Journal of Statistical Software

May 2015, Volume 65, Issue 2. *http://www.jstatsoft.org/*

CompPD: A MATLAB Package for Computing Projection Depth

Xiaohui Liu
Jiangxi University of Finance and Economics

Yijun Zuo
Michigan State University

Abstract

Since the seminal work of Tukey (1975), depth functions have proved extremely useful in robust data analysis and inference for multivariate data. Many notions of depth have been developed in the last decades. Among others, projection depth appears to be very favorable. It turns out that (Zuo 2003; Zuo, Cui, and He 2004; Zuo 2006), with appropriate choices of univariate location and scale estimators, the projection depth induced estimators usually possess very high breakdown point robustness and finite sample relative efficiency. However, the computation of the projection depth seems hopeless and intimidating if not impossible. This hinders the further inference procedures development in practice. Sporadically algorithms exist in individual papers, though an unified computation package for projection depth has not been documented. To fill the gap, a MATLAB package entitled **CompPD** is presented in this paper, which is in fact an implementation of the latest developments (Liu, Zuo, and Wang 2013; Liu and Zuo 2014). Illustrative examples are also provided to guide readers through step-by-step usage of package **CompPD** to demonstrate its utility.

Keywords: projection depth, adjusted projection depth, projection depth regions, projection depth median, Stahel-Donoho estimators, MATLAB.

1. Introduction

In statistics, order statistics play very important roles in defining many desirable robust methods, which can serve as powerful alternatives to their traditional counterparts when outliers are present. Nevertheless, such methods depend heavily on the ordering of data. For univariate observations, the order principle is clear, and can naturally arise from the intrinsic order on the real line R^1. However, a tool that can provide a desirable ordering for multivariate observations is not straightforward available in higher dimensions R^p $(p \geq 2)$.

To serve this purpose, Tukey (1975) heuristically introduced a new depth notion, i.e., halfspace

depth. In the literature, this depth notion is also commonly referred to as "Tukey depth" in order to reflect Tukey's seminal work in this field. The most outstanding property of this depth notion is that it can provide the observations with a new ordering way. That is, unlike the traditional case, observations would be sorted center-outwardly according to this new principle. The deeper the point relative to the data cloud, the higher the depth value it has. Such an ordering has proved extremely powerful in extending univariate tools of signs and ranks, order statistics, quantiles, and outlyingness measures to the multivariate setting in a unified way (Serfling 2006).

In addition to the Tukey depth, there exist other depth notions in the literature. Among others, the most prominent and prevailing ones are simplicial depth (Liu 1990), projection depth (Liu 1992; Zuo 2003) and zonoid depth (Koshevoy and Mosler 1997) in the location setting, and regression depth (Rousseeuw and Hubert 1999) in the regression setting. Ideally, a desirable notion of depth should satisfy four key properties as stated in Zuo and Serfling (2000), namely, affine equivariance, maximality at center, monotonicity relative to the deepest point and vanishing at infinity. Many of the early results were summarized in Zuo and Serfling (2000), and the updated results for depths can be found in Liu, Serfling, and Souvaine (2006).

Depth-induced statistics (or procedures) usually inherit the desirable properties of their univariate counterparts. That is, they are affine equivariant, and possess very high breakdown robustness. The exception of the aforementioned depth notions is the zonoid depth, which centers at the regular mean and is consequently non-robust (Koshevoy and Mosler 1997). As a result, depth-induced statistics have been paid increasing attentions in the practice of inducing many favorable methods when outliers are present. For example, Ghosh and Chaudhuri (2005) proposed a depth-based classification technique for unequal prior problems. Yeh and Singh (1997) investigated a nonparametric method for constructing bootstrap confidence regions based on the depth-induced ordering. Chenouri and Small (2012) developed several nonparametric tests for testing the equality of two or more multivariate populations relying on statistical depth functions such as halfspace depth.

However, the price to be paid is that the computation of most depth-induced procedures is usually very challenging. Many authors have devoted themselves to overcoming this problem in the last decades. To name but a few, Rousseeuw and Ruts (1996) proposed two efficient algorithms for exactly computing bivariate halfspace depth and simplicial depth, respectively. Rousseeuw and Struyf (1998) studied the exact and approximate computation of halfspace depth in higher dimensions. Hallin, Paindaveine, and Šiman (2010) and Paindaveine and Šiman (2012) considered the computing issue of halfspace regions (or contours) in spaces with dimension $p \geq 2$ by using parametric linear programming. Dyckerhoff, Mosler, and Koshevoy (1996) and Mosler, Lange, and Bazovkin (2009) (see also Bazovkin and Mosler (2012)) investigated the efficient geometric algorithms for computing the zonoid depth value of a single point and the depth regions in any dimension, respectively. Publicly accessible packages on computing halfspace depth and zonoid depth are now available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=depth` (Genest, Masse, and Plante 2012) and `http://www.jstatsoft.org/v47/i13/` (Bazovkin and Mosler 2012), respectively.

In this paper, we are interested to address the computing issue of projection depth by offering a new MATLAB (The MathWorks Inc. 2009) package **CompPD**. Our motivation lies in the following fact. As a major depth notion, projection depth enjoys many desirable properties. It has continuous depth regions and does not vanish even outside the convex hull of the data,

which can bring great convenience to practical applications, such as classification (Ghosh and Chaudhuri 2005) and outlier identification (Dang and Serfling 2010). Furthermore, under mild conditions, the projection depth median is unique. This uniqueness property seldom holds true for many other depth notions such as halfspace depth and simplicial depth (Zuo 2013). With appropriate choices of univariate location and scale estimators, the projection depth induced estimators usually possess very high breakdown point robustness and finite sample relative efficiency (Zuo 2003; Zuo *et al.* 2004; Zuo 2006). However, application packages which can be used to compute the projection depth and its related estimators are currently lacking. The only available package is **ExPD2D** developed by Zuo and Ye (2009) in R. Unfortunately, **ExPD2D** is designed primitively for bivariate data as an implementation of Zuo and Lai (2011) and the code is inefficient and does not take account of the x-free property, namely, that the computation of optimal direction vectors is independent of the point x for which the depth value is being computed, of projection depth (Liu *et al.* 2013). In this paper, we fill the gap by presenting a MATLAB package **CompPD**, which is in fact an implementation of the latest developments (Liu *et al.* 2013; Liu and Zuo 2014). Codes of fast approximate algorithms are also provided for higher dimensions. Projection depth and all of its induced estimators can be conveniently computed by utilizing the current package.

The rest of this paper is organized as follows. In Section 2, we give necessary backgrounds on the definitions of projection depth, adjusted projection depth and their associated estimators. In Section 3, we describe the major functions contained in the MATLAB package. In Section 4, we provide some illustrative examples to demonstrate the utility of the package through step-by-step usage. In Section 5, we end the paper with a brief summary.

2. Preliminary

In this section, we will describe the definitions of the projection depth and adjusted projection depth and their associated estimators. Since the discussion focuses mainly on the computing issue, we only provide their sample versions in the following.

In one dimension, to measure the outlyingness of a point z relative to a given data set $\mathcal{Z}^n = \{Z_1, Z_2, \ldots, Z_n\}$, a robust method is to use the following outlyingness function

$$o_1(z, \mathcal{Z}^n) = \frac{|z - \mathrm{Med}(\mathcal{Z}^n)|}{\mathrm{MAD}(\mathcal{Z}^n)},$$

where $\mathrm{Med}(\mathcal{Z}^n) = (Z_{(\kappa_1)} + Z_{(\kappa_2)})/2$ and $\mathrm{MAD}(\mathcal{Z}^n) = \mathrm{Med}\{|Z_i - \mathrm{Med}(\mathcal{Z}^n)|, i = 1, 2, \ldots, n\}$ denote the median and the median absolute deviation (MAD) from the median of $\mathcal{Z}^n$, respectively. Here $\kappa_1 = \lfloor (n+1)/2 \rfloor$, $\kappa_2 = \lfloor (n+2)/2 \rfloor$, and $Z_{(1)} \leq Z_{(2)} \leq \ldots \leq Z_{(n)}$ denote the order statistics based on $\mathcal{Z}^n$ with $\lfloor \cdot \rfloor$ being the floor function.

By utilizing the projection pursuit technique and taking the supremum of all the univariate outlyingness values of the projections of x onto lines, one can obtain the following p-dimensional outlyingness function (Stahel 1981; Donoho 1982)

$$\mathrm{O}(x, \mathcal{X}^n) = \sup_{u \in \mathcal{S}^{p-1}} o_1(u^\top x, u^\top \mathcal{X}^n),$$

where $\mathcal{S}^{p-1} = \{v \in \mathcal{R}^p : \|v\| = 1\}$ with $\|\cdot\|$ standing for the Euclidean norm, $u^\top \mathcal{X}^n = \{u^\top X_1, u^\top X_2, \ldots, u^\top X_n\}$. Based on $\mathrm{O}(x, \mathcal{X}^n)$, the projection depth value of a point x with

respect to $\mathcal{X}^n$ in $\mathcal{R}^p$ then can be defined as (Liu 1992; Zuo 2003)

$$\mathrm{PD}(x,\, \mathcal{X}^n) = (1 + \mathrm{O}(x,\, \mathcal{X}^n))^{-1}\,.$$

Obviously, $0 \le \mathrm{PD}(x,\, \mathcal{X}^n) \le 1$. If no confusion arises, in the sequel we denote $p(x) = \mathrm{PD}(x,\, \mathcal{X}^n)$.

As a major depth notion, projection depth can induce affine equivariant, nested and convex depth regions (Zuo 2003):

$$\mathrm{PR}(\alpha, \mathcal{X}^n) = \{x \in R^p : p(x) \ge \alpha\},$$

where $0 \le \alpha \le \alpha^* = \sup_{x \in R^p} p(x)$. In practical applications, depth regions $\mathrm{PR}(\alpha, \mathcal{X}^n)$ play similar roles as the quantiles in univariate statistics. As a special case, the innermost region

$$\mathrm{PR}(\alpha^*, \mathcal{X}^n) = \{x \in R^p : p(x) \ge \alpha^*\}$$

contains only a single point θ under some mild conditions (Zuo 2013). In the literature, θ is usually referred to as projection depth median $\mathrm{PM}(\mathcal{X}^n)$.

Furthermore, by choosing proper weight functions such as

$$\omega_k(r) = \frac{\exp(-K(1-(r/C)^k)^{2k}) - \exp(-K)}{1-\exp(-K)} I(r < C) + I(r \ge C), \quad k = 1, 2,$$

one can define highly robust multivariate estimators, e.g., the Stahel-Donoho estimators (Zuo *et al.* 2004)

$$\begin{aligned}
\widehat{\mu}_n &:= \mathrm{PWM}(\mathcal{X}^n) = \sum_{i=1}^n \omega_{1,i} X_i \Big/ \sum_{i=1}^n \omega_{1,i}\,, \\
\widehat{\Sigma}_n &:= \mathrm{PWS}(\mathcal{X}^n) = \sum_{i=1}^n \omega_{2,i} (X_i - \widehat{\mu}_n)(X_i - \widehat{\mu}_n)^\top \Big/ \sum_{i=1}^n \omega_{2,i}\,,
\end{aligned}$$

and the projection depth trimmed mean (Zuo 2006)

$$\mathrm{PTM}(\mathcal{X}^n) = \sum_{i:p(X_i)\ge\alpha} \omega_{1,i} X_i \Big/ \sum_{i:p(X_i)\ge\alpha} \omega_{1,i}\,,$$

where $\omega_{k,i} = \omega_k(p(X_i))$.

Note that for any direction u ($\in \mathcal{S}^{p-1}$), the univariate outlyingness function $o_1(u^\top x,\, u^\top \mathcal{X}^n)$ is always symmetrical about $\mathrm{Med}(u^\top \mathcal{X}^n)$ with respect to (w.r.t.) x. As a result, the projection depth may fail to capture the real shape of the data cloud when skewed data are present (Hubert and Van der Veeken 2010). This motivates us to consider an adjusted version of the aforementioned projection depth as follows:

$$\mathrm{APD}(x,\, \mathcal{X}^n) = \left(1 + \sup_{u \in \mathcal{S}^{p-1}} o_1^*(u^\top x,\, u^\top \mathcal{X}^n)\right)^{-1},$$

where

$$o_1^*(u^\top x,\, u^\top \mathcal{X}^n) = \begin{cases} \dfrac{u^\top x - \mathrm{Med}(u^\top \mathcal{X}^n)}{\mathrm{Q}_1(u^\top \mathcal{X}^n) - \mathrm{Med}(u^\top \mathcal{X}^n)}, & \text{if } u^\top x < \mathrm{Med}(u^\top \mathcal{X}^n) \\ \dfrac{u^\top x - \mathrm{Med}(u^\top \mathcal{X}^n)}{\mathrm{Q}_3(u^\top \mathcal{X}^n) - \mathrm{Med}(u^\top \mathcal{X}^n)}, & \text{if } u^\top x \geq \mathrm{Med}(u^\top \mathcal{X}^n) \end{cases},$$

where $\mathrm{Q}_1(\mathcal{Z}^n) = Z_{(q_1)}$ and $\mathrm{Q}_3(\mathcal{Z}^n) = Z_{(q_2)}$, $q_1 = \lceil n/4 \rceil$, $q_2 = n - q_1 + 1$, and $\lceil \cdot \rceil$ denotes the ceiling function. For convenience, hereafter we denote $a(x) = \mathrm{APD}(x,\, \mathcal{X}^n)$. Without loss of generality, we call it adjusted projection depth. Using similar proofs to Liu *et al.* (2013), one can show that this adjusted projection depth can also be exactly computed and shares the x-property of the projection depth.

Based on the adjusted projection depth, one can define the adjusted projection depth regions $\mathrm{APR}(\alpha,\, \mathcal{X}^n)$ and median $\mathrm{APM}(\mathcal{X}^n)$ as

$$\begin{aligned} \mathrm{APR}(\alpha,\, \mathcal{X}^n) &= \{x \in R^p : a(x) \geq \alpha\}, \\ \mathrm{APM}(\mathcal{X}^n) &= \arg\sup_{x \in R^p} a(x) \end{aligned}$$

by following a similar fashion to that of projection depth.

3. The MATLAB package CompPD

This paper includes a MATLAB implementation of the latest developments (Zuo and Lai 2011; Liu *et al.* 2013; Liu and Zuo 2014) concerning the computation of projection depth and adjusted projection depth and their associated estimators as described in the earlier paragraph. Let us review the included code entitled **CompPD**.

The package consists of thirty files, including two data sets, one script, and twenty-seven functions. All of these files must be in MATLAB's current directory or on the search path to use the package **CompPD**. Tables 1 and 2 present a brief list of the main files included in the package.

The general usage of the code is as follows. The function `PertX` is used to perform data preprocessing when the input data are from a real application, and may be not in general position. Handled by `PertX`, `X` would be perturbed by adding some random noises of a very small magnitude $e\times$ `TINY0`, and then centralized at its mean vector, where e is uniformly distributed over $(-1, 1)$. The default value of `TINY0` is 10^{-6}. The user also can choose another value through calling `PertX(X, TINY0)` with a user-defined `TINY0`.

Note that both of projection depth and adjusted projection depth possess the x-free property (Liu *et al.* 2013). We construct special functions to compute the optimal direction vectors, based on which one can conveniently obtain the exact depth value(s) and the related estimators. Hereafter we refer to the optimal direction vectors as ODVs.

For the case of projection depth, this task can be fulfilled by calling `ExVecPD2D(X)` and `ExVecPDHD(X)`. `ExVecPD2D` finds the ODVs counter-clockwise and is constructed only for bivariate data. `ExVecPDHD` evaluates the ODVs in higher dimensions by utilizing the breadth-first search algorithm, and is in fact an implementation of the algorithm developed in Liu and Zuo (2014). The input argument `X` should be an $n \times p$ matrix, where n denotes the sample size. Both of them return a `struct` with the form

File name	Type	Typical usage	Description
`BostData.mat`	Dataset	`load BostData`	Part of the Boston housing data (65×3 matrix).
`quakes.mat`	Dataset	`load quakes`	Locations of 1000 seismic events of MB > 4.0 occurred in a cube near Fiji since 1964 (1000×2 matrix).
`Examples.m`	Script	`edit examples`	Examples of this section.
`APC2D.m`	Function	`APC2D(X, vec, alpha)`	Compute the αth bivariate adjusted projection depth contour(s) of X based on `vec`.
`APC3D.m`	Function	`APC3D(X, vec, alpha)`	Compute the αth 3-dimensional adjusted projection depth contour(s) of X based on `vec`.
`APDVal.m`	Function	`apdv = APDVal(X, vec, x)`	Compute the adjusted projection depth value(s) of x w.r.t. a p-dimensional ($p \geq 2$) data set X based on `vec`.
`APM.m`	Function	`apm0 = APM(X, vec)`	Compute the adjusted projection depth median of X based on `vec`.
`AppVecAPD.m`	Function	`vec = AppVecAPD(X, m)`	Generate m direction vectors for approximating the p-dimensional ($p \geq 2$) adjusted projection depth.
`AppVecPD.m`	Function	`vec = AppVecPD(X, m)`	Generate m direction vectors for approximating the p-dimensional ($p \geq 2$) projection depth.
`DDplot.m`	Function	`DDplot(X, Y)`	Depth versus depth plot of X versus Y.
`DSplot.m`	Function	`DSplot(X, pdv)`	Depth-size scatter plot of a bivariate or trivariate sample X.
`ExVecAPD2D.m`	Function	`vec = ExVecAPD2D(X)`	Compute the optimal direction vectors of the adjusted projection depth w.r.t. a bivariate sample X.
`ExVecAPDHD.m`	Function	`vec = ExVecAPDHD(X)`	Compute the optimal direction vectors of the adjusted projection depth w.r.t. a 3-dimensional sample X.
`ExVecPD2D.m`	Function	`vec = ExVecPD2D(X)`	Compute the optimal direction vectors of the projection depth w.r.t. a bivariate sample X.
`ExVecPDHD.m`	Function	`vec = ExVecPDHD(X)`	Compute the optimal direction vectors of the projection depth w.r.t. a 3-dimensional sample X.

Table 1: List of the main files included in **CompPD**. Part I.

File name	Type	Typical usage	Description
`outlierplot.m`	Function	`outlierplot(X, pdv, alpha)`	Scatter plot matrix of the outliers of X.
`OutVal.m`	Function	`outv = OutVal(X, vec, x)`	Compute the outlyingness value(s) of x w.r.t. a p-dimensional ($p \geq 2$) data set X based on `vec`.
`PC2D.m`	Function	`PC2D(X, vec, alpha)`	Compute the αth bivariate projection depth contour(s) of X based on `vec`.
`PC3D.m`	Function	`PC3D(X, vec, alpha)`	Compute the αth 3-dimensional projection depth contour(s) of X based on `vec`.
`PDVal.m`	Function	`pdv = PDVal(X, vec, x)`	Compute the projection depth value(s) of x w.r.t. a p-dimensional ($p \geq 2$) data set X based on `vec`.
`PertX.m`	Function	`Xp = PertX(X)`	Perturb and centralize X.
`PM.m`	Function	`pm0 = PM(X, vec)`	Compute the projection depth median of X based on `vec`.
`PTM.m`	Function	`ptm0 = PTM(X, pdv, alpha)`	Compute the αth projection depth trimmed mean of X based on pdv.
`PWM.m`	Function	`pwm0 = PWM(X, pdv)`	Compute the depth weighted mean of X based on pdv.
`PWS.m`	Function	`pws0 = PWS(X, pdv)`	Compute the depth weighted scatter matrix of X based on pdv.
`RndVecAPD.m`	Function	`vec = RndVecAPD(X, m)`	Generate m random direction vectors for computing the p-dimensional ($p \geq 2$) random adjusted projection depth by using the uniform distribution on the sphere.
`RndVecPD.m`	Function	`vec = RndVecPD(X, m)`	Generate m random direction vectors for computing the p-dimensional ($p \geq 2$) random projection depth by using the uniform distribution on the sphere.

Table 2: List of the main files included in **CompPD**. Part II.

```
vecpd = struct('u', [], 'MedVal', [], 'MADVal', [], 'NumDirVecU', [])
```

where the field `u` contains m computed ODVs and is a $p \times m$ matrix, `MedVal` and `MADVal` contain the median and MAD values of the projections onto `u`, respectively, and `NumDirVecU` contains the number of ODVs. For the case of adjusted projection depth, similar functions are `ExVecAPD2D`, `ExVecAPDHD`, both of which return a `struct` with the form

```
vecapd = struct('u', [], 'MedV', [], 'UpQV', [])
```

It is worth mentioning that `ExVecPDHD` and `ExVecAPDHD` are ideally limited to work in spaces with dimension $3 \leq p \leq 8$, because they need to call a built-in function `convhulln` (Barber, Dobkinm, and Huhdanpaa 1996), which has such a dimensional limitation. However, *we do not recommend using them when* $p > 3$ since they consume quite considerable CPU times in such spaces. Both of them search the whole p-dimensional space quadrant-by-quadrant. In each quadrant, the default maximum number of iterations is 35000. Therefore, when n and/or p increase, they may miss finding some ODVs. A similar trick is used in `compContourM1`, which aims at computing the high dimensional halfspace depth contours (Paindaveine and Šiman 2012). `compContourM1` has been developed by Prof. Davy Paindaveine and his coauthors, and can be downloaded from `http://homepages.ulb.ac.be/~dpaindav/`.

In addition, we also construct a function `AppVecPD` to generate m direction vectors for approximating the projection depth and its associated estimators. Its typical usage is `vec = AppVecPD(X, m)`. The resulting direction vectors are data-dependent and affine equivariant; see the code for more details. The function `RndVecPD` is used to generate a finite number of random direction vectors `vec` by using the uniform distribution on the sphere. `vec` has the same form as `vecpd`. `RndVecPD` is computationally simple, but cannot generate random unit vectors in an *affine equivariant* way. So this function is *not useful* in computing estimators such as the Stahel-Donoho estimator *if the affine equivariance is a big concern*. Similarly, for the case of adjusted projection depth, the interested user is advised to refer to `AppVecAPD` and `RndVecAPD`, which return a `struct` with the same form as `vecapd`.

The main functions `OutVal(X, vec, x)` and `PDVal(X, vec, x)` are used to evaluate the outlyingness and projection depth value(s) of `x` w.r.t. `X` based on `vec`, respectively. If `vec` is returned from functions such as `ExVecPD2D`, the resulting outlyingness/projection depth value(s) is (are) exact. The function `[pm0, pdpm0] = PM(X, vec, IsLP)` is constructed to compute the projection depth median based on `vec`, and returns the computed median `pm0` and its corresponding depth value `pdpm0`. `IsLP` is an indicator, and `IsLP = true` means calling the function `linprog`. It is worth mentioning that, when the number of direction vectors contained in `vec` are large, `linprog` called in `PM` consumes quite considerable CPU times. In such a case, we recommend the user to employ only the following code instead

```
[u, indx] = unique(DirVecU.u', 'rows');
MedV = DirVecU.MedVal(indx);
MADV = DirVecU.MADVal(indx);

p = size(X, 1);
AMat = u ./ (MADV * ones(1, p));
bvec = MedV ./ MADV;
tmpv = 1;
```

```
initx0 = median(X, 2);
while tmpv > 1e-12
  outfun = @(x0, AMat, bvec) max(AMat * x0 - bvec);
  [pm0, outv] = fminsearch(@(x0)outfun(x0, AMat, bvec), initx0);
  tmpv = norm(initx0 - pm0);
  initx0 = pm0;
end
pdvpm = 1 / (1 + outv);
```

by calling `PM(X, vec)` or `PM(X, vec, false)`, because one can show that the projection depth median is a global optimum and no local optimum exists based on a finite number of direction vectors `vec` (Liu *et al.* 2013). The function `fminsearch` works well if there is no local optimum in the objective function; see its help file for details. Remarkably, `fminsearch` is much faster than `linprog`, although the result of `fminsearch` has slightly lower accuracy than that of `linprog`.

To learn more details about the files included in **CompPD**, the interested user is advised to consult the help text and comments within each files, including optional input and output arguments that are not discussed here. Code for running the examples can be found in the script file `Examples`.

4. Using CompPD

In this section, we will show line-by-line the sequence of commands by utilizing some illustrative examples.

4.1. Bivariate example: `quakes` data

Our first example is a R data set from package **datasets** (R Core Team 2015). The data set gives the locations of 1000 seismic events of $MB > 4.0$ occurred in a cube near Fiji since 1964. The original data frame includes 5 variables. In the following, we pick up only the first two variables, namely, the latitude and longitude of events, in the discussion.

The first step in the analysis is to load the data and perform some preliminary operations.

```
load quakes
X = quakes(1:75, :);
[n, p] = size(X);
figure(1);
plot(X(:, 1), X(:, 2), 'ko', 'MarkerSize', 4)
xlabel('X_{1}'); ylabel('X_{2}');
box on;
```

The resulting scatter plot is reported in Figure 1.

Next, the function `ExVecPD2D(X)` will be called to find the ODVs. Doing so illustrates how to compute the projection depth and the associated estimators.

```
ExVec1 = ExVecPD2D(X);
x = X(1:5, :);
```

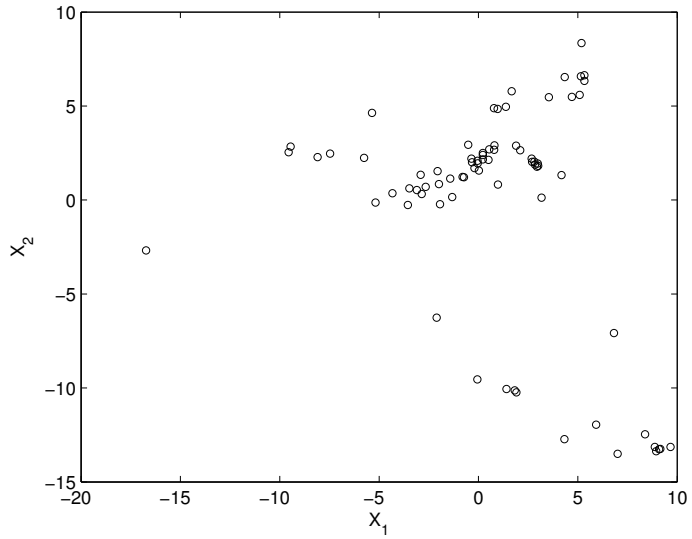


Figure 1: Scatter plot of the first 75 data points contained in `quakes.mat`.

```
expdvx = PDVal(X, ExVec1, x);
NumU = 10000;
RndVec1 = RndVecPD(X, NumU);
rndpdvx = PDVal(X, RndVec1, x);
AppVec1 = AppVecPD(X, NumU);
appdvx = PDVal(X, AppVec1, x);
```

Here `expdvx`, `RndVec1` and `AppVec1` denote the exact, random and approximate projection values of `x`, repetitively. The results are demonstrated by calling

```
num2str([expdvx, rndpdvx, appdvx, rndpdvx - expdvx, appdvx - expdvx], 6)
```

which leads to the following results

```
ans =
    0.603059     0.603064      0.60311 5.28183e-006 5.09181e-005
    0.711562     0.711635     0.711562 7.28634e-005 5.55112e-016
    0.180479     0.180578     0.180479 9.87916e-005 2.22045e-016
     0.43621     0.436371      0.43621  0.000160671 3.33067e-016
    0.530174     0.530178     0.530236 4.16612e-006 6.18999e-005
```

The results may be *slightly different* in each run due to the fact that random direction vectors are considered.

Based on the computed ODVs `ExVec1`, one also can obtain the projection depth induced estimators by calling the following code

```
expdv = PDVal(X, ExVec1);
pm0 = PM(X, ExVec1, true);
pwm0 = PWM(X, expdv);
```

```
ptm015 = PTM(X, expdv, 0.15);
[pm0, pwm0, ptm015]

pws = PWS(X, expdv)
covx = cov(X)
```

where `cov(X)` computes the ordinary covariance matrix of `X`. These codes yield that

```
ans =
    0.0512    0.0822   -0.0305
    1.6608    1.8505    2.2970

pws =
   11.0011    2.5561
    2.5561    5.6772

covx =
   22.4704  -10.3226
  -10.3226   35.4954
```

Since `ExVec1` is returned from `ExVecPD2D`, the results above are exact w.r.t. `X`. Furthermore, one can call the following code to compute approximately the associated estimators

```
appdv = PDVal(X, AppVec1);
appm0 = PM(X, AppVec1);
appwm0 = PWM(X, appdv);
apptm015 = PTM(X, appdv, 0.15);
[appm0, appwm0, apptm015]
appws = PWS(X, appdv)
```

to obtain the results as follows:

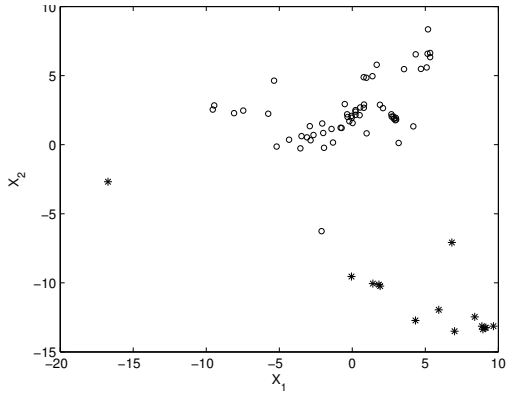
```
ans =
    0.0511    0.0846    0.0148
    1.6609    1.8524    2.3484

appws =
   11.0040    2.5694
    2.5694    5.6870
```

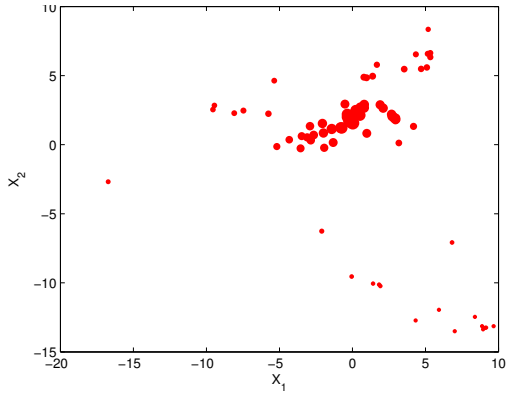
Based on the exact depth values of `X`, the functions

```
gamm0 = 0.20;
outlierplot(X, expdv, gamm0)
DSplot(X, expdv)
```

are used to plot the $\lfloor 0.2n \rfloor$-outlier scatter plot and projection depth-size scatter plot of data `X`; see Figures 2(a) and 2(b), respectively. Here the α-outliers are defined as $\lfloor n\alpha \rfloor$ data points having smallest projection depth values.



(a) $\lfloor 0.2n \rfloor$-outlier scatter plot, where the asterisked points stand for the outliers.



(b) Projection depth-size scatter plot, where the size of data points is proportional to their projection depth values. The larger the depth value, the bigger the plotted point.

Figure 2: $\lfloor 0.2n \rfloor$-outlier scatter plot and the projection depth-size scatter plot of the data.

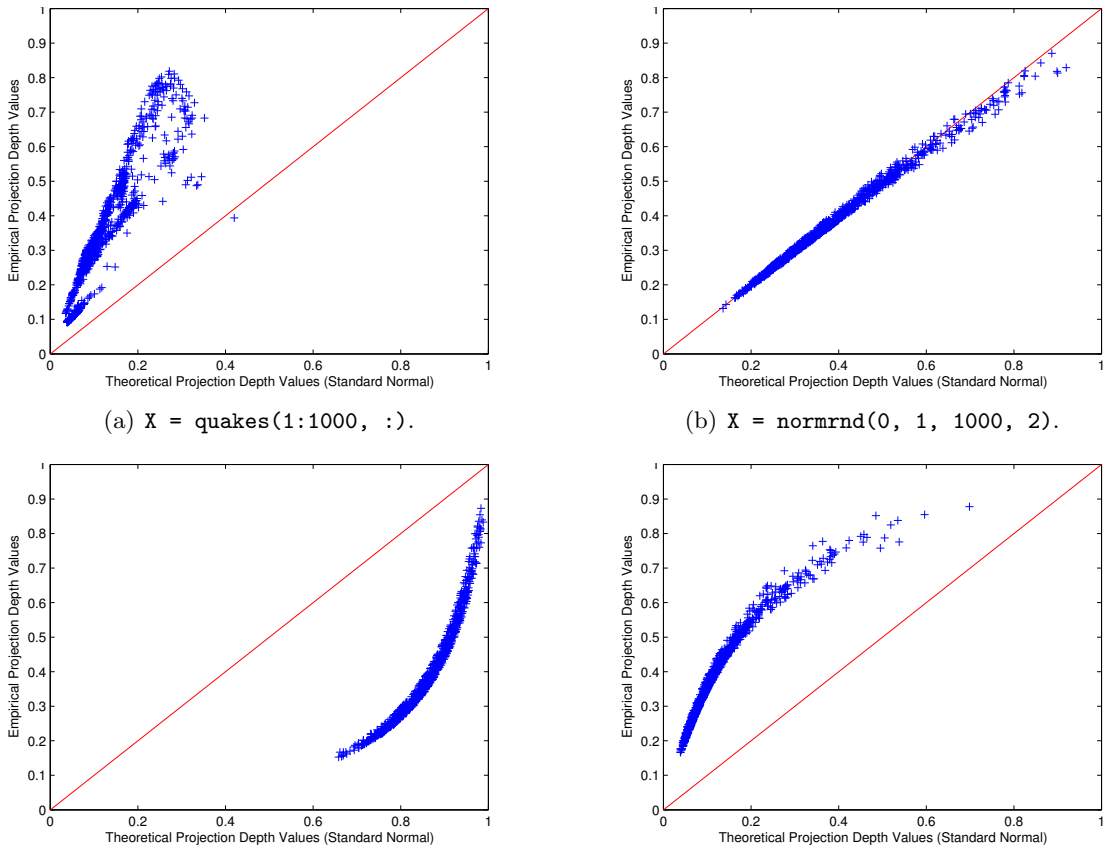
In univariate descriptive statistics, the QQ-plot (quantile versus quantile plot) is a very useful graphical method for comparing two probability distributions by plotting their quantiles against each other. In higher dimensions, a similar tool is the DD-plot (depth versus depth plot) introduced by Liu, Parelius, and Singh (1999). This package contains a function `DDplot` of this kind based on the projection depth.

As a special case of such applications, `DDplot` can be used to test for multinormality (with a unspecified mean vector and covariance matrix) by calling `DDplot(X)`. Usually, there are two natural ways to do that on the basis of the DD-plot. The first one is to use theoretical depth values associated with the multinormal distribution with mean and covariance parameter values that are matching the sample mean vector and covariance matrix of the data at hand. The second one is to compute the DD-plot on the basis of standardized data (where standardization, again, is based on the sample mean and sample covariance matrix). From affine-invariance, both procedures are equivalent. Nevertheless, the second one has the advantage that no new functions have to be introduced, and hence is utilized in `DDplot`. Provided in the sequel are some illustrations of its usage, and the results are shown in Figure 3.

```
DDplot(quakes)
DDplot(normrnd(0, 1, 1000, 2))
DDplot(normrnd(0, 0.1, 1000, 2))
DDplot(normrnd(0, 5, 1000, 2))
```

Ideally, if the observations are normally distributed, the points in the DD-plot would approximately lie on the line $y = x$; see Figure 3(b). In this sense, it is improper to conclude that `X = quakes` are generated from a normal distribution; see Figure 3(a).

In recent years, the DD-plot was mostly used in a classification context. In this two-sample setup, `DDplot(X, Y, Z)` reports a scatter plot with x-coordinates of each point being the depth of the corresponding data point `Z` within the observations of the first sample `X` and y-coordinates of each point being the depth of the corresponding data point `Z` within observations of the second sample `Y`. New observations to be classified, in the most standard case,



(a) `X = quakes(1:1000, :).`

(b) `X = normrnd(0, 1, 1000, 2).`

(c) `X = normrnd(0, 0.1, 1000, 2).`

(d) `X = normrnd(0, 5, 1000, 2).`

Figure 3: DD-plots of `X` versus the standard normal distribution.

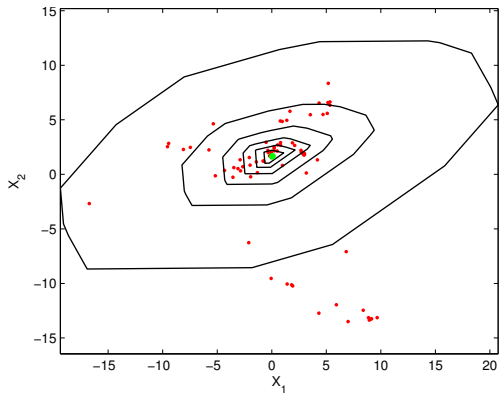
are then classified into population one or two according to their position with respect to the main bisector. See Li, Cuesta-Albertos, and Liu (2012) and references therein for more details about depth-based classifiers.

We move on to compute the projection depth regions. The function `PC2D` is constructed to serve this purpose. Several regions can be plotted simultaneously. The exact and approximate regions can be computed by calling

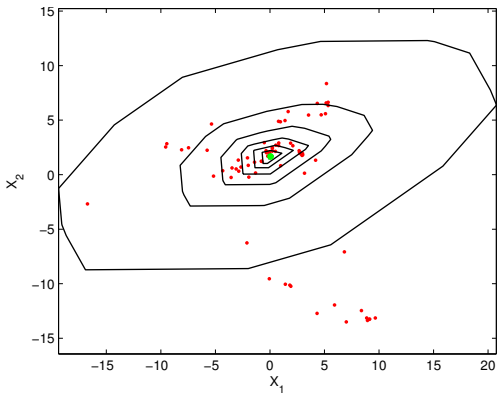
```
alpha0 = 0.1:0.1:1;
PC2D(X, ExVec1, alpha0, false)
PC2D(X, AppVec1, alpha0, false)
```

where `alpha0` denotes the projection depth values.

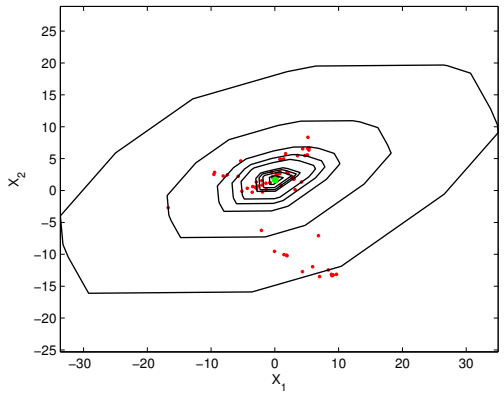
If the given `alpha0` $> \alpha^*$ (the maximum depth value), `PC2D` does nothing. The resulting exact and approximate depth regions are showed in Figures 4(a) and 4(b), respectively. Here depth regions are indexed by their "depth" α_0. Equivalently, they may be indexed by their probability content, i.e., in the sample case, by the proportion of data points *outside* the region. In practical applications, this alternative indexing may be very useful and can be fulfilled by calling the following code (see Figures 4(c)–4(d)).



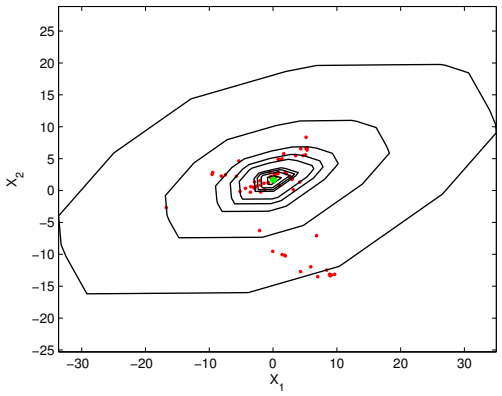
(a) Exact projection depth regions (in depth).



(b) Approximate projection depth regions (in depth).



(c) Exact projection depth regions (in probability).



(d) Approximate projection depth regions (in probability).

Figure 4: $\alpha_0 = 0.1, 0.2, \ldots, 0.7$ bivariate exact and approximate projection depth regions, where α_0 stand for the projection depth values in Figures 4(a)–4(b) and for the proportions of data points outside the region in Figures 4(c)–4(d).

```
PC2D(X, ExVec1, alpha0)
PC2D(X, AppVec1, alpha0)
```

where `alpha0` stands for the proportions of data points outside the region. Otherwise, call instead

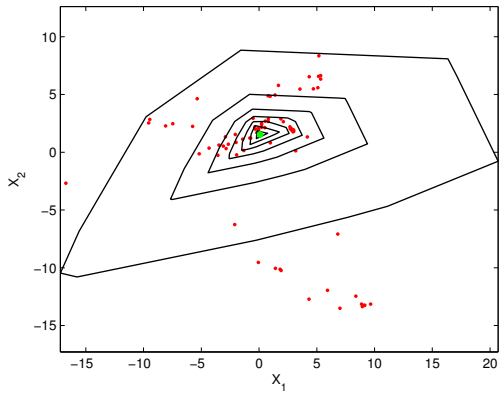
```
PC2D(X, ExVec1, alpha0, true)
PC2D(X, AppVec1, alpha0, true)
```

if `alpha0` are the depth values.

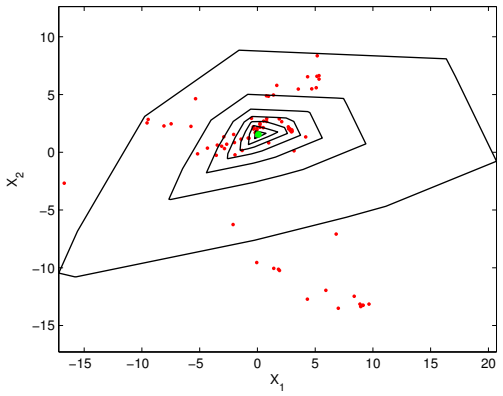
An another usage of `PC2D` is `VPMat = PC2D(X, ExVec1, alpha0, [], false)`, which returns only the vertices of the computed region(s), without any figure.

Similarly, the adjusted projection depth and its associated depth regions of `X` can be computed by calling

```
ExVec2 = ExVecAPD2D(X);
x = X(1:5, :);
```



(a) Exact adjusted projection depth regions.



(b) Approximate adjusted projection depthregions.

Figure 5: 0.1, 0.2, . . . , 0.7 bivariate exact and approximate adjusted projection depth regions.

```
exapdvx = APDVal(X, ExVec2, x);
NumU = 5000;
RndVec2 = RndVecAPD(X, NumU);
rndapdvx = APDVal(X, RndVec2, x);
AppVec2 = AppVecAPD(X, NumU);
appapdvx = APDVal(X, AppVec2, x);
num2str([exapdvx, rndapdvx, appapdvx, rndapdvx - exapdvx,...
  appapdvx - exapdvx], 6)

alpha0 = 0.1:0.1:1;
APC2D(X, ExVec2, alpha0, false)
APC2D(X, AppVec2, alpha0, false)
```

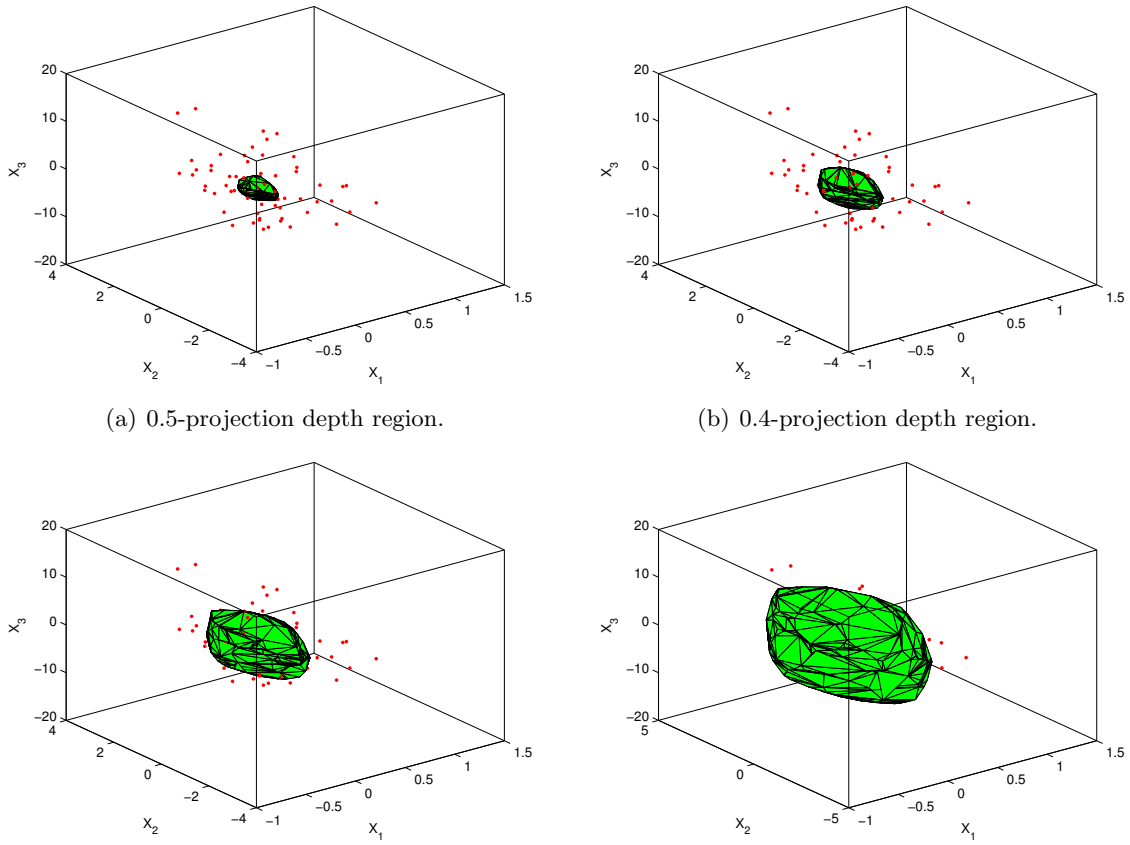
which yield

```
ans =
    0.604664     0.604702     0.604808  3.7736e-005  0.000143296
    0.758284     0.758666     0.758284   0.00038219 2.22045e-016
    0.121588       0.1217     0.121777  0.000112147  0.000189343
    0.461663     0.461688     0.461663  2.4472e-005 1.66533e-016
    0.541332     0.541355     0.541576  2.3166e-005  0.000244265
```

and Figure 5; see Figures 5(a) and 5(b) for the exact and approximate adjusted projection depth regions based on `ExVec2` and `AppVec2`, respectively.

4.2. Trivariate example: Boston housing data

Our next example is a trivariate data set, which is actually a part of the Boston housing data downloaded from http://lib.stat.cmu.edu/datasets/boston/. In the sequel we take the first 65 items of variables `rm`, `dis` and `lstat`, where `rm` denotes the average number of rooms per dwelling, `dis` the weighted distances to five Boston employment centers, and `lstat` the percentage of lower status of the population, respectively.



(a) 0.5-projection depth region.

(b) 0.4-projection depth region.

(c) 0.3-projection depth region.

(d) 0.2-projection depth region.

Figure 6: 0.2, 0.3, 0.4, 0.5 projection depth regions of `BostData`.

The data can be obtained through calling

```
load BostData
X = BostData;
```

The function `ExVecPDHD` is used to find the ODVs. One can trace this program by calling

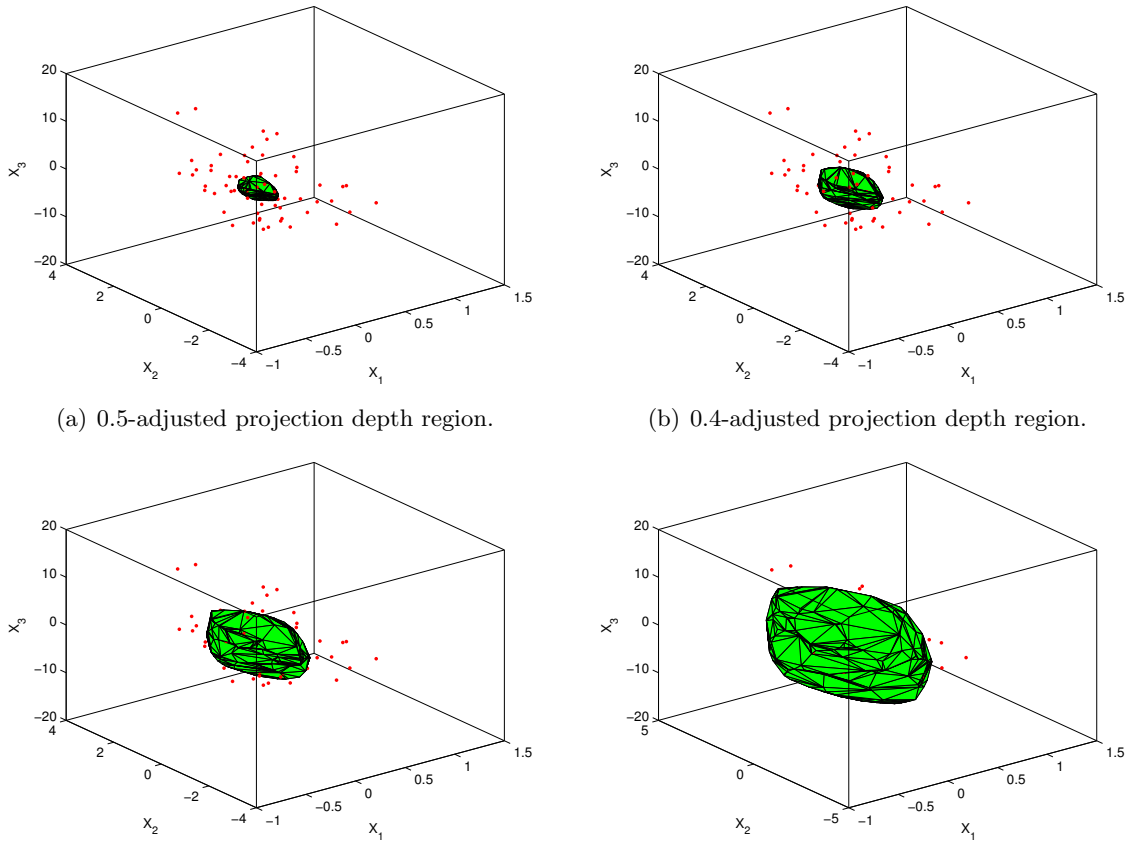
```
ExVec3 = ExVecPDHD(X, true)
```

This leads to

```
Sub-region NO.:    1         Elapsed time is (s):    14.6268
Sub-region NO.:    2         Elapsed time is (s):    9.4688
Sub-region NO.:    3         Elapsed time is (s):    17.3068
Sub-region NO.:    4         Elapsed time is (s):    3.946

ExVec3 =

             u: [3x111318 double]
```



(a) 0.5-adjusted projection depth region.

(b) 0.4-adjusted projection depth region.

(c) 0.3-adjusted projection depth region.

(d) 0.2-adjusted projection depth region.

Figure 7: 0.2, 0.3, 0.4, 0.5 adjusted projection depth regions of `BostData`.

```
    MedVal: [111318x1 double]
    MADVal: [111318x1 double]
NumDirVecU: 111318
```

Based on `ExVec3`, the projection depth and its induced estimators can be computed by calling functions such as `PDVal` and `PM` similar to the bivariate example given above. The exception is the computation of depth regions, which needs to call the function `PC3D`. That is,

```
alpha0 = [0.5, 0.4, 0.3, 0.2];
PC3D(X, ExVec3, alpha0, false)
```

The resulting 0.5, 0.4, 0.3, 0.2 regions are by default plotted in separated figures; see Figure 6. Similarly, by calling

```
ExVec4 = ExVecAPDHD(X, true);
exapdv = APDVal(X, ExVec4)

APC3D(X, ExVec4, alpha0, false)
```

one can obtain the adjusted projection depth values of all the points of `X`, and its related depth regions; see Figure 7.

5. Summary

As a major depth notion, projection depth has many desirable properties. It can induce favorable estimators with high breakdown point robustness, without having to pay a price of losing too much efficiency. It has continuous depth regions. For a given data cloud, the depth value does not vanish even outside the convex hull. Its induced median is unique, etc. However, application packages which can be used to compute the projection depth and its related estimators are currently lacking, although sporadically algorithms exist in individual papers.

In this paper, we present a MATLAB package **CompPD**, which is in fact a MATLAB implementation of the latest developments about the computation of projection depth and its related estimators (Zuo and Lai 2011; Liu *et al.* 2013; Liu and Zuo 2014). The current version has its limitations. We are committed to continuous improvement of the package as research advances in this field. We wish that the developed package has the potential to help practitioners to obtain satisfactory results when the data are contaminated or heavy-tailed.

Acknowledgments

The authors would like to thank Prof. Paindaveine Davy for providing to us the code `compContourM1.m` (`http://homepages.ulb.ac.be/~dpaindav/`), and Dr. Šiman for helpful discussions during the preparation of this package. The authors also greatly appreciate the thoughtful and constructive remarks of the co-editors and two anonymous referees, which led to many improvements in this paper.

This research is partly supported by the National Natural Science Foundation of China (Grant No. 11461029, No. 11161022, No. 61263014), the Natural Science Foundation of Jiang-xi Province (No. 20142BAB211014, No. 20122BAB201023, No. 20132BAB201011, No. 20142BCB23013), and the Youth Science Fund Project of Jiangxi provincial education department (No. GJJ14350, No. KJLD14034), Program for Excellent Youth Talents of JXUFE (201401).

References

Barber CB, Dobkinm DP, Huhdanpaa H (1996). "The Quickhull Algorithm for Convex Hulls." *ACM Transactions on Mathematical Software*, **22**(4), 469–483.

Bazovkin P, Mosler K (2012). "An Exact Algorithm for Weighted-Mean Trimmed Regions in Any Dimension." *Journal of Statistical Software*, **47**(13), 1–29. URL `http://www.jstatsoft.org/v47/i13/`.

Chenouri S, Small C (2012). "A Nonparametric Multivariate Multisample Test Based on Data Depth." *Electronic Journal of Statistics*, **6**, 760–782.

Dang X, Serfling R (2010). "Nonparametric Depth-Based Multivariate Outlier Identifiers, and Masking Robustness Properties." *Journal of Statistical Planning and Inference*, **140**(1), 198–213.

Donoho DL (1982). *Breakdown Properties of Multivariate Location Estimators*. Ph.D. thesis, Department of Statistics, Harvard University.

Dyckerhoff R, Mosler K, Koshevoy G (1996). "Zonoid Data Depth: Theory and Computation." In A Prat (ed.), *COMPSTAT 1996 – Proceedings in Computational Statistics*, pp. 235–240. Physica-Verlag, Heidelberg.

Genest M, Masse JC, Plante JF (2012). ***depth**: Depth Functions Tools for Multivariate Analysis*. R package version 2.0-0, URL http://CRAN.R-project.org/package=depth.

Ghosh AK, Chaudhuri P (2005). "On Maximum Depth and Related Classifiers." *Scandinavian Journal of Statistics*, **32**(2), 328–350.

Hallin M, Paindaveine D, Šiman M (2010). "Multivariate Quantiles and Multiple-Output Regression Quantiles: From L_1 Optimization to Halfspace Depth." *The Annals of Statistics*, **38**(2), 635–669.

Hubert M, Van der Veeken S (2010). "Robust Classification for Skewed Data." *Advances in Data Analysis and Classification*, **4**(4), 239–254.

Koshevoy G, Mosler K (1997). "Zonoid Trimming for Multivariate Distributions." *The Annals of Statistics*, **25**(5), 1998–2017.

Li J, Cuesta-Albertos J, Liu RY (2012). "DD-Classifier: Nonparametric Classification Procedures Based on DD-Plots." *Journal of the American Statistical Association*, **107**(498), 737–753.

Liu RY (1990). "On a Notion of Data Depth Based on Random Simplices." *The Annals of Statistics*, **18**(1), 191–219.

Liu RY (1992). "Data Depth and Multivariate Rank Tests." In Y Dodge (ed.), *L_1-Statistical Analysis and Related Methods*, pp. 279–294. North-Holland, Amsterdam.

Liu RY, Parelius JM, Singh K (1999). "Multivariate Analysis by Data Depth: Descriptive Statistics, Graphics and Inference." *The Annals of Statistics*, **27**(3), 783–858.

Liu RY, Serfling R, Souvaine DL (2006). *Data Depth: Robust Multivariate Analysis, Computational Geometry, and Applications*, volume 72 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society.

Liu XH, Zuo YJ (2014). "Computing Projection Depth and Its Associated Estimators." *Statistics and Computing*, **24**(1), 51–63.

Liu XH, Zuo YJ, Wang ZZ (2013). "Exactly Computing Bivariate Projection Depth Contours and Median." *Computational Statistics & Data Analysis*, **60**, 1–11.

Mosler K, Lange T, Bazovkin P (2009). "Computing Zonoid Trimmed Regions of Dimension $d \geq 2$." *Computational Statistics & Data Analysis*, **53**(7), 2500–2510.

Paindaveine D, Šiman M (2012). "Computing Multiple-Output Regression Quantile Regions." *Computational Statistics & Data Analysis*, **56**(4), 840–853.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rousseeuw PJ, Hubert M (1999). "Regression Depth." *Journal of the American Statistical Association*, **94**(446), 388–433.

Rousseeuw PJ, Ruts I (1996). "Algorithm AS 307: Bivariate Location Depth." *Journal of the Royal Statistical Society C*, **45**(4), 516–526.

Rousseeuw PJ, Struyf A (1998). "Computing Location Depth and Regression Depth in Higher Dimensions." *Statistics and Computing*, **8**(1), 193–203.

Serfling R (2006). "Depth Functions in Nonparametric Multivariate Inference." In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 72. American Mathematical Society.

Stahel WA (1981). "Breakdown of Covariance Estimators." *Research Report 31*, Fachgruppe für Statistik. ETH, Zürich.

The MathWorks Inc (2009). *MATLAB – The Language of Technical Computing, Version 7.8.0*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Tukey JW (1975). "Mathematics and the Picturing of Data." In *Proceedings of the International Congress of Mathematicians*, pp. 523–531. Montreal. Canadian Mathematical Congress.

Yeh A, Singh K (1997). "Balanced Confidence Sets Based on the Tukey's Depth and the Bootstrap." *Journal of the Royal Statistical Society B*, **59**(3), 639–652.

Zuo YJ (2003). "Projection-Based Depth Functions and Associated Medians." *The Annals of Statistics*, **31**(5), 1460–1490.

Zuo YJ (2006). "Multidimensional Trimming Based on Projection Depth." *The Annals of Statistics*, **34**(5), 2211–2251.

Zuo YJ (2013). "Multidimensional Medians and Uniqueness." *Computational Statistics & Data Analysis*, **66**, 82–88.

Zuo YJ, Cui HJ, He XM (2004). "On the Stahel-Donoho Estimators and Depth-Weighted Means for Multivariate Data." *The Annals of Statistics*, **32**(1), 189–218.

Zuo YJ, Lai S (2011). "Exact Computation of Bivariate Projection Depth and the Stahel-Donoho Estimator." *Computational Statistics & Data Analysis*, **55**(3), 1173–1179.

Zuo YJ, Serfling R (2000). "General Notions of Statistical Depth Function." *The Annals of Statistics*, **28**(2), 461–482.

Zuo YJ, Ye X (2009). ***ExPD2D**: Exact Computation of Bivariate Projection Depth Based on Fortran Code*. R package version 1.0.1, URL http://CRAN.R-project.org/src/contrib/Archive/ExPD2D/.

Affiliation:

Xiaohui Liu
School of Statistics
Research Center of Applied Statistics
Jiangxi University of Finance and Economics
Nanchang, Jiangxi 330013, China
E-mail: csuliuxh912@gmail.com
URL: http://stat.jxufe.cn/html/20131123/n2976385.html

Department of Statistics and Probability
Michigan State University
East Lansing, MI 48823, United States of America
E-mail: zuo@msu.edu
URL: http://www.stt.msu.edu/users/zuo/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 65, Issue 2 *Submitted:* 2013-05-13
May 2015 *Accepted:* 2014-08-08