathematics and the picturing of data. In: Proceedings of the international congress of mathematicians. Vol. 2. pp. 523–531.

Williams, B., Toussaint, M., Storkey, A. J., 2008. Modelling motion primitives and their timing in biologically executed movements. In: Advances in neural information processing systems. pp. 1609–1616.

Yazidi, A., Hammer, H., 2017. Multiplicative update methods for incremental quantile estimation. IEEE transactions on cybernetics 49 (3), 746–756.

Zavrtanik, V., Kristan, M., Skočaj, D., 2021. Reconstruction by inpainting for visual anomaly detection. Pattern Recognition 112, 107706.

Zuo, Y., Serfling, R., 2000. General notions of statistical depth function. Annals of statistics, 461–482.

Supplementary Material

S.1. Computation of Depth Error for Elliptic Distributions

For elliptic distributions, the α-depth contours coincide with the contours of the distribution and, secondly, the Tukey halfplanes are tangent planes to the contours (Kong and Mizera, 2012). For a multivariate normal distribution with expectation vector μ and covariance matrix Σ, the depth of any point w can be found analytically. First, find the inward pointing unit length normal vector to the tangent plane at w:

$$\operatorname{tang}(w) = -\frac{\Sigma^{-1}(w-\mu)}{\|\Sigma^{-1}(w-\mu)\|_2}$$

and then compute the depth as the probability to be within the tangent plane halfspace defined by $\operatorname{tang}(w)$

$$D(w,P) = \Phi\left(\operatorname{tang}(w)^T w; \operatorname{tang}(w)^T \mu, \sqrt{\operatorname{tang}(w)^T \Sigma^{-1} \operatorname{tang}(w)}\right)$$

where $\Phi(\,\cdot\,; m, s)$ refers to the cumulative distribution function of the univariate normal distribution with expectation m and standard deviation s.

S.2. Synthetic Experiments - Static Data Streams - Additional Results

Let X refer to a multivariate normal distribution defined and define the lognormal distribution $Y = \exp(X)$. Y is both highly unsymmetrical and highly heavy tailed. The results are shown in Figure 7. Due to the flexibility of the depth concept, the method performs equally well for non-elliptic distributions.

Recall the multivariate normal and lognormal static data streams considered in Section 5.1. Figures 8 to 11 show results using the ShiftQ algorithm

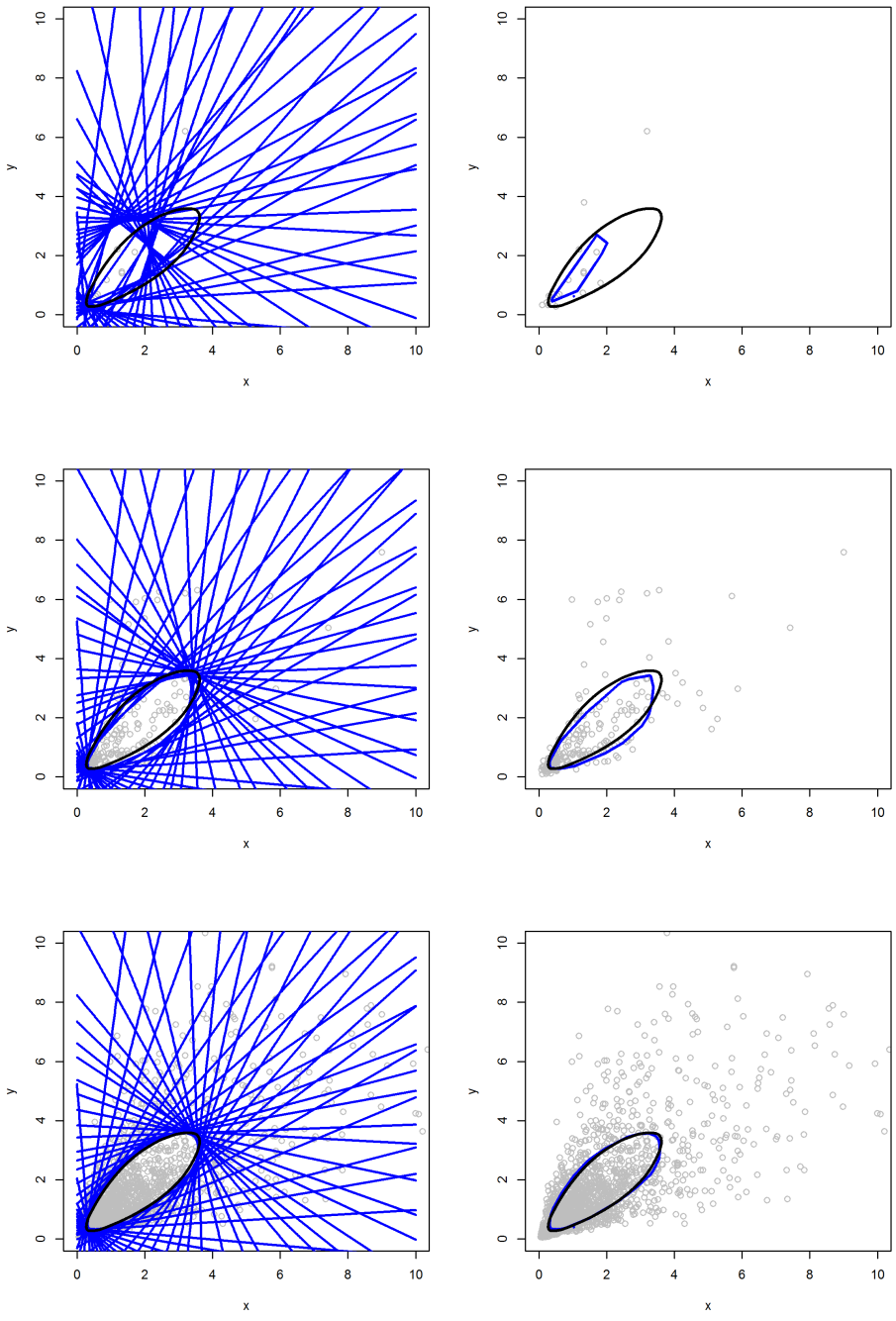

Figure 7: Multivariate lognormal distribution case. Estimation of α-depth region for $\alpha = 0.1$ using $n_u = 50$ directional vectors. The rows from top to bottom show estimates for 20, 200 and 2000 observations. The left and right column show all the half planes and the resulting envelopes in blue, respectively. The black curves show the true α-depth contour.

with $\lambda_n = 1/n$ to estimate α-depth regions for $\alpha = 0.05$, 0.2 and 0.4. We see that the algorithm converges well for both Gaussian and non-Gaussian distributions. For the non-Gaussian case, we see that the true α-depth contours (black) are not smooth which may seem surprising. However, it is only for elliptic distributions that the α-depth regions guaranteed are smooth, see Kong and Mizera (2012) and Massé (2004) for details and more examples.

Figure 12 and 13 show the results where P was the standard multivariate normal distribution, i.e. X was a vector of independent standard normally distributed stochastic variables. The CPU time refers to the time to estimate one α-depth region to the given precision running on a single CPU core. To remove Monte Carlo error from the results, the computations were run for a total of $3 \cdot 10^6$ iterations and in the computation of MADE and ED in (7) and (8) a total of $n_v = 4p$ lines where used. From the figures, we see that with a given number of directional vectors, the estimation precision reaches a limit after some time and more directional vectors are necessary to get an estimate with higher precision. For most practical purposes, a mean absolute depth error of 0.01 is satisfactory and from Figure 12, we see that the number of directional vectors and the resulting CPU time to reach this precision depends strongly on the dimension p. E.g. for $p = 2$, $p = 4$ and $p = 10$, we reached this precision in 0.02 seconds, 0.1 second and 1.2 minutes, respectively. Reaching this precision in a little over one minute for dimension $p = 10$ is quite impressive.

We now present results for the multivariate normal distribution case with strong dependencies as given by Equation 10. The results are shown in Figures 14 and 15. By comparing Figures 12 and 13 with 14 and 15, we

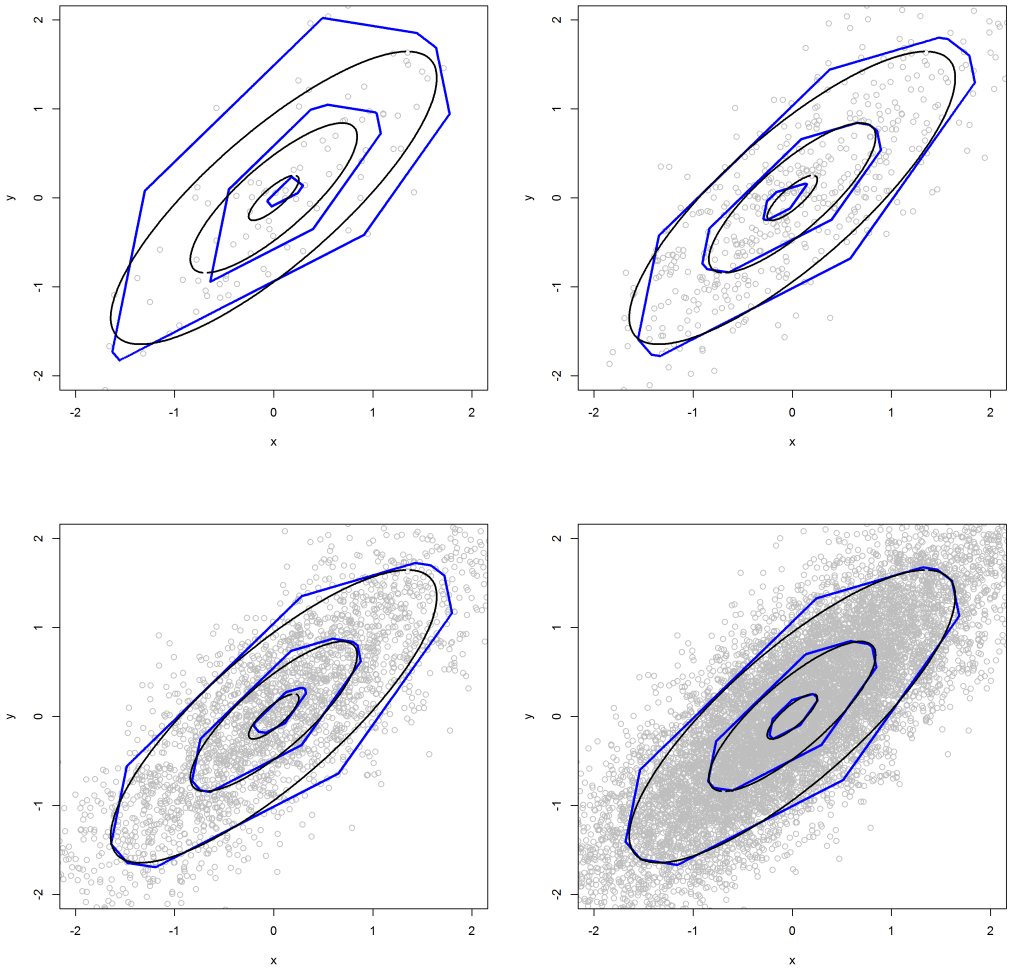

Figure 8: Multivariate normal distribution case. Simultaneous estimation of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 using $n_u = 10$ directional vectors. The panels from top left to bottom right show the estimation when 100, 500, 2500 and 10^4 observations were received from the data stream. The black and blue curves show the true depth contours and the envelope estimates, respectively.

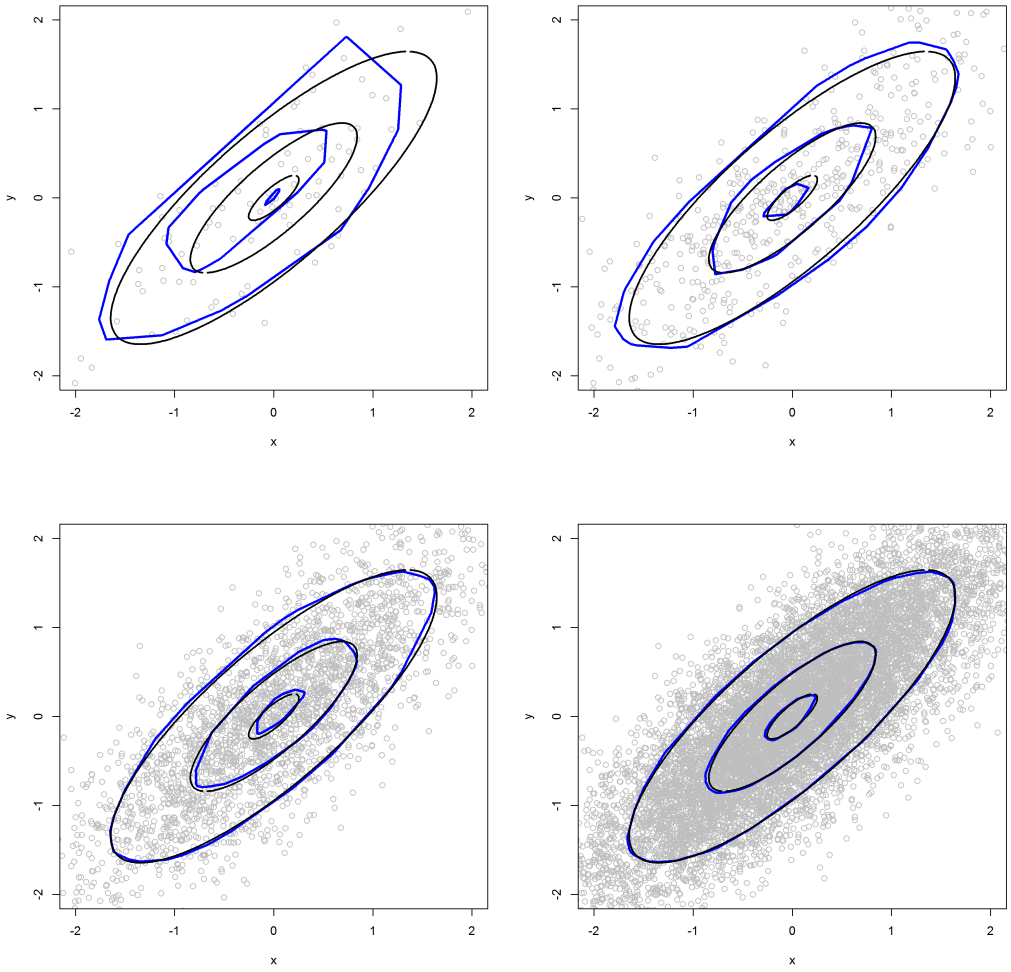

Figure 9: Multivariate normal distribution case. Simultaneous estimation of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 using $n_u = 50$ directional vectors. The panels from top left to bottom right show the estimation when 100, 500, 2500 and 10^4 observations were received from the data stream. The black and blue curves show the true depth contours and the envelope estimates, respectively.

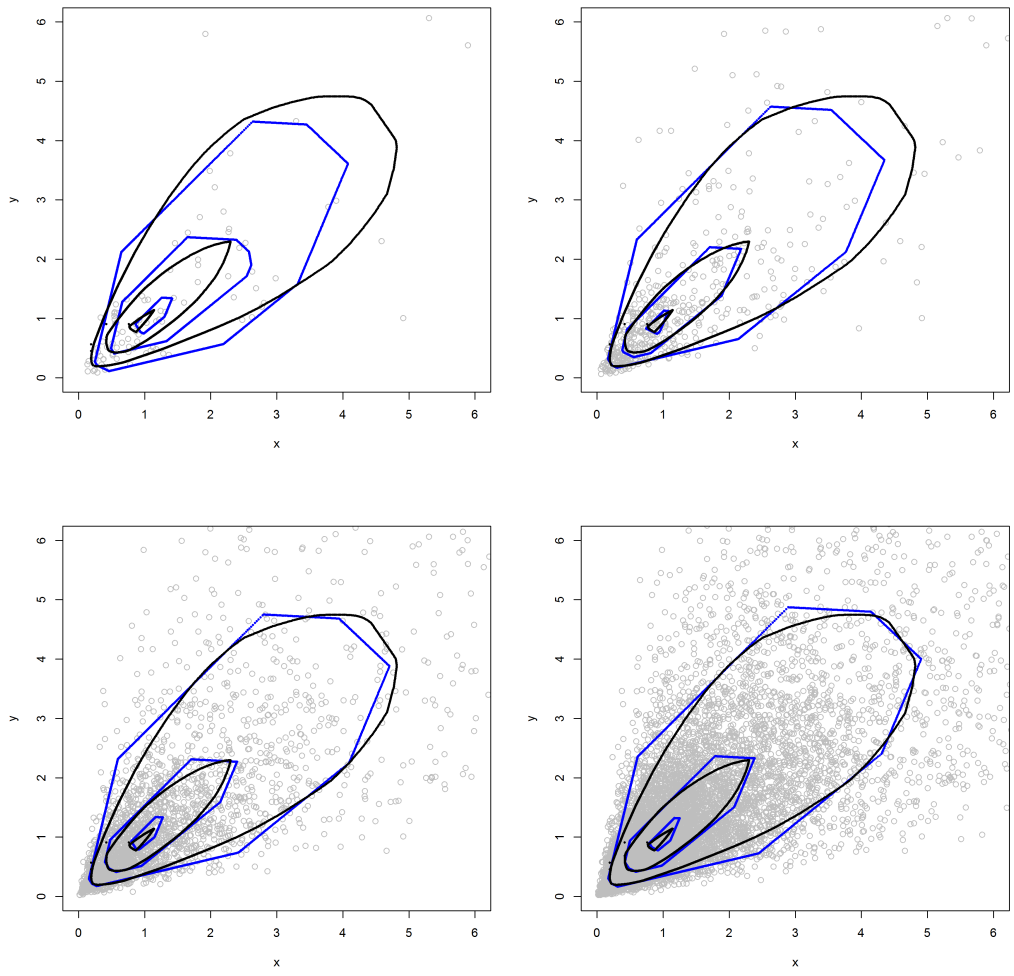

Figure 10: Multivariate lognormal distribution case. Simultaneous estimation of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 using $n_u = 10$ directional vectors. The panels from top left to bottom right show the estimation when 100, 500, 2500 and 10^4 observations were received from the data stream. The black and blue curves show the true depth contours and the envelope estimates, respectively.

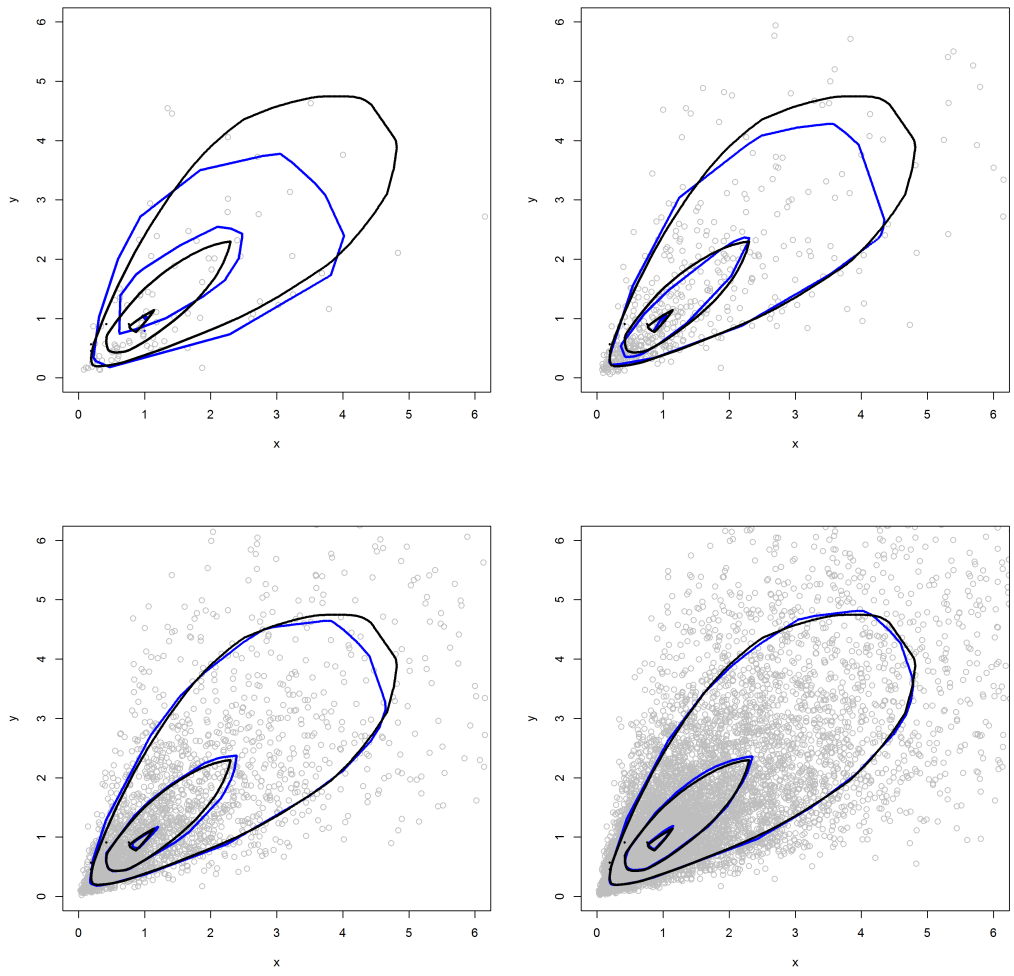

Figure 11: Multivariate lognormal distribution case. Simultaneous estimation of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 using $n_u = 50$ directional vectors. The panels from top left to bottom right show the estimation when 100, 500, 2500 and 10^4 observations were received from the data stream. The black and blue curves show the true depth contours and the envelope estimates, respectively.

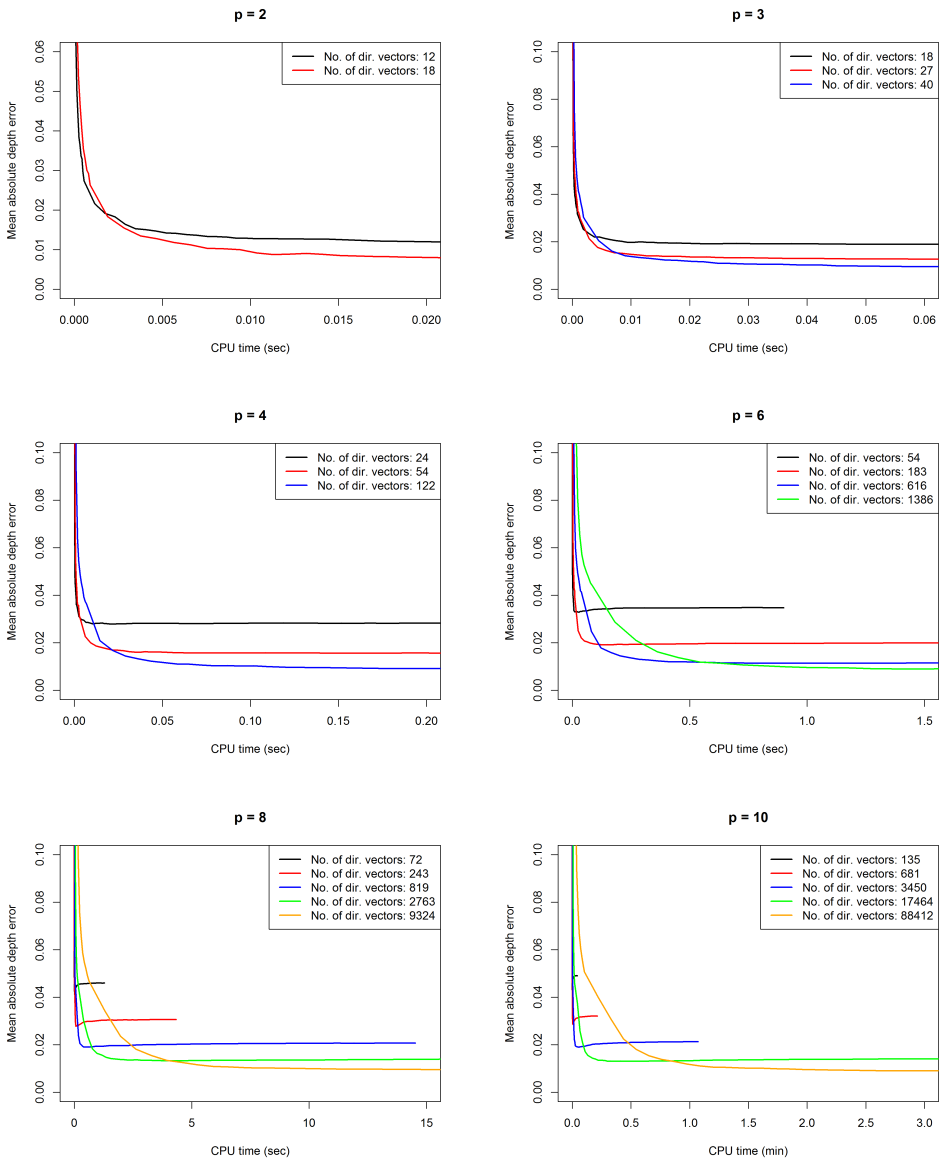

Figure 12: Multivariate standard normal distribution case: Each panel shows mean absolute depth error, Equation (7), as a function of CPU time for different number of directional vectors.

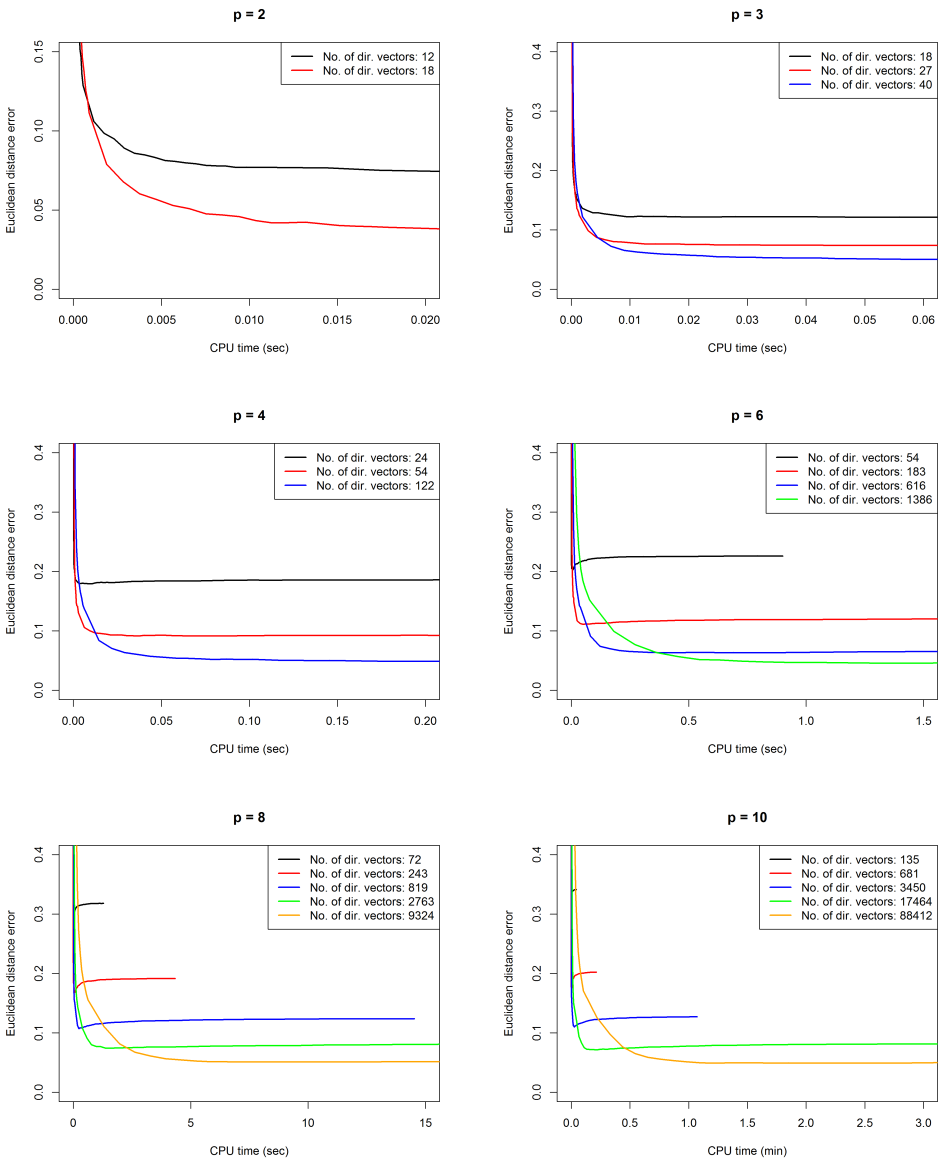

Figure 13: Multivariate standard normal distribution case: Each panel shows Euclidean distance error, Equation (8), as a function of CPU time for different number of directional vectors.

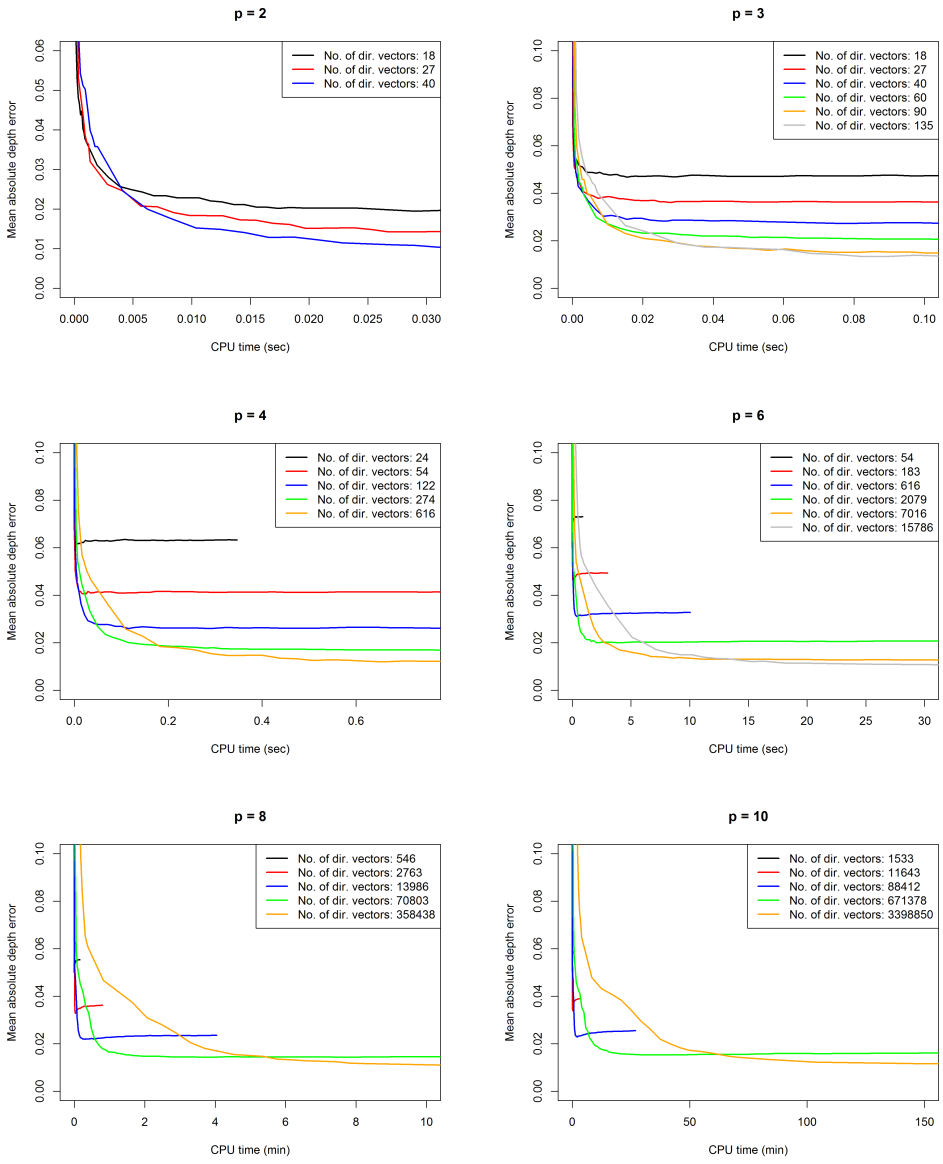

Figure 14: Multivariate normal distribution case: Each panel shows mean absolute depth error, Equation (7), as a function of CPU time for different number of directional vectors.

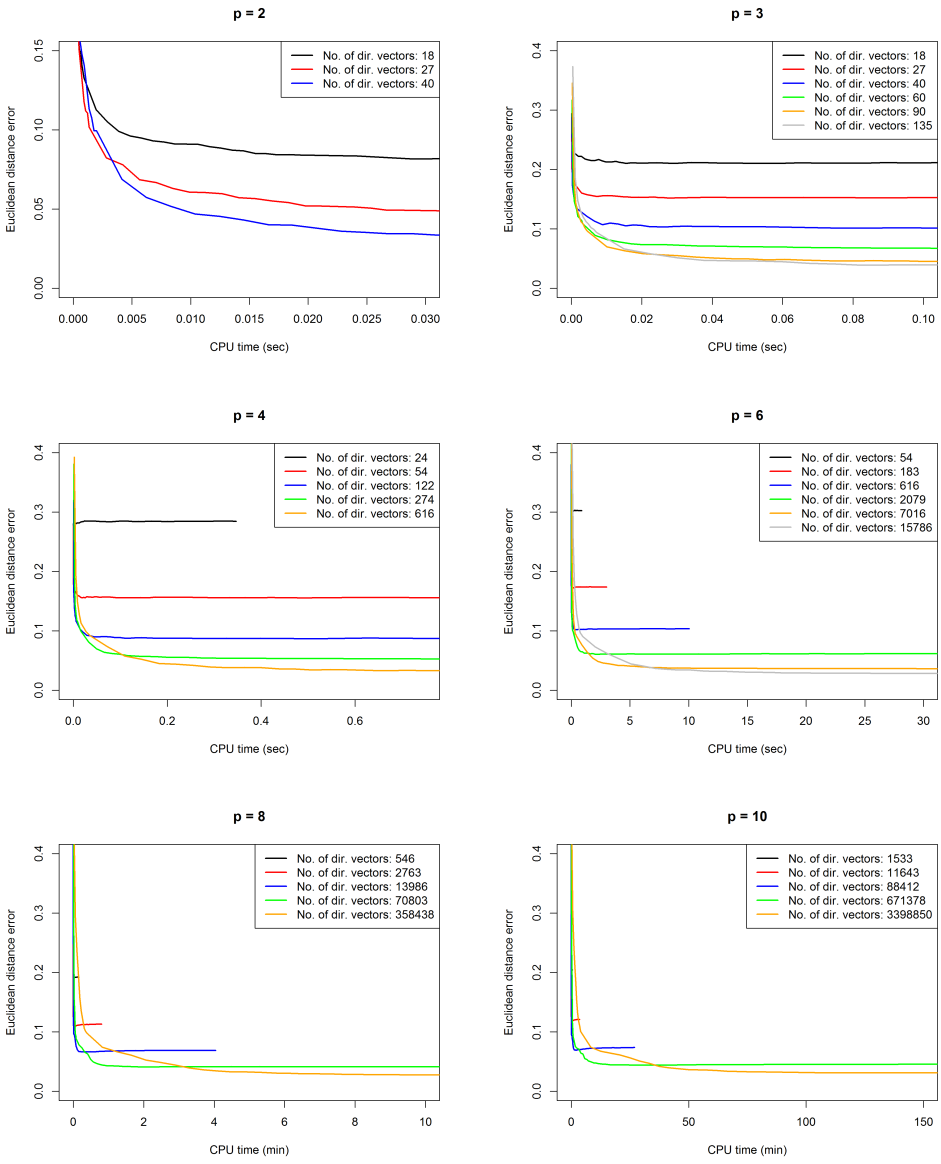

Figure 15: Multivariate normal distribution case: Each panel shows Euclidean distance error, Equation (8), as a function of CPU time for different number of directional vectors.

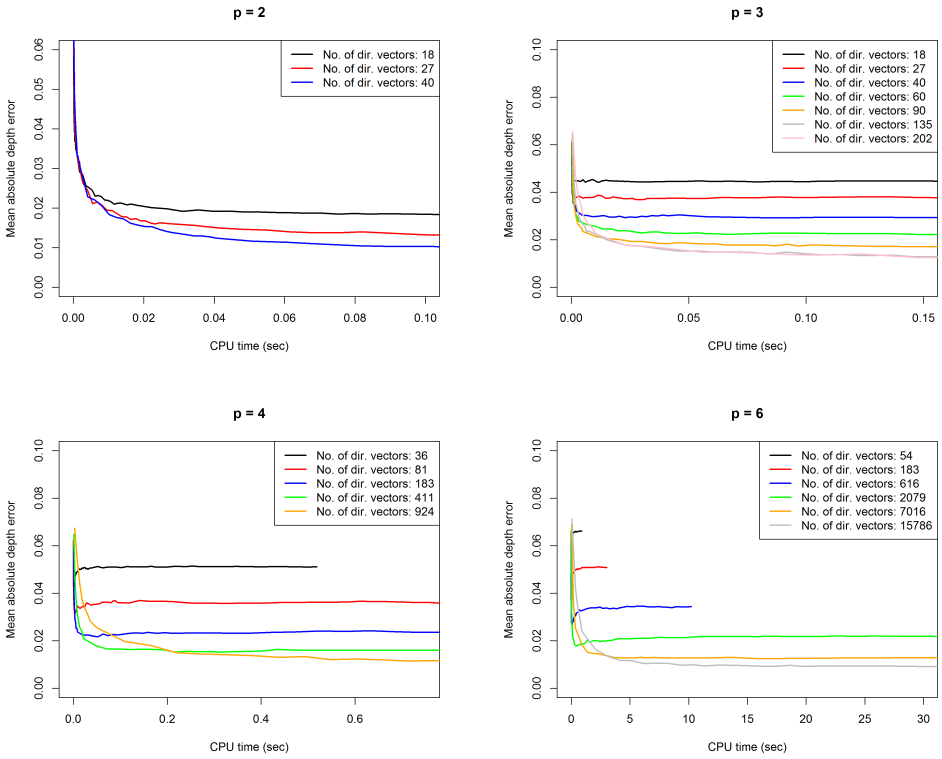

Figure 16: Multivariate lognormal distribution case: Each panel shows mean absolute depth error, Equation (7), as a function of CPU time for different number of directional vectors.

see that the number of directional vectors and CPU time needed depend strongly on the dependence structure of X. E.g. for dimensions $p = 2$, 4 and 10, the computation time to reach a MADE of 0.01 goes from 0.02 to 0.03 seconds, from 0.1 to 0.8 seconds and from 1.2 to 125 minutes by just going from independence to dependence between the variables.

Figures 16 and 17 present results for the multivariate lognormal distribution, $Y = \exp(X)$, where X is a multivariate normally distributed variable with zero expectation vector and covariances as given in (10).

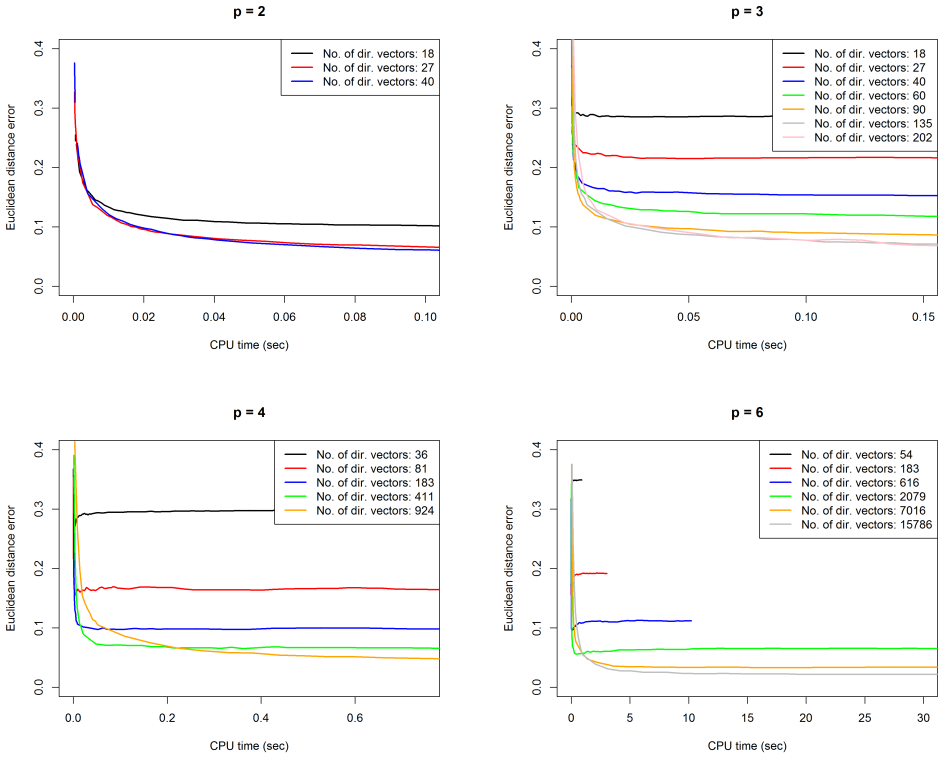

Figure 17: Multivariate lognormal distribution case: Each panel shows Euclidean distance error, Equation (8), as a function of CPU time for different number of directional vectors.

S.3. Synthetic Experiments - Dynamic Data Streams - Additional Results

Recall the systematic dynamic experiments leading to Tables 5 and 6. Figures 18 and 19 show tracking error for every choice of the tuning parameter λ (and $\gamma = \lambda$). For a period of $T = 10^3$, an optimal value of λ is about 0.01. For a period of $T = 10^4$, an optimal value of λ is about 0.025. Generally, the performance of the algorithm is fairly robust with respect to the value of λ. Further, Hammer et al. (2021) documents that the performance of the ShiftQ algorithm is highly robust on the choice of the tuning parameter γ.

Recall the systematic dynamic experiments with equidistantly distributed directional vectors leading to Table 7. Figure 20 shows tracking error for every choice of the tuning parameter λ (and $\gamma = \lambda$).

S.4. Real-life Data Example - Twitter Event Detection

A lot of research has been devoted to resort to Twitter messages (tweets) to detect events in the real world, see Atefeh and Khreich (2015) and Hasan et al. (2017) for recent surveys.

On July 22 2011 Norway was hit by a horrific terrorist attack initiated by a bomb going of in Oslo at 3:25 p.m local time. The upper panel in Figure 21 shows the number of new tweets and retweets posted every minute by Norwegian twitter users in the hours before and after the attack. The bottom panel shows the portions of tweets being retweets. As expected, the number of posted tweets increased rapidly after the attack, but an interesting observation is that the number of posted retweets increased *even more rapidly* than the number of new tweets. Before the attack the portion of posted tweets being retweets were less than 10% and jumped to about 25% after the attack. A popular approach to detect and characterize real-world events is

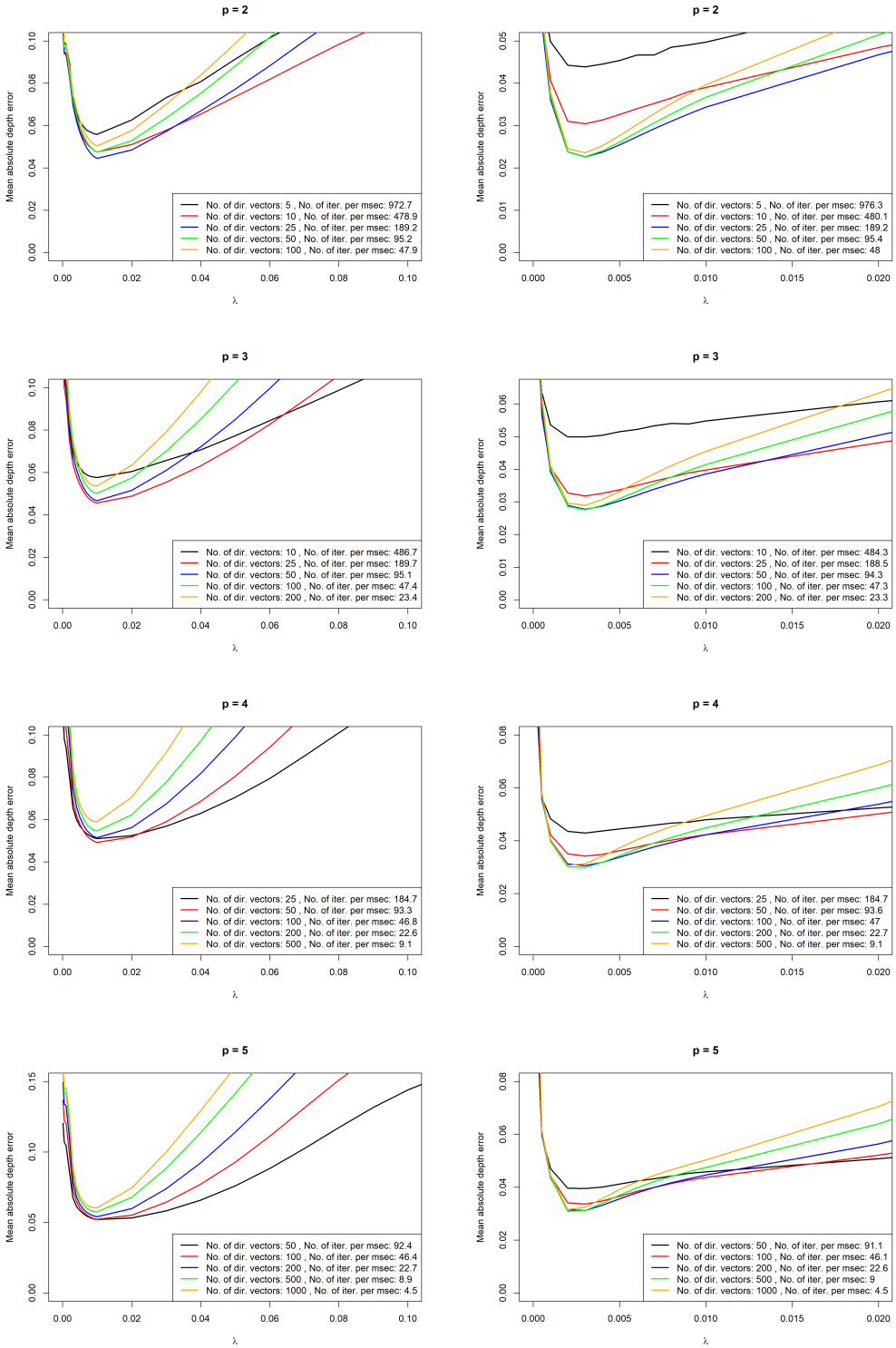

Figure 18: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12). Tracking error is measured using MADE. The left and right column show results for $T = 10^3$ $T = 10^4$, respectively. Rows from top to bottom represent dimensions $p = 2$, 3, 4 and 5, respectively. The number of iterations per msec, refers to how many times per millisecond we can update the estimate of one α-depth

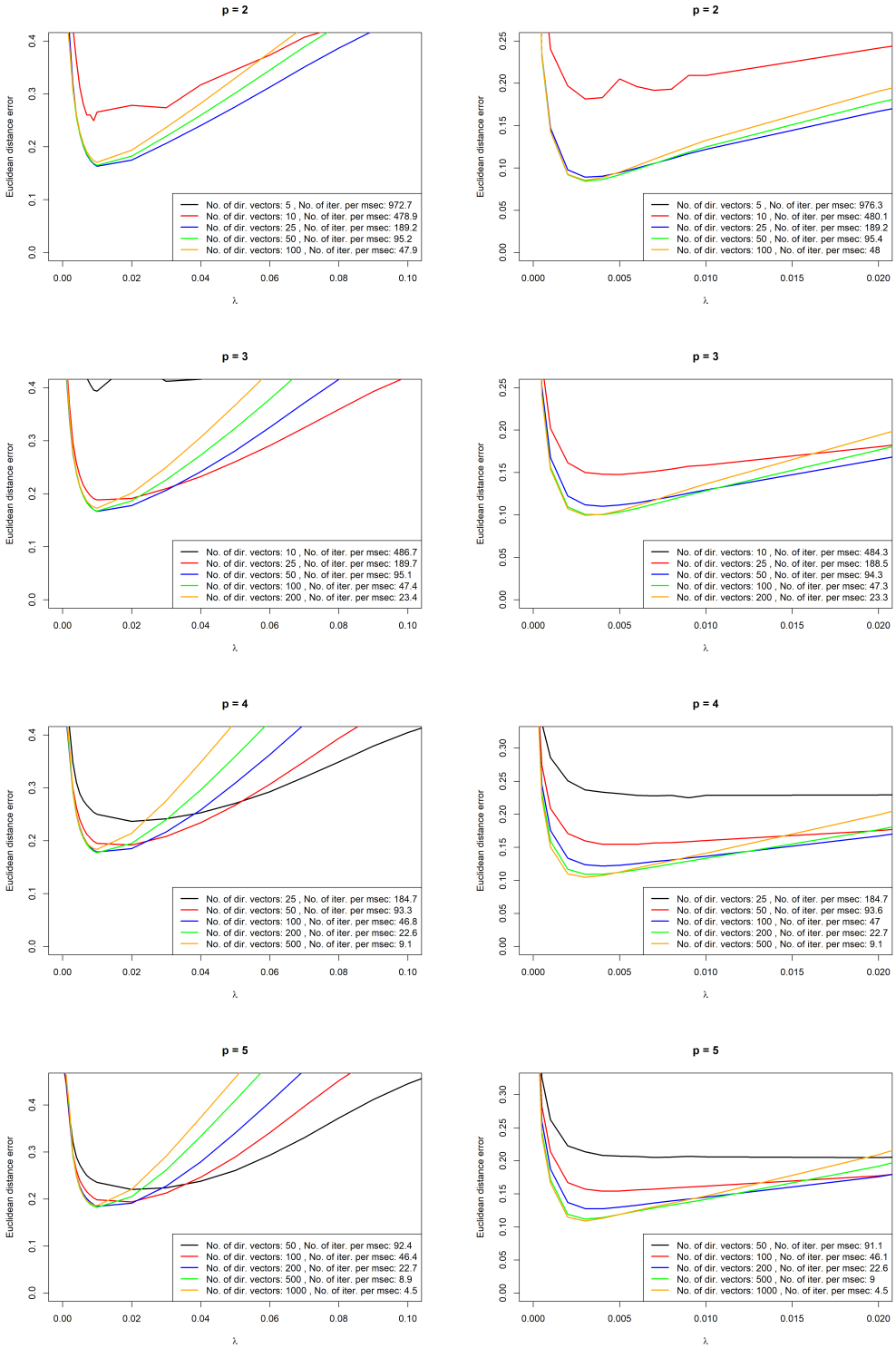

Figure 19: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12). Tracking error is measured using ED. The left and right column show results for $T = 10^3$ $T = 10^4$, respectively. Rows from top to bottom represent dimensions $p = 2$, 3, 4 and 5, respectively. The number of iterations per msec, refers to how many times per millisecond we can update the estimate of one α-depth region.

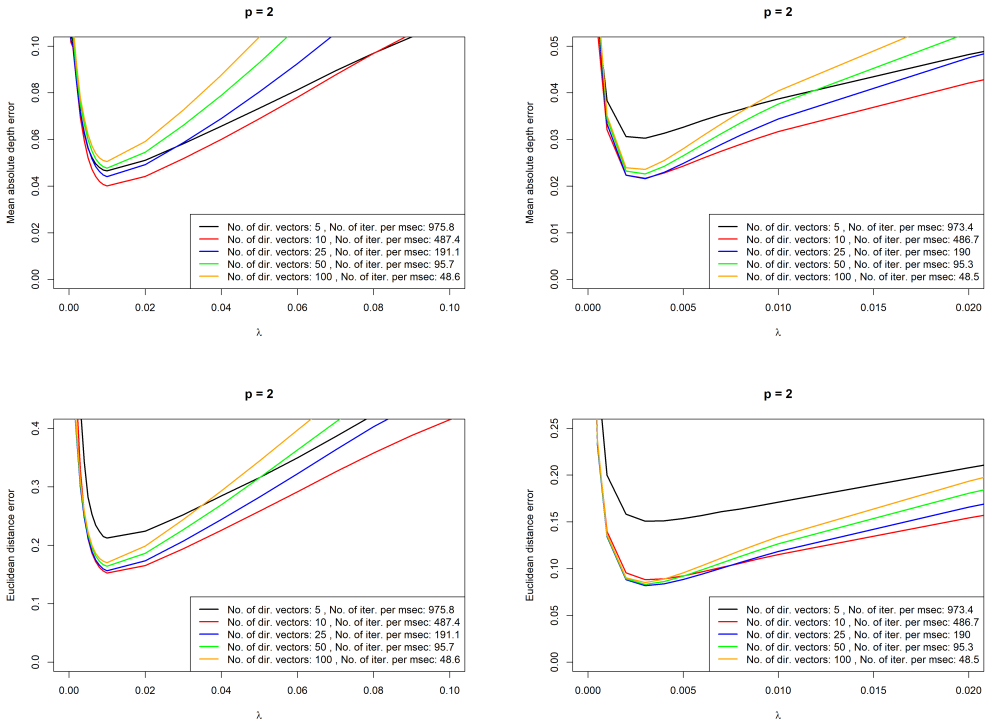

Figure 20: Tracking of α-depth regions for $\alpha = 0.05$, 0.2 and 0.4 for the distribution characterized by (11) and (12) using fairly equidistant directional vectors. The upper and lower panels show MADE and ED tracking errors, respectively. The left and right panels show results for $T = 10^3$ and 10^4, respectively. Dimension is $p = 2$.

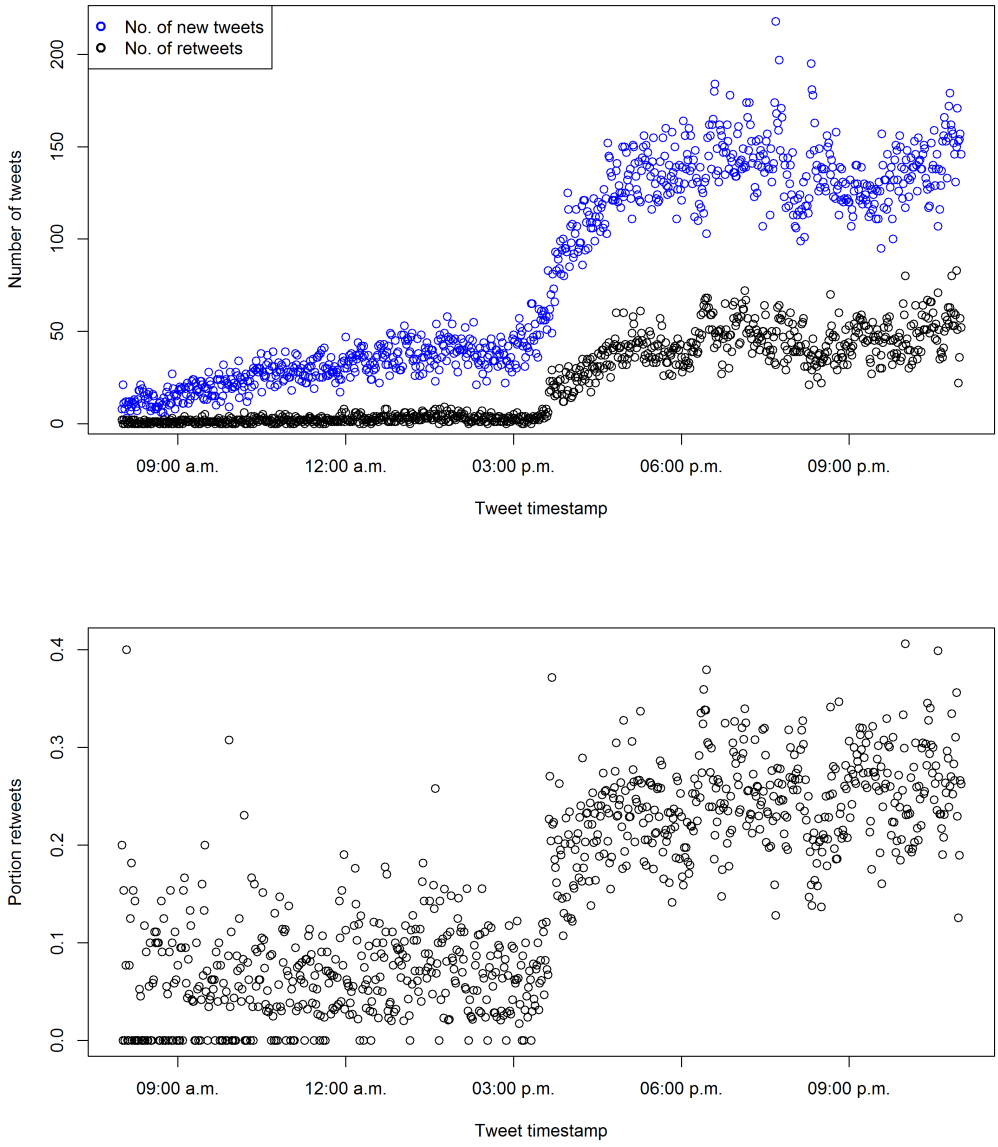

Figure 21: Upper panel: Number of new tweets (blue dots) and retweets (black dots) posted by Norwegian twitter user every minute on July 22 2011 from 08:00 a.m to 11:00 p.m. Bottom panel: Portion of posted tweets being retweets.

to monitor the number of posted tweets (Alkhamees and Fasli, 2016). Figure 21 suggests a generalization of this by studying the *simultaneous* distribution of the number of new tweets and retweets posted.

If the data stream distribution suddenly changes, the α-depth contour tracking algorithm naturally will need some time to adjust to these changes. This can be used to develop efficient event detection methods. Let $X_{n1}, \ldots, X_{np}$ represent the possible outcomes from the p dimensional data stream distribution at time n. Define $\theta(X_n)$ as the medians of the marginal distributions, i.e. $\theta(X_n) = (Q(\alpha = 0.5, X_{n1}), \ldots, Q(\alpha = 0.5, X_{np}))$, representing the center of the data stream distribution. Further, let $c(\alpha, w)$ define the intersection between the ray going through $\theta(X_n)$ and some point w and $D(\alpha)$. Define the *Tukey depth distance* as the Euclidean distance between $\theta(X_n)$ and w relative to the distance between $\theta(X_n)$ and $D(\alpha)$

$$d(w, X_n, \alpha) = \frac{\|w - \theta(X_n)\|_2}{\|c(\alpha, w) - \theta(X_n)\|_2} \tag{13}$$

The distance measure is similar to what Hubert et al. (2017) defined as *bag distance* and account for both the center and dispersion of the multivariate distribution and is affine invariant. One may consider generalizations of the distance measure based on multiple α-depth regions, but is not explored in this example.

When an event occurs, we expect the portion of received samples being inside $D(\alpha)$ to change until the tracking procedure adjusts to these changes. Consequently, we expect the distances $d(w, X_n, \alpha)$ to change rapidly until the tracking procedure adjusts.

Since $D(\alpha)$ characterizes the tweet retweet distribution in a very flexible way this event detection procedure can in practice detect *any* change in the

distribution.

We tracked the medians and the α-depth contour for $\alpha = 0.01$ of the distribution. $n_u = 10$ directional quantiles were estimated using DUMIQE with $\lambda = 0.05$. For every received sample (number of new tweets and retweets), we computed the Tukey depth distance, $d(w, X_n, \alpha)$, using the current estimates of the medians and the α-depth contour. With $\alpha = 0.01$, we expect the distances to be less than one for almost all samples except if an event has happened.

Figure 22 shows the measured distance for every received sample, in the days before and after the terror attack. The tracking procedure tracks the daily changes in the distribution in the days before the attack very well and most of the observed Tukey depth distances were, as expected, below one. After the terror attack, the distances instantly increased rapidly.

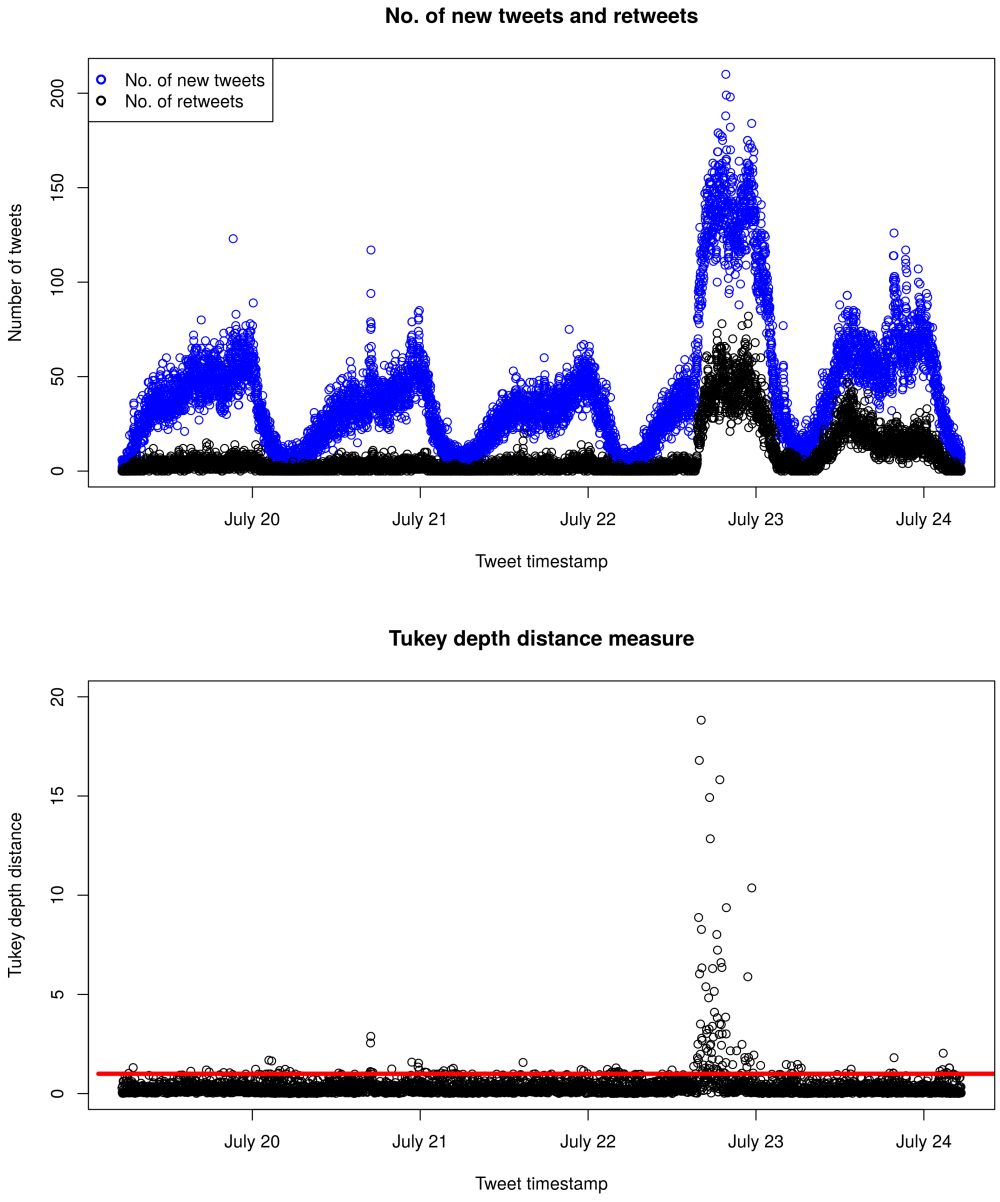

Figure 22: Upper panel: Number of new tweets (blue dots) and retweets (black dots) posted by Norwegian twitter user every minute in the days before and after the terror attack. Bottom panel: Tukey depth distance (13) of every received sample. The red line shows depth distance equal to one.