# LIBRA: a MATLAB Library for Robust Analysis

Sabine Verboven [*] Mia Hubert[†]

June 18, 2004

**Abstract**

Since MATLAB is very popular in industry and academia, and is frequently used by chemometricians, statisticians, chemists, and engineers, we introduce a MATLAB library of robust statistical methods. Those methods were developed because their classical alternatives produce unreliable results when the data set contains outlying observations. Our toolbox currently contains implementations of robust methods for location and scale estimation, covariance estimation (FAST-MCD), regression (FAST-LTS, MCD-regression), principal component analysis (RAPCA, ROBPCA), principal component regression (RPCR), partial least squares (RSIMPLS) and classification (RDA). Only a few of these methods will be highlighted in this paper. The toolbox also provides many graphical tools to detect and classify the outliers. The use of these features will be explained and demonstrated through the analysis of some real data sets.

**Keywords:** Robustness, Multivariate calibration, PCA, PCR, PLS, Classification, MATLAB library.

## 1 Introduction

The need and effectiveness of robust methods has been described in many papers and books, see e.g. [1, 2, 3]. Robust methods are developed because atypical observations in a data

[*]Universiteit Antwerpen, Departement Wiskunde-Informatica, Middelheimlaan 1, B-2020 Antwerpen, sabine.verboven@ua.ac.be

[†]Katholieke Universiteit Leuven, Departement Wiskunde, W. de Croylaan 54, B-3001 Leuven, mia.hubert@wis.kuleuven.ac.be

set heavily affect the classical estimates. Consider for example, estimating the center of the following univariate data set: (10 10.1 10.1 10.1 10.2 10.2 10.3 10.3). For these data, the classical mean is 10.16 whereas the robust median equals 10.15. They do not differ very much as there are no outliers. However, if the last measurement was wrongly recorded as 103 instead of 10.3, the mean becomes 21.75 whereas the median still equals 10.15. As in this example, outliers can occur by mistake (misplacement of a comma), or e.g. through a malfunction of the machinery or a measurement error. These are typically the samples that should be discovered and removed from the data unless one is capable to correct their measurements. Another type of atypical observations are those that belong to another population than the one under study (for example, by a change in the experimental conditions) and often they reveal unique properties. Consequently, finding this kind of outliers can lead to new discoveries. Outliers are thus not always wrong or bad, although this terminology is sometimes abusively used.

Over the years, several robust methods became available in SAS [4] and S-PLUS/R [5], [6]. To make them also accessible for MATLAB users, we started collecting robust methods in a MATLAB library. The toolbox mainly contains implementations of methods that have been developed at our research groups at the University of Antwerp and the Katholieke Universiteit Leuven. It currently contains functions for location and scale estimation, covariance estimation (FAST-MCD), regression (FAST-LTS, MCD-regression), principal component analysis (RAPCA, ROBPCA), principal component regression (RPCR), partial least squares (RSIMPLS) and classification (RDA).

In this paper we will highlight some methods of the toolbox and apply them to real data sets. We distinguish between low and high-dimensional data since they require a different approach. The application of robust methods not only yield estimates which are less influenced by outliers, they also allow to detect the outlying observations by looking at the residuals from a robust fit. That's why we have included in our toolbox many graphical tools for model checking and outlier detection. In particular we will show how to interpret several diagnostic plots that are developed to visualize and classify the outliers.

In Section 2 a few estimators of location, scale and covariance (including PCA) will be considered. Some robust regression methods are discussed in Section 3. Robust classification is illustrated in Section 4. Finally, Section 5 contains a list of the currently available main functions.

# 2 Location, scale and covariance estimators

## 2.1 Low dimensional estimators

Robust estimators of location and scale for univariate data include the median, the median absolute deviation, and M-estimators. In our toolbox we have included several methods which are described in [7] (and references therein). Also the medcouple, which is a robust estimator of skewness [8], is available. Here, we will concentrate on the problem of location and covariance estimation of multivariate data as it is the cornerstone of many multivariate techniques such as PCA, calibration and classification.

In the multivariate location and scatter setting we assume that the data are stored in an $n \times p$ data matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ with $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ the $i$th observation. Hence $n$ stands for the number of objects and $p$ for the number of variables. In this section we assume in addition that the data are low-dimensional. Here, this means that $p$ should at least be smaller than $n/2$ (or equivalently that $n > 2p$).

Robust estimates of the center $\boldsymbol{\mu}$ and the scatter matrix $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$ can be obtained by the Minimum Covariance Determinant (MCD) estimator [9]. The MCD method looks for the $h(> n/2)$ observations (out of $n$) whose classical covariance matrix has the lowest possible determinant. The raw MCD estimate of location is then the average of these $h$ points, whereas the raw MCD estimate of scatter is their covariance matrix, multiplied with a consistency factor. Based on these raw MCD estimates, a reweighting step can be added which increases the finite-sample efficiency considerably [10].

The MCD estimates can resist $(n - h)$ outliers, hence the number $h$ (or equivalently the proportion $\alpha = h/n$) determines the robustness of the estimator. The highest resistance towards contamination is achieved by taking $h = [(n + p + 1)/2]$. When a large proportion of contamination is presumed, $h$ should thus be chosen close to $\alpha n$ with $\alpha = 0.5$. Otherwise an intermediate value for $h$, such as $0.75n$, is recommended to obtain a higher finite-sample efficiency. This is also the default setting in our MATLAB implementation. It will give accurate results if the data set contains at most 25% of aberrant values, which is a reasonable assumption for most data sets.

The computation of the MCD estimator is non-trivial and naively requires an exhaustive investigation of all $h$-subsets out of $n$. In [10] a fast algorithm is presented (FAST-MCD) which avoids such a complete enumeration. It is a resampling algorithm which starts by drawing 500 random $p+1$ subsets from the full data set. This number is chosen to ensure a high probability of sampling at least one clean subset. The *mcdcov* function in our toolbox is an implementation of this FAST-MCD algorithm. Note that the MCD can only be computed if $p < h$, otherwise the covariance matrix of any $h$-subset has zero determinant. Since $n/2 < h$, we thus require that $p < n/2$. However, detecting several outliers becomes intrinsically delicate when $n/p$ is small as some data points may become coplanar by chance. This is an instance of the "curse of dimensionality". It is therefore recommended that $n/p > 5$ when $\alpha = 0.5$ is used. For small $n/p$ it is preferable to use the MCD with $\alpha = 0.75$.

With our toolbox the reweighted MCD-estimator of a data set $X$ is computed by typing

```
>> out=mcdcov(X)
```

at the command line. Doing so, the user accepts all the default settings: $\alpha = 0.75$, 'plots'=1, 'cor'=0, 'ntrial'=500. It means that diagnostic plots will be drawn, no correlation matrix will be computed and the algorithm uses 500 random initial $p+1$ subsets.

If the user wants to change one or more of these default settings, the input arguments and their new values have to be specified. Assume for example that we have no idea about the amount of contamination and we prefer to apply a highly robust method. If in addition, we are interested in the corresponding MCD correlation matrix, we set:

```
>> out=mcdcov(X,'alpha',0.50,'cor',1)
```

Similar to all MATLAB built-in graphical functions, we have chosen to work with variable input arguments. More precisely, the input arguments in the function header consist of $N$ required input arguments and a variable range of optional arguments

```
>> result = functionname(required1,required2,...,requiredN,varargin)
```

Depending on the application, required input arguments are e.g. the design matrix $\boldsymbol{X}$, the response variable $\boldsymbol{y}$ in regression, the group numbers in a classification context etc. The function call should assign a value to all the required input arguments in the correct order. However, the optional arguments can be omitted (which implies that the defaults are used) or they can be called in an arbitrary order. For example,

```
>> out=mcdcov(X,'cor',1,'alpha',0.50)
```

or

```
>> out=mcdcov(X,'cor',1,'alpha',0.50,'plots',1)
```

would produce the same result. All our main functions use these user-friendly variable input arguments. Standard MATLAB functions on the other hand (except the graphical ones) contain all the input arguments in an ordered way such that none of the in-between arguments can be left out.

The output of *mcdcov* (and of many other functions in the toolbox) is stored as a structure. A structure in MATLAB is an array variable which can contain fields of different types and/or dimensions. If a single output variable is assigned to the *mcdcov* function, the given solution is the reweighted MCD-estimator. To obtain the results of the raw MCD estimator, a second output variable has to be assigned

```
>> [rew,raw]=mcdcov(X)
```

The structure and content of the output variable(s) is explained in the next example.

**Example 1:** To illustrate the MCD method we analyze the Stars data set [3]. This is a bivariate data set ($p = 2$) where the surface temperature and the light intensity of $n = 47$ stars were observed. The data were preprocessed by taking a logarithmic transformation of both variables. Applying 'out=mcdcov(X)' yields the following output structure:

```
 center : [4.4128   4.9335]
    cov : [2 × 2 double]
    cor : []
      h : 36
  alpha : 0.7500
     md : [1 × 47 double]
     rd : [1 × 47 double]
   flag : [1 × 47 logical]
 cutoff : [1 × 1 struct]
  plane : []
 method : [1 × 42 char]
  class : 'MCDCOV'
classic : 0
      X : [47 × 2 double]
```

Here, the output consists of several fields that contain the location estimate ('out.center'), the estimated covariance matrix ('out.cov') and eventually the correlation matrix ('out.cor'). Other fields such as 'out.h' and 'out.alpha' contain information about the MCD method, while some of the components (e.g. 'out.rd', 'out.flag', 'out.md' and 'out.cutoff') can be used for outlier detection and to construct some graphics. The 'out.class' field is used by the *makeplot* function to detect which figures should be created for this analysis. Detailed information about this output structure (and that of the raw estimates) can be found in the help-file of the *mcdcov* function.

The robust distance of an observation $i$ is used to detect whether it is an outlier or not. It is defined as

$$\mathrm{RD}_i = \sqrt{(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_{\mathrm{MCD}})^T \hat{\boldsymbol{\Sigma}}_{\mathrm{MCD}}^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_{\mathrm{MCD}})} \tag{1}$$

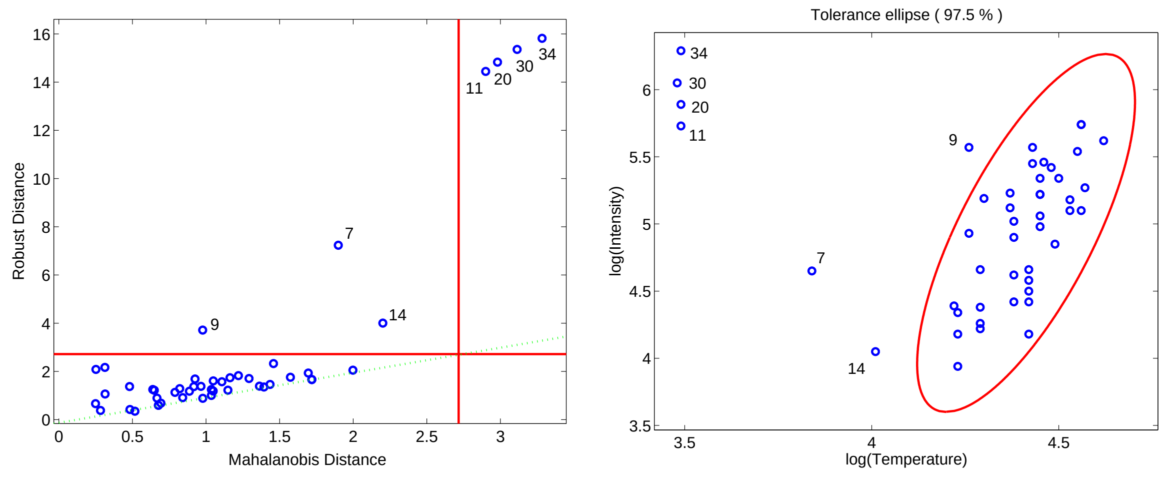
with $\hat{\boldsymbol{\mu}}_{\mathrm{MCD}}$ and $\hat{\boldsymbol{\Sigma}}_{MCD}$ the MCD location and scatter estimates. This robust distance is the straightforward robustification of the Mahalanobis distance

$$\mathrm{MD}_i = \sqrt{(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})} \tag{2}$$

which uses the classical mean $\bar{\boldsymbol{x}}$ and empirical covariance matrix $\boldsymbol{S}$ as estimates of location and scatter. Under the normal assumption, the outliers are those observations having a robust distance larger than the cutoff-value $\sqrt{\chi^2_{p,0.975}}$, and they receive a flag equal to zero. The regular observations whose robust distance does not exceed $\sqrt{\chi^2_{p,0.975}}$ have a flag equal to one.

Based on the robust distances and the Mahalanobis distances, several graphical displays are provided to visualize the outliers and to compare the robust and the classical results. One of them is the distance-distance plot [10] which displays for each observation its robust distance $\mathrm{RD}_i$ versus its Mahalanobis distance $\mathrm{MD}_i$. A horizontal and a vertical line are drawn at the cut-off value $\sqrt{\chi^2_{p,0.975}}$. For the stars data, we obtain Figure 1(a) with the cutoff-lines drawn at 2.72. Looking at Figure 1(a) we clearly see four outlying observations: 11, 20, 30 and 34. Those four observations are known to be giant stars and hence strongly differ from the other stars in the main sequence. Both a classical and a robust analysis would identify these observations as they exceed the horizontal and the vertical cut-off line. Further we observe 3 observations (7, 9, and 14) with a large robust but a small Mahalanobis distance. They would not be recognized with a classical approach.

As this data set is bivariate, we can understand its structure much better by making a scatter plot as in Figure 1(b). Superimposed is the 97.5% robust confidence ellipse defined as the set of points whose robust distance is equal to $\sqrt{\chi^2_{p,0.975}}$. Obviously, observations outside this tolerance ellipse then correspond with the outliers. We again notice the 7 outliers found with the MCD method.



(a) (b)

Figure 1: Analysis of the Star data set : (a) distance-distance plot ; (b) data with 97.5% tolerance ellipse.

Note that these plots are automatically generated by the *mcdcov* function. Unless the input argument 'plots' is set to 0, all the main functions call the *makeplot* function at the end of the procedure. A small menu with buttons is then displayed, from which the user can choose one, several, or all available plots associated with the analysis made. For example, *mcdcov* offers the menu shown in Figure 2. If the input argument 'classic' is also set to one in the call to the main function, some buttons also yield the figures associated with the classical analysis. To illustrate, the plots in Figure 3 are the result of computing

```
>> out=mcdcov(X,'classic',1)
```

and pressing the buttons 'Tolerance ellipse' and 'Index plot of the distances'. We see in Figure 3(a) that the robust and the classical tolerance ellipse are superimposed on the same plot which is easier for comparison. To generate plots, one can also set the input argument 'plots' equal to zero, and make a separate call to the *makeplot* function. In our example, by typing

```
>>   out=mcdcov(X,'classic',1,'plots',0)
>>   makeplot(out,'classic',1)
```

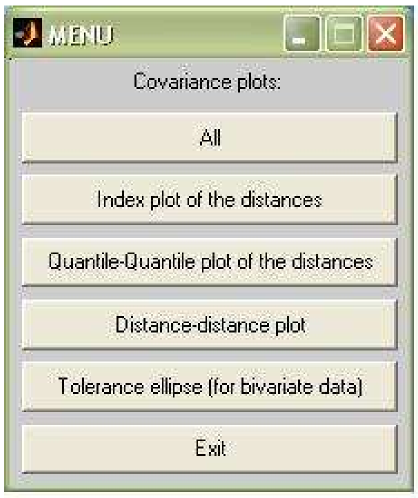
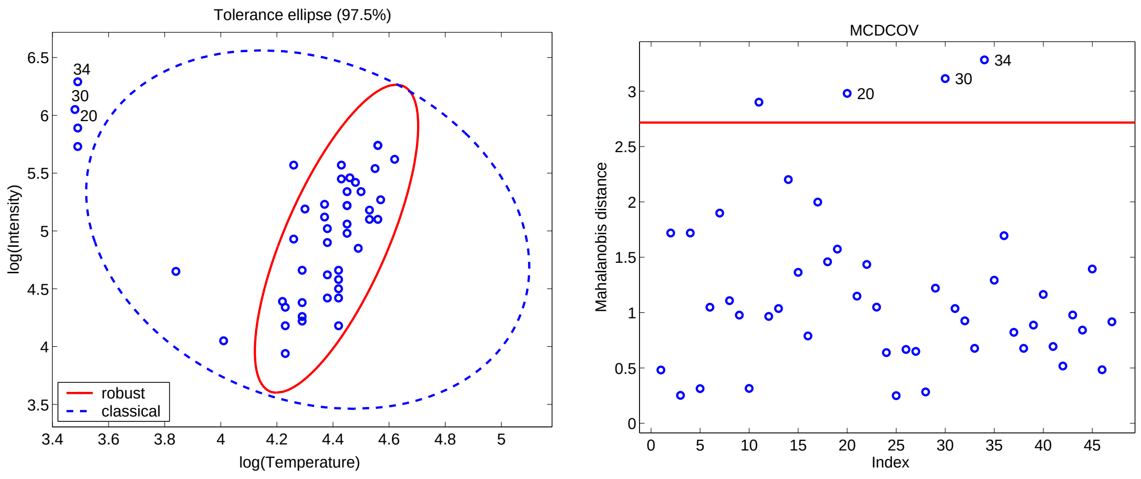

MENU
Covariance plots:
All
Index plot of the distances
Quantile-Quantile plot of the distances
Distance-distance plot
Tolerance ellipse (for bivariate data)
Exit

Figure 2: The graphical user interface invoked by the *makeplot* function, applied to an MCDCOV attribute.



(a) (b)

Figure 3: Analysis of the Star data set : (a) data with 97.5% classical and robust tolerance ellipses; (b) Index plot of the Mahalanobis distances.

The appearance of the graphical menu can be avoided by adding the name of the desired graph as input argument to the *makeplot* function. To generate Figure 1 we could e.g. use the commands

```
>>  out=mcdcov(X,'plots',0)
>>  makeplot(out,'nameplot','dd')
>>  makeplot(out,'nameplot','ellipse')
```

whereas Figure 3 can be created by typing

```
>>  out=mcdcov(X,'classic',1,'plots',0)
>>  makeplot(out,'nameplot','ellipse','classic',1)
>>  makeplot(out,'nameplot','mahdist','classic',1)
```

## 2.2 High dimensional estimators

If $\boldsymbol{X}$ contains more variables than observations ($p \gg n$) its covariance structure can be estimated by means of a principal component analysis (PCA). In general, PCA constructs a new set of $k \ll p$ variables, called loadings, which are linear combinations of the original variables and which contain most of the information. These loading vectors span a $k$-dimensional subspace. Projecting the observations onto this subspace yields the scores $\boldsymbol{t}_i$ which for all $i = 1, \ldots, n$ satisfy

$$\boldsymbol{t}_i = \boldsymbol{P}_{k,p}^T(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_x) \tag{3}$$

with the loading matrix $\boldsymbol{P}_{p,k}$ containing the $k$ loading vectors columnwise, and $\hat{\boldsymbol{\mu}}_x$ being an estimate of the center of the data. From here on, the subscripts to a matrix serve to recall its size, e.g. $\boldsymbol{P}_{p,k}$ is an $p \times k$ matrix. In classical PCA these loadings vectors correspond with the $k$ dominant eigenvectors of $\boldsymbol{S}$, the empirical covariance matrix of the observations $\boldsymbol{x}_i$, whereas $\hat{\boldsymbol{\mu}}_x$ is just the sample mean. As a byproduct of a PCA analysis, an estimate of the covariance structure of $\boldsymbol{X}$ can be obtained by the matrix product

$$\hat{\boldsymbol{\Sigma}}_{p,p} = \boldsymbol{P}_{p,k}\boldsymbol{L}_{k,k}\boldsymbol{P}_{k,p}^T \tag{4}$$

with $\boldsymbol{L}_{k,k}$ a diagonal matrix containing the eigenvalues from the PCA analysis.

A robust PCA method yields robust loadings $\boldsymbol{P}$, robust eigenvalues $\boldsymbol{L}$, a robust center $\hat{\boldsymbol{\mu}}_x$ and robust scores according to (3). Our toolbox contains two robust PCA methods. The first one, called RAPCA [11], is solely based on projection pursuit, and is very fast. The second method, ROBPCA [12] combines projection pursuit ideas with the MCD estimator and outperforms RAPCA in many situations. The computation time of ROBPCA is still very reasonable although slower than RAPCA (e.g. it takes only 4.7 seconds on a Pentium IV with 2.40 GHz to perform ROBPCA on a data set with $n = 30$ and $p = 2050$).

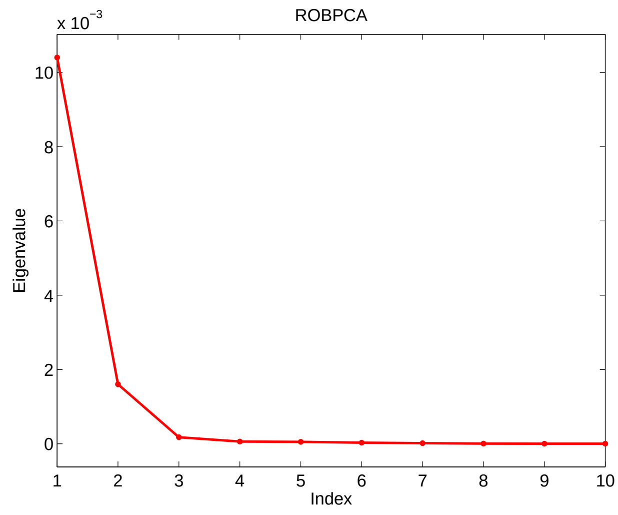
**Example 2:** We will illustrate ROBPCA and its diagnostic tools on the Octane data set [13] which consists of $n = 39$ NIR-spectra of gasoline at $p = 251$ wavelengths. When we would know the optimal number of components $k$, we could call the *robpca* method with $k$ specified, e.g.

```
>> out=robpca(X,'k',3)
```

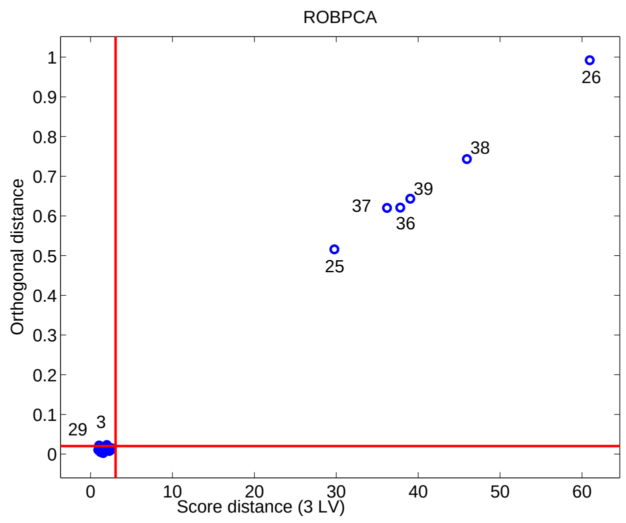
Otherwise, we need to select the number of principal components $k$. The function call is then just simply

```
>> out=robpca(X)
```

which generates a scree plot. This is a graph of the monotone decreasing eigenvalues. The optimal number $k$ is then often selected as the one where the kink in the curve appears. From the scree plot of the Octane data in Figure 4(a), we decide to retain $k = 3$ components for further analysis. To visualize and to classify the outliers, we can make a (score) outlier



(a)



(b)

Figure 4: PC analysis of the Octane data set : (a) Screeplot ; (b) Score diagnostic plot.

map. For each observation, it displays on the $x$-axis the score distance $\mathrm{SD}_i$ within the PCA subspace

$$\mathrm{SD}_i = \sqrt{\boldsymbol{t}_i^T L^{-1} \boldsymbol{t}_i} \tag{5}$$

and on the $y$-axis the orthogonal distance to the PCA subspace

$$\mathrm{OD}_i = ||\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_x - \boldsymbol{P}_{p,k}\boldsymbol{t}_i||. \tag{6}$$

This yields a classification of the observations as summarized in Table 1: regular data (with small SD and small OD), good PCA-leverage points (with large SD and small OD), orthogonal outliers (with small SD and large OD) and bad PCA-leverage points (with large SD and large OD). The latter two types of observations are highly influential for classical PCA as this method tries to make all orthogonal distances as small as possible. For more information on this outlier map and the horizontal and vertical cut-off values, we refer to Reference [12]. On the outlier map of the Octane data in Figure 4(b) we immediately spot the samples 25, 26, 36-39 as bad PCA-leverage points. It is known that these samples contain added alcohol, hence their spectra are indeed different from the other spectra. This becomes very clear from the outlier map.

Table 1: Overview of the different types of observations based on their score distance and their orthogonal distance.

| Distances | small SD | large SD |
|---|---|---|
| large OD | orthogonal outlier | bad PCA-leverage point |
| small OD | regular observation | good PCA-leverage point |

## 3 Robust calibration

### 3.1 Low dimensional regressors

The multiple linear regression model assumes that in addition to the $p$ regressors or $x$-variables, a response variable $y$ is measured, which can be explained as an affine combination of the predictors. More precisely, the model says that for all observations $(\boldsymbol{x}_i, y_i)$ with $i = 1, \ldots, n$, it holds that

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i \qquad i = 1, \ldots, n \tag{7}$$

where the errors $\epsilon_i$ are assumed to be independent and identically distributed with zero mean and constant variance $\sigma^2$. Applying a regression estimator to the data yields $p+1$ regression coefficients $\hat{\boldsymbol{\theta}} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)$. The residual $r_i$ of case $i$ is defined as $y_i - \hat{y}_i = y_i - \hat{\boldsymbol{\theta}}^T \boldsymbol{x}_i$.

The ordinary least squares (OLS) estimator minimizes the sum of the squared residuals, but is very sensitive to outliers. Our toolbox contains the Least Trimmed Squares (LTS) estimator [9] which minimizes the sum of the $h$ smallest squared residuals, or

$$\hat{\boldsymbol{\theta}}_{\mathrm{LTS}} = \min_{\theta} \sum_{i=1}^{h} (r^2)_{i:n} \tag{8}$$

Note that the residuals are first squared, and then ordered. The interpretation of the $h$ value is the same as for the MCD estimator, and should be chosen between $n/2$ and $n$. Taking $h = n$ yields the OLS estimator. The MATLAB function *ltsregres* contains an implementation of the FAST-LTS algorithm [14] which is similar to the FAST-MCD method. The output of *ltsregres* includes many graphical tools for model checking and outlier detection, such as a normal quantile plot of the residuals, and a residual outlier map. The latter displays the standardized LTS residuals (the residuals divided by a robust estimate of their scale) versus the robust distances obtained by applying the MCD estimator on the $x$-variables. Its interpretation will be discussed in Section 3.2.

If we want a reliable prediction of $q > 1$ properties at once, based on a set of $p$ explanatory variables, the use of a *multivariate* regression method is appropriate. The linear regression model is then formulated as

$$\boldsymbol{y}_i = \boldsymbol{\beta}_0 + \mathcal{B}^T \boldsymbol{x}_i + \boldsymbol{\varepsilon}_i \qquad i = 1, \ldots, n \tag{9}$$

with $\boldsymbol{y}_i$ and $\boldsymbol{\varepsilon}_i$ being $q$-dimensional vectors that contain the response values, respectively the error terms of the $i$th observation with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$.

To cope with such data, we can use the MCD-regression method [15, 16]. It is based on the MCD estimates of the joint $(x, y)$ variables and thereby yields a highly robust method. Within our toolbox, this method can be called with the *mcdregres* function.

## 3.2 High dimensional regressors

In case the number of variables $p$ is larger than the number of observations $n$, there is no unique OLS solution, and neither can the LTS estimator be computed. Moreover, in any data set with highly correlated variables, called multicollinearity, both OLS and LTS have a high variance. Two very popular solutions to this problem are offered by Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR). Both methods first project the $x$-data onto a lower dimensional space without loosing too much important information. Then a multiple or multivariate regression analysis is performed in this lower dimensional design space.

More precisely, both PCR and PLSR assume the following bilinear model which expresses the connection between the response variable(s) and the explanatory variables as:

$$\boldsymbol{x}_i = \boldsymbol{\mu}_x + \boldsymbol{P}_{p,k}\boldsymbol{t}_i + \boldsymbol{f}_i \tag{10}$$

$$\boldsymbol{y}_i = \boldsymbol{\mu}_y + \mathcal{A}_{q,k}\boldsymbol{t}_i + \boldsymbol{g}_i \tag{11}$$

with $\boldsymbol{t}_i$ the $k$-dimensional scores, $k \ll p$ and $q \geq 1$.

To construct the scores $\boldsymbol{t}_i$, PCR applies PCA to the $x$-variables, whereas in PLSR they are constructed by maximizing the covariance between linear combinations of the $x$ and $y$-variables. Robust versions of PCR and PLSR have been constructed in [16] and [17] with corresponding MATLAB functions *rpcr* and *rsimpls*. In analogy with classical PCR and PLSR, they apply ROBPCA on the $x$-variables, respectively the joint $(x, y)$ variables. Next, a robust regression is applied to model (11). For PCR we can use the LTS or the MCD regression, whereas for PLSR a regression based on the ROBPCA results is performed.

**Example 2 continued:** Let us look again at the Octane data from Section 2.2. The univariate ($q = 1$) response variable contains the octane number. We know from the previous analysis that in samples 25,26,36-39 alcohol was added. We cannot use an ordinary least squares regression since the data clearly suffers from multicollinearity, so we apply a robust PCR analysis. We then first need to determine the optimal number of components $k$. To this end, the robust cross-validated RMSE can be computed at the model with $k = 1, \ldots, k_{\max}$ components. For a formal definition, see [16]. This is a rather time-consuming approach, but faster methods for its computation have been developed and will become part of the toolbox [18]. Alternatively, a robust $R^2$-plot can be made. To look at the RMSECV-curve, the function call in MATLAB becomes:

```
>> out=rpcr(X,y,'rmsecv',1)
```

Here, the robust RMSECV-curve in Figure 5(a) shows $k = 3$ as the optimal number of latent variables (LV).

The output of the *rpcr* function also includes the score and orthogonal distances from the robust PCA analysis. So by default the PCA-outlier map will again be plotted (Figure 4b).

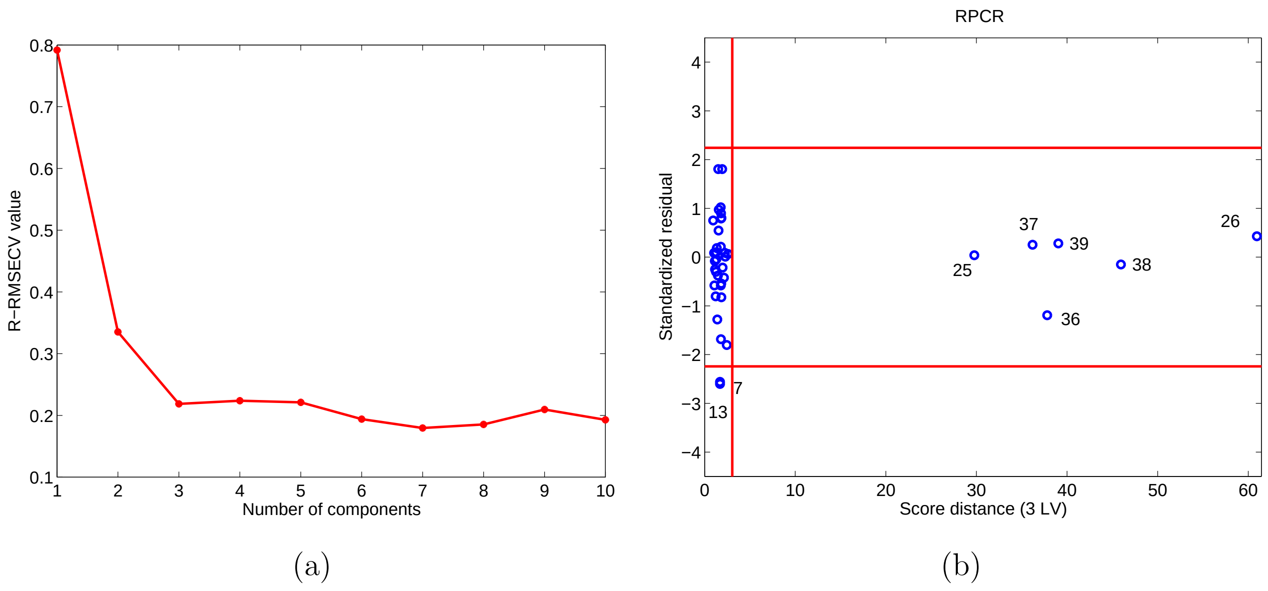


Figure 5: Analysis of the Octane data set : (a) Robust RMSECV-curve; (b) Regression outlier map.

Table 2: Overview of the different types of observations based on their score distance and their absolute residual.

| Distances | small SD | large SD |
|---|---|---|
| large absolute residual | vertical outlier | bad leverage point |
| small absolute residual | regular observation | good leverage point |

In addition, a residual outlier map is generated. It displays the standardized LTS residuals versus the score distances, and can be used to classify the observations according to the regression model. As can be seen from Table 2, we distinguish regular data (small SD and small absolute residual), good leverage points (large SD but small absolute residual), vertical outliers (small SD and large absolute residual), and bad leverage points (large SD and large absolute residuals). Large absolute residuals are those that exceed $\sqrt{\chi^2_{1,0.975}} = 2.24$. Under normally distributed errors, this happens with probability 2.5%. The vertical outliers and bad leverage points are the most harmful for classical calibration methods as they disturb the linear relationship.

Note that the observations that are labeled on the outlier map are by default the three cases with the largest score distance, and those three that have the largest absolute standardized residual. To change these settings, one should assign different values to the ‘labod’, 'labsd' and 'labrd' input arguments. For example, Figure 5(b) was the result of

```
>> makeplot(out,'labsd',6,'labrd',2)
```

In multivariate calibration ($q > 1$) the residuals are also $q$-dimensional with estimated covariance matrix $\hat{\boldsymbol{\Sigma}}_\varepsilon$. The outlier map then displays on the vertical axis the residual distances, defined as

$$\text{ResD}_i = \sqrt{\boldsymbol{r}_i^T \hat{\boldsymbol{\Sigma}}_\varepsilon^{-1} \boldsymbol{r}_i}$$

with cut-off value $\sqrt{\chi^2_{q,0.975}}$.

## 4 Classification

Classification is concerned with separating different groups. Classification rules or discriminant rules aim at finding boundaries between several populations, which then allow to classify new samples. We assume that our data are split into $m$ different groups, with $n_j$ observations in group $j$ (and $n = \sum_j n_j$). The Bayes discriminant rule classifies an observation to the group $j$ for which the discriminant score $d_j^Q(\boldsymbol{x})$ is maximal, with

$$d_j^Q(\boldsymbol{x}) = -\frac{1}{2}\ln|\boldsymbol{\Sigma}_j| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j) + \ln(p_j). \tag{12}$$

Here, $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the center and (non-singular) covariance matrix of the $j$th population, and $p_j$ is its membership (or prior) probability. The resulting discriminant rule is quadratic in $\boldsymbol{x}$, from which the terminology 'quadratic discriminant analysis' is derived. If the covariance structure of all groups is equal, or $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$, the discriminant scores reduce to

$$d_j^L(\boldsymbol{x}) = \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \ln(p_j) \tag{13}$$

which is linear in $\boldsymbol{x}$.

Classical classification uses the group means and covariances in (12) and (13), whereas the membership probabilities are often estimated as $n_j/n$. In robust discriminant analysis [19] the MCD estimates of location and scatter of each group are substituted in (12). For the linear case, we can use the pooled MCD scatter estimates as an estimate for the overall $\boldsymbol{\Sigma}$. Finally, the membership probabilities can be robustly estimated as the relative frequency of *regular* observations in each group, which are those whose flag resulting from the MCD procedure is equal to one.

One also needs a tool to evaluate a classification rule, i.e. we need an estimate of the associated *probability of misclassification.* Our MATLAB function contains three procedures. The *apparent probability of misclassification* counts the (relative) frequencies of badly classified observations from the training set. But it is well known that this yields a too optimistic estimate as the same observations are used to determine and to evaluate the discriminant rule. With the *cross-validation* approach the classification rule is constructed by leaving out one observation at a time and then one sees whether this observation is correctly classified or not. Because it makes little sense to evaluate the discriminant rule on outlying observations, one could apply this procedure by leaving out the non-outliers one by one, and counting the percentage of badly classified ones. This approach is rather time-consuming, especially at larger data sets, but approximate methods can be used [18] and will become available in the toolbox. As a third and fast approach, the proportion of misclassified observations from the *validation set* can be computed. Because it can happen that this validation set also contains outlying observations which should not be taken into account, we estimate the misclassification probability $\mathrm{MP}_j$ of group $j$ by the proportion of non-outliers from the validation set that belong to group $j$ and that are badly classified.

**Example 3:** The Fruit data set analyzed in [19] was obtained by personal communication with Colin Greensill (Faculty of Engineering and Physical Systems, Central Queensland University, Rockhampton, Australia) and resumed here. The original data set consists of 2818 spectra measured at 256 wavelengths. It contains 6 different cultivars of a cantaloupe. We proceed in the same way as in [19] where only 3 cultivars (D, M, and HA) were considered, leaving 1096 spectra. Next a robust principal component analysis was applied to reduce the dimension of the data. It would be dangerous to apply classical PCA here as the first components could be highly attracted by possible outliers and would not give a good low-dimensional representation of the data. (Also in [20] it was illustrated that preprocessing high dimensional spectra with a robust PCA method resulted in a better classification.) For illustrative purposes two components were retained and a linear discriminant analysis was performed. To perform the classification, the data were randomly split up into a training set and a validation set, consisting of 60% resp. 40% of the observations. It was told that the third cultivar HA consists of 3 groups obtained with different illumination systems. However, the subgroups were treated as a whole since it was not clear whether they would behave differently. Given this information the optional input argument 'alpha' which measures the fraction of outliers the MCD-algorithm should resist, was set to 0.5. The MATLAB call to *rda* additionally includes the training set $X$, the corresponding vector with group numbers $g$, the validation set $Xtest$ and its group labels $gtest$, and the method used ('linear' or 'quadratic'):

```
>> out=rda(X,g,'valid',Xtest,'groupvalid',gtest,'alpha',0.5,'method','linear')
```

For the fruit data, we obtain $\hat{p}_D = 51\%, \hat{p}_M = 13\%$ and $\hat{p}_{HA} = 36\%$ whereas the misclassification probabilities were 16% in cultivar D, 95% in cultivar M, and 6% in cultivar HA. Cultivar M's high misclassification probability is due to its overlap with the other groups and because of its rather small number of samples ($n_M = 106$) in contrast to the other 2 groups ($n_D = 490$ and $n_{HA} = 500$). The misclassification probability of HA on the other hand is very small because its estimate does not take into account the outlying data points. In Figure 6 a large group of outliers is spotted on the right side. The 97.5% tolerance ellipses are clearly not affected by that group of anomalous data points. In fact, the outlying group coincides with a subgroup of the cultivar HA which was caused by the change in the illumination system.

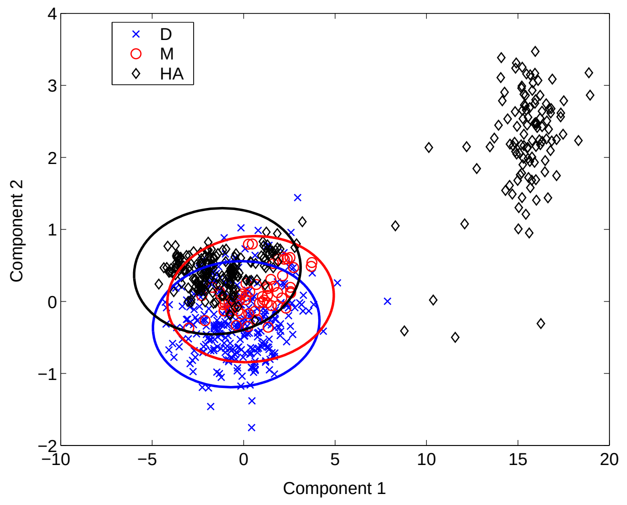


Figure 6: Analysis of the Fruit data set: 97.5% robust tolerance ellipses for each cultivar obtained from a linear discriminant rule.

Note that when data are high-dimensional, the approach of this section can not be applied anymore because the MCD becomes uncomputable. In the example of the fruit data, this was solved by applying a dimension reduction procedure (PCA) on the *whole* set of observations.

Instead, one can also apply a PCA method on *each group* separately. This is the idea behind the SIMCA method (Soft Independent Modeling of Class Analogy) [21]. A robust variant of SIMCA can be obtained by applying a robust PCA method, such as ROBPCA (Section 2.2), within each group. This approach is currently under development [22].

## 5 Availability and outlook

LIBRA, the MATLAB library for Robust Analysis, can be downloaded from one of the following web sites:

http://www.wis.kuleuven.ac.be/stat/robust.html
http://www.agoras.ua.ac.be/

These programs are provided free for non-commercial use only. They can be used with MATLAB 5.2, 6.1 and 6.5. Some of the functions require the MATLAB Statistics toolbox. The main functions currently included in the toolbox are listed in Table **??**. In the near future, we will add implementations of fast methods for cross-validation [18], S-estimators of location and covariance [23], S-estimators of regression [24], the LTS-subspace estimator [25], an adjusted boxplot for skewed distributions [26], and robust SIMCA [22].

### Acknowledgments

We acknowledge (in alphabetical order) Guy Brys, Sanne Engelen, Peter Rousseeuw, Nele Smets, Karlien Vanden Branden, and Katrien Van Driessen for their important contributions to the toolbox.

## References

[1] P.J. Huber, Robust Statistics, Wiley: New York, 1981.

[2] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, W.A. Stahel, Robust Statistics: The Approach Based on Influence Functions, Wiley: New York, 1986.

[3] P.J. Rousseeuw, A. Leroy, Robust Regression and Outlier Detection, John Wiley: New York, 1987.

[4] C. Chen, Robust Regression and Outlier Detection with the ROBUSTREG Procedure in Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, SAS Institute Inc, Cary, NC, 2002.

[5] A. Marazzi, Algorithms, Routines and S Functions for Robust Statistics, Belmont: Wadsworth, 1993.

[6] S-PLUS 6 Robust Library User's Guide, Insightfull Corporation, Seattle, WA.

[7] P.J. Rousseeuw, S. Verboven, Comput. Statist. Data Anal. 40 (2002) 741 – 758.

[8] G. Brys, M. Hubert, A. Struyf, J. Comput. Graph. Statist., in press, available at http://www.wis.kuleuven.ac.be/stat/robust.html.

[9] P.J. Rousseeuw, J. Amer. Statist. Assoc. 79 (1984) 891 – 880.

[10] P.J. Rousseeuw, K. Van Driessen, Technometrics 41 (1999) 212 – 223.

[11] M. Hubert, P.J. Rousseeuw, S. Verboven, Chemom. Intell. Lab. Syst. 60 (2002) 101 – 111.

[12] M. Hubert, P.J. Rousseeuw, K. Vanden Branden, Technometrics, in press, available at http://www.wis.kuleuven.ac.be/stat/robust.html.

[13] K.H. Esbensen, Multivariate Data Analysis in Practice, Camo Process AS (5th edition), 2001.

[14] P.J. Rousseeuw, K. Van Driessen, in W. Gaul, O. Opitz, and M. Schader (Eds.), Data Analysis: Scientific Modeling and Practical Application, Springer-Verlag, New York, 2000, pp. 335 – 346.

[15] P.J. Rousseeuw, S. Van Aelst, K. Van Driessen, A. Agullo, Technometrics, in press, available at http://www.agoras.ua.ac.be.

[16] M. Hubert, S. Verboven, J. Chemom. 17 (2003) 438 – 452.

[17] M. Hubert, K. Vanden Branden, J. Chemom. 17 (2003) 537 – 549.

[18] S. Engelen, M. Hubert, Fast cross-validation in robust PCA, COMPSTAT 2004 Proceedings, edited by J. Antoch, Springer, Physica-Verlag.

[19] M. Hubert, K. Van Driessen, Comput. Statist. Data Anal. 45 (2004) 301 − 320.

[20] M. Hubert, S. Engelen, Bioinformatics, 20 (2004), 1728-1736.

[21] S. Wold, Pattern Recognition 8 (1976) 127 − 139.

[22] K. Vanden Branden, M. Hubert, Robust classification of high-dimensional data, COMPSTAT 2004 Proceedings, edited by J. Antoch, Springer, Physica-Verlag.

[23] L. Davies, Ann. Statist. 15 (1987) 1269–1292.

[24] P.J. Rousseeuw, V.J. Yohai, in J. Franke, W. Härdle and R.D. Martin (Eds.), Robust and Nonlinear Time Series Analysis, Lecture Notes in Statistics No. 26. Springer-Verlag: New York, 1984, pp. 256 − 272.

[25] S. Engelen, M. Hubert, K. Vanden Branden, A comparison of three procedures for robust PCA in high dimensions, Proceedings of CDAM2004, BSU Publishing House.

[26] E. Vandervieren, M. Hubert, An adjusted boxplot for skewed distributions, COMPSTAT 2004 Proceedings, edited by J. Antoch, Springer, Physica-Verlag.

| **Function** | **Description** |
|---|---|
| **Robust estimators of location/scale/skewness** | |
| mlochuber | M-estimator of location, with Huber psi-function. |
| mloclogist | M-estimator of location, with logistic psi-function. |
| hl | Hodges-Lehmann location estimator. |
| unimcd | MCD estimator of location and scale. |
| mad | Median absolute deviation with finite sample correction factor (scale). |
| mscalelogist | M-estimator of scale, with logistic psi-function. |
| qn | $Q_n$-estimator of scale. |
| adm | Average distance to the median (scale). |
| mc | Medcouple, a robust estimator of skewness. |
| **Robust multivariate analysis** | |
| l1median | $L_1$ median of multivariate location. |
| mcdcov | Minimum Covariance Determinant estimator of multivariate location and covariance. |
| rapca | Robust principal component analysis (based on projection pursuit). |
| robpca | Robust principal component analysis (based on projection pursuit and MCD estimation). |
| rda | Robust linear and quadratic discriminant analysis. |
| robstd | Columnwise robust standardization. |
| **Robust regression methods** | |
| ltsregres | Least Trimmed Squares regression. |
| mcdregres | Multivariate MCD regression. |
| rpcr | Robust principal component regression. |
| rsimpls | Robust partial least squares regression. |
| **Plot functions** | |
| makeplot | creates plots for the main statistical functions (including residual plots and outlier maps). |
| normqqplot | normal Quantile-Quantile plot. |
| **Classical multivariate analysis and regression** | |
| ols | Ordinary (multiple linear) least squares regression. |
| classSVD | Singular value decomposition if $n > p$. |
| kernelEVD | Singular value decomposition if $n < p$. |
| cpca | Classical principal component analysis. |
| mlr | Multiple linear regression (with one or several response variables). |
| cpcr | Classical principal component regression. |
| csimpls | Partial least squares regression (SIMPLS). |
| cda | Classical linear and quadratic discriminant analysis. |

Table 3: List of the current main functions in LIBRA, the MATLAB Library for Robust Analysis.