

A fast method for robust principal components with applications to chemometrics

Mia Hubert*, Peter J. Rousseeuw, and Sabine Verboven

Revised version: March 23, 2001

Department of Mathematics and Computer Science, University of Antwerp (UIA), Universiteitsplein 1, B-2610 Wilrijk, Belgium

Abstract

When faced with high-dimensional data, one often uses principal component analysis (PCA) for dimension reduction. Classical PCA constructs a set of uncorrelated variables, which correspond to eigenvectors of the sample covariance matrix. However, it is well-known that this covariance matrix is strongly affected by anomalous observations. It is therefore necessary to apply robust methods that are resistant to possible outliers.

Li and Chen [1] proposed a solution based on projection pursuit (PP). The idea is to search for the direction in which the projected observations have the largest robust scale. In subsequent steps, each new direction is constrained to be orthogonal to all previous directions. This method is very well suited for high-dimensional data, even when the number of variables p is higher than the number of observations n . However, the algorithm of Li and Chen has a high computational cost. In [2, 3] a computationally much more attractive method is presented, but in high dimensions (large p) it has a numerical accuracy problem and still consumes much computation time.

In this paper we construct a faster two-step algorithm that is more stable numerically. The new algorithm is illustrated on a data set with 4 dimensions and on two chemometrical data sets with 1200 and 600 dimensions.

*Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium).

1 Introduction

High-dimensional data are often hard to cope with. First of all, the computational effort to analyze such data (e.g. by cluster analysis) can become very high. Furthermore many statistical analyses, like multiple regression, are sensitive to the presence of correlated variables, also known as multicollinearity. To overcome these problems, principal component analysis (PCA) is often used to reduce the dimension by creating a new set of uncorrelated variables.

Classical PCA first rotates the data such that the new coordinate axes (i.e. the principal components) are the principal axes of the sample covariance matrix S , and then selects a few components that together explain a large proportion of the total variance. To do this, one computes and ranks the eigenvalues of S .

However, it is well-known that the sample covariance matrix is strongly affected by anomalous observations. As a result, the classical principal components may describe the shape of the majority of the data incorrectly (see e.g. [3, 4, 5]). It is therefore necessary to apply robust methods that are resistant to possible outliers.

A first approach is to robustify the covariance matrix (see e.g. [4]). In [6] it is shown that the MCD-estimator [7] and S-estimators of location and shape [8, 9] are well-suited for this purpose. Recently, this approach has also been used for robust factor analysis [10]. However, an important limitation of these methods is that they can only be applied if the number of observations n is larger than the number of variables p . It is equally important to have a robust PCA method for situations with $p > n$ which are often encountered in chemometrics.

Li and Chen [1] proposed a solution based on projection pursuit (PP). The idea is to search for the direction in which the projected observations have the largest robust scale. In subsequent steps, each new direction is then constrained to be orthogonal to all previous directions. However, the algorithm of Li and Chen has a high computational cost. In [2, 3] a computationally much more attractive method is presented. Unfortunately, we will see in Section 2 that the latter algorithm has a numerical accuracy problem in high dimensions (large p) and still consumes much computation time. In Section 3 we construct a modified version that is more stable numerically. In Section 4 we introduce a much faster two-step version, that is used to analyze several data sets in Section 5. A simulation study can be found in Section 6. Conclusions and an outlook for further research are presented in Section 7. Finally, the Appendix contains mathematical details of the algorithms.

Throughout this paper we will use the following notation. A matrix is denoted by a

capital letter, often with two subindices that indicate the dimensions of the matrix. For instance, $X_{n,p}$ stands for a matrix with n rows and p columns, where n is the number of objects and p is the number of original variables. A vector is represented by a boldface lowercase letter like \mathbf{v} . It is always assumed to be a column vector. A vector \mathbf{v} with p components is thus written as $\mathbf{v} = (v_1, \dots, v_p)^t$.

2 The algorithm of Croux and Ruiz-Gazen

Let $X_{n,p}$ be the original data matrix. The PP-algorithm of Croux and Ruiz-Gazen [2, 3], from now on called the C-R algorithm, proceeds as follows.

As an initial step, the data are centered around their L^1 -median $\hat{\boldsymbol{\mu}}^R$. This is a highly robust and orthogonally equivariant location estimator, also known as the spatial median. It is defined as the point $\boldsymbol{\theta}$ which minimizes the sum of distances to all observations, i.e.

$$\hat{\boldsymbol{\mu}}^R = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\theta}\|.$$

To compute $\hat{\boldsymbol{\mu}}^R$ one can use the algorithm of Hössjer and Croux [11]. Often an estimator is required to be affine equivariant, which ensures that the estimate will transform properly when the axes are rescaled and rotated. (For precise definitions see [8]). But here we only need orthogonal equivariance, since even the classical PCA procedure is only orthogonally equivariant.

To simplify notations, we denote from now the centered observations by $\mathbf{x}_i^{(1)} = \mathbf{x}_i - \hat{\boldsymbol{\mu}}^R$. Further we set

$$M^R = \mathbf{1}_n (\hat{\boldsymbol{\mu}}^R)^t$$

where $\mathbf{1}_n = (1, \dots, 1)^t$ is a column vector whose n components are all equal to 1. The rows of the matrix M^R are all identical, and equal to $(\hat{\boldsymbol{\mu}}^R)^t$. The centered data matrix $(\mathbf{x}_1^{(1)}, \dots, \mathbf{x}_n^{(1)})^t$ is thus equal to $X_{n,p} - M^R$.

Having preprocessed the data, the robust principal components are constructed. Let r be the rank of the centered data matrix, thus $r \leq \min(n-1, p)$. An r -dimensional subspace spanned by orthogonal unit vectors \mathbf{v}_l (for $l = 1, \dots, r$) is to be found, such that the robust scale s_l of the projected observations on \mathbf{v}_l is maximal. Analogously to classical PCA, we will call these directions \mathbf{v}_l eigenvectors, and the squared scales s_l^2 the corresponding eigenvalues. One first looks for the direction \mathbf{v}_1 such that the projection of the data on

\mathbf{v}_1 has maximal robust scale s_1 . Once \mathbf{v}_1 is found, the observations are projected onto the orthogonal complement of \mathbf{v}_1 . The next vector \mathbf{v}_2 must then belong to this orthogonal complement and the robust scale s_2 of the projected observations on \mathbf{v}_2 should again be maximal. This loop is continued until r directions are found.

It has been proved by Li and Chen [1] that the resulting method is highly robust, since it inherits the breakdown value of the robust scale estimator. The breakdown value tells us which percentage of data points may be corrupted while still yielding a bounded estimate [8, page 10]. Croux and Ruiz-Gazen [3] compared the efficiency and robustness of several robust scale estimators that can be used in this PP-algorithm, and concluded that the Q_n estimator [12] yielded the best results. This scale estimator is essentially the first quartile of all pairwise differences between two data points. To be precise, for any univariate data set (z_1, \dots, z_n) , the estimate is defined as

$$Q_n(z_1, \dots, z_n) = 2.2219 * c_n * \{|z_i - z_j|; i < j\}_{(k)} \quad (1)$$

with $k = \binom{h}{2} \approx \binom{n}{2}/4$ and $h = \lfloor \frac{n}{2} \rfloor + 1$. The constant c_n is a small-sample correction factor, which makes Q_n an unbiased estimator (note that c_n only depends on the sample size n , and that $c_n \rightarrow 1$ for increasing n). The breakdown value of Q_n is 50% whereas the statistical efficiency of the resulting eigenvectors at the gaussian distribution equals 67%, which is better than many other robust methods. Therefore we will use the Q_n estimator from now on.

To make the algorithm computationally feasible, Croux and Ruiz-Gazen have restricted the collection of directions to be investigated to all directions that pass through $\hat{\boldsymbol{\mu}}^R$ and a data point. A precise description of the algorithm can be found in the Appendix.

Although the C-R algorithm performs well in low dimensions, it lacks numerical stability in high dimensions, as will be shown in the following example.

[Figures 1 and 2 about here]

Example. Energy Dispersive X-Ray Fluorescence (ED-XRF) is a fast non-destructive multi-element analysis technique. We will analyze a data set obtained from Pascal Lemberge containing 50 XRF spectra of aqueous solutions of metal salts at 1200 channels. The original spectra are shown in Figure 1. This XRF data set is then \log_{10} -preprocessed and mean-centered to lower the influence of the high background at the last channels (800 to 1200). In

Figure 2 we see that the robust scales coming out of the C-R algorithm first decrease, but from the 28th onwards they increase again.

3 An improved algorithm

The reason for this numerical instability is that all computations are performed in the high-dimensional space of dimension $p = 1200$. The succession of projections leads to an accumulation of round-off errors, and finally yields unreliable estimates. Therefore we propose an improvement of this algorithm through dimension reduction. Note that the basic principles of the C-R algorithm (the centering by the L^1 -median, the stepwise search for orthogonal directions \mathbf{v}_l , the use of the Q_n -estimator, and the search in the direction of the data points) remain the same, and will therefore not be described again.

Suppose we have found the first direction \mathbf{v}_1 . We now apply an orthogonal data transformation U_1 such that \mathbf{v}_1 is mapped onto the first basis vector $\mathbf{e}_1 = (1, 0, \dots, 0)^t$. It turns out that the fastest way to do this is not by constructing a large orthogonal matrix but by means of a reflection. To construct this reflection U_1 we only need to compute the unit normal vector $\mathbf{n}_1 = (\mathbf{e}_1 - \mathbf{v}_1) / \|\mathbf{e}_1 - \mathbf{v}_1\|$. Each centered data point $\mathbf{x}_i^{(1)}$ is then transformed to

$$\mathbf{x}_i^{(2)} = U_1(\mathbf{x}_i^{(1)}) = \mathbf{x}_i^{(1)} - 2\langle \mathbf{x}_i^{(1)}, \mathbf{n}_1 \rangle \mathbf{n}_1 \quad (2)$$

and thus $U_1(\mathbf{v}_1) = \mathbf{e}_1$. Here $\langle \mathbf{a}, \mathbf{b} \rangle$ stands for the inner product of two vectors \mathbf{a} and \mathbf{b} , which is computed as $\mathbf{a}^t \mathbf{b}$. Figure 3 illustrates the reflection U_1 in three dimensions.

[Figure 3 about here]

If we now want to project the data on the orthogonal complement of $U_1(\mathbf{v}_1)$ we simply have to remove the first coordinate of $\mathbf{x}_i^{(2)}$ for every i . This reduces our space by one dimension. Moreover, all operations so far are numerically accurate. The reflection (2) only requires elementary sums, and the projection itself needs no auxiliary computations.

This procedure is repeated until $r = \text{rank}(X_{n,p} - M^R)$ directions have been found. At each step, we look for a new direction \mathbf{v}_l in the reduced $(p - l + 1)$ -dimensional space. Then this vector \mathbf{v}_l is reflected onto the first basis vector of that space, and again the projections of the reflected data are obtained immediately by omitting their first coordinate.

In order to interpret the results, we of course have to transform each direction \mathbf{v}_l back to the original p -dimensional space. Note that this operation is again numerically stable since the inverse of a reflection is the reflection itself, so no matrix inversions are needed. We will refer to this as the R-step approach, where R stands for *reflection*. The details can be found in the Appendix.

In Figure 2 we see that the robust scales s_l obtained by the R-step are now monotone decreasing.

If we denote by $P_{p,r}$ the matrix with the backtransformed eigenvectors as columns, we obtain a robust decomposition of $X_{n,p}$ into an $(n \times r)$ score matrix $T_{n,r}$ and a $(p \times r)$ matrix of loadings $P_{p,r}$. Indeed, putting $T_{n,r} = (X_{n,p} - M^R)P_{p,r}$ yields

$$X_{n,p} - M^R = T_{n,r}P_{p,r}^t. \quad (3)$$

Note that $T_{n,r}$ contains the coordinates of the data points in the space spanned by the principal components.

Finally, we can reduce the original number of variables p by considering only the first k columns of P . The number k thus stands for the number of principal components we want to retain. This type of algorithm benefits most from that situation, since it only needs to be run until \mathbf{v}_k is found instead of computing all r vectors $\mathbf{v}_1, \dots, \mathbf{v}_r$. This saves a lot of computation time.

Remark. Often a PCA analysis starts by prestandardizing the data in order to obtain variables that all have the same spread. Otherwise, the variables with a large variance compared to the others will dominate the first principal components. Standardizing by the mean and the standard deviation of each variable yields a PCA analysis based on the correlation matrix instead of the covariance matrix. We can also standardize each variable l in a robust way, by first subtracting its median $med(x_{1l}, \dots, x_{nl})$ and then dividing by its robust scale estimate $Q_n(x_{1l}, \dots, x_{nl})$.

4 A faster two-step algorithm

The R-step approach is numerically more stable than the C-R implementation, but it takes more computation time. Table 1 shows the computation time (in seconds on a 166Mhz PC

running MATLAB) of the C-R algorithm and the R-step approach applied to the XRF data set of Section 2, with $n = 50$ and $p = 1200$. The number k indicates the number of principal components that are actually computed. We see that the R-step approach is roughly 10 times slower than the C-R algorithm, so we would like to speed it up.

[Table 1 about here]

Since the high computation time is caused by the large value of p , we propose the following two-step algorithm. First, we reduce the data space to the affine subspace spanned by the n observations. This can be done fast and accurately by classical PCA (e.g. using the kernel version of the eigenvalue decomposition [13]), without losing any information contained in the data. We thus obtain scores $\tilde{T}_{n,r}$ that satisfy

$$(X_{n,p} - M^C)\tilde{P}_{p,r} = \tilde{T}_{n,r} \quad (4)$$

where now $\hat{\boldsymbol{\mu}}^C$ is the classical mean vector, $M^C = \mathbf{1}_n(\hat{\boldsymbol{\mu}}^C)^t$ and $r = \text{rank}(X_{n,p} - M^C) \leq n - 1$. This step is useful as soon as $p > r$. When $p \gg n$ we obtain a huge reduction. For the XRF data set, this reduces the 1200-dimensional original data set to one in only 49 dimensions.

Secondly, the R-step is performed on the scores matrix $\tilde{T}_{n,r}$. According to (3) we obtain the decomposition

$$(\tilde{T}_{n,r} - M_T^R)\tilde{\tilde{P}}_{r,k} = T_{n,k} \quad (5)$$

with $k \leq r$ the number of latent variables one wants to compute, and M_T^R the robust centering matrix of $\tilde{T}_{n,r}$. Using equations (4) and (5) and the orthogonal equivariance of the L^1 -median we finally obtain

$$T_{n,k} = (X_{n,p} - M_T^R)P_{p,k} \quad (6)$$

with $P_{p,k} = \tilde{P}_{p,r}\tilde{\tilde{P}}_{r,k}^t$ and $M^R = M^C + M_T^R\tilde{P}_{r,p}^t$. Also note that the columns of $P_{p,k}$ are still orthogonal, since the matrix multiplication $P_{p,k} = \tilde{P}_{p,r}\tilde{\tilde{P}}_{r,k}$ preserves the orthogonality. And since the R-step approach is orthogonally equivariant, the two-step algorithm yields the same eigenvectors (and thus also the same scores) as does the R-step approach.

The two-step algorithm will be denoted by RAPCA, which stands for *reflection-based algorithm for principal components analysis*.

We see in Table 1 that RAPCA is a much faster version of the R-step approach. Also the speed improvement with regard to C-R is substantial. We see that the RAPCA time

is almost constant in k , whereas the computation time of the other algorithms increases strongly when more latent variables are required.

We have implemented the RAPCA algorithm in the MATLAB environment. The code can be obtained from the authors by sending email to `sabine.verboven@ua.ac.be`.

5 Examples

5.1 Near-Infrared-Reflectance (NIR) Spectra of Biscuit Dough

We consider the well-known chemometrics data set of Osborne et al. [14]. It contains 40 NIR spectra of biscuit dough with measurements every 2 nanometers, from 1200nm up to 2400nm. The data are first scaled using a logarithmic transformation in order to eliminate drift and background scatter. Originally the dataset consisted of 700 variables, but the ends were discarded because of the lower instrumental reliability. Following Marx and Eilers [15] we used the first differences to remove constants and sudden shifts. But contrary to previous analyses, we will leave the observation 23 (which is usually considered as an outlier on other grounds) in the data set. After this preprocessing we end up with a data set of 40 observations in 600 dimensions, shown in Figure 4, to which we applied the RAPCA algorithm.

[Figure 4 about here]

Figure 5 shows the scree plot of the robust eigenvalues versus their index. We see that the robust eigenvalues decrease nicely. Often a scree plot helps us to decide how many principal components should be preserved. We should look for the 'elbow', the first substantial kink in the line [16, page 45]. Here the first twist is observed around 5, so we will retain $k = 6$ principal components.

[Figure 5 about here]

Next, we would like to identify the outliers in our data set. For this we will use the distance-distance plot of Rousseeuw and Van Driessen [17] that shows for each data point its robust distance versus its classical distance. Denote by (T^C, s_j^C) and (T^R, s_j^R) the scores matrix and the square root of the j th eigenvalue as obtained by the classical PCA and the

robust RAPCA method. Then for each data point i its classical distance (CD_i) and its robust distance (RD_i) are given by

$$CD_i = \sqrt{\sum_{j=1}^k \left(\frac{t_{ij}^C}{s_j^C}\right)^2} \quad \text{and} \quad RD_i = \sqrt{\sum_{j=1}^k \left(\frac{t_{ij}^R}{s_j^R}\right)^2}.$$

Here k stands for the selected number of components, which is 6 in our example. Observations whose distance exceeds the cut-off value $\sqrt{\chi_{k,0.975}^2}$ are pinpointed as outliers. In Figure 6 we see that a few samples have robust distances above the cutoff value, where 19, 20 and 7 can be seen as borderline cases, 24 is outlying, and 23 is a far outlier. In contrast, the distances based on the classical PCA hardly detect observation 23.

[Figure 6 about here]

5.2 Hawkins-Bradu-Kass data set

The Hawkins-Bradu-Kass data set [8, page 94] is an artificial data set that is often used to illustrate robust regression techniques. It consists of 75 observations in 4 dimensions that satisfy the linear regression model

$$y_i = \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \theta_4 + e_i \quad \text{for all } i = 1, \dots, 75 \quad (7)$$

where e_i denotes the noise of observation i . The response variable Y can thus be quite well described as a linear combination of the three explanatory variables X_1, X_2 and X_3 plus an intercept term. However, it is known that this data set contains several outliers. The first 10 observations have outlying \mathbf{x}_i and y_i values, whereas the next 4 points have only outlying \mathbf{x}_i values. These 14 data points are also called the 'bad' points, as they do not follow the linear model (7). On the other hand all regular observations are labeled as being 'good'.

[Figure 7 about here]

Figure 7(a) shows the scores (t_{i1}^C, t_{i2}^C) with respect to the first two components of the classical PCA applied to all four variables X_1, X_2, X_3 , and Y . We have also drawn the 97.5% tolerance ellipse given by the equation

$$\left(\frac{t_{i1}^C}{s_1^C}\right)^2 + \left(\frac{t_{i2}^C}{s_2^C}\right)^2 = \chi_{2,0.975}^2$$

where $\chi_{2,0.975}^2 = 7.38$. Figure 7(b) analogously shows the scores $(t_{i_1}^R, t_{i_2}^R)$ obtained by RAPCA, with the corresponding tolerance ellipsoid given by $(t_{i_1}^R/s_1^R)^2 + (t_{i_2}^R/s_2^R)^2 = 7.38$. Observations outside such a 97.5% tolerance ellipse can be flagged as outliers. We see that classical PCA only detects the smallest group of 4 outliers, whereas RAPCA clearly separates the good points from the bad ones. Moreover, we see that the classical PCA does not center the data points correctly, because the classical mean of these data points is outside the cloud of the good observations. On the other hand, the robust center in Figure 7(b) corresponds nicely to the middle of the main group.

Next, we have performed the classical PCA on the reduced data set obtained by removing the outlying observations 1-14. Figure 7(c) shows the corresponding scores. Now we see that the mean of the regular data is close to the origin and that all outliers are identified. This shows that the RAPCA analysis is comparable with the classical PCA based on the good data. This can also be seen in Table 2 which displays the eigenvalues of the three different analyses: classical PCA and RAPCA on the whole data set (PCA_{full} and $RAPCA_{full}$), and classical PCA on the reduced data set (PCA_{red}). Based on these results, we would decide to retain three principal components with $RAPCA_{full}$ and PCA_{red} , whereas PCA_{full} would keep only a single component.

[Table 2 about here]

From Table 2 we also see that the eigenvalues computed with $RAPCA_{full}$ are larger than those obtained with PCA_{red} . This is because the outliers are more dispersed than the majority of the data, which increases Q_n at the full data set. However, if we apply RAPCA to the reduced (uncontaminated) data set ($RAPCA_{red}$) we see in the last column of Table 2 that now the eigenvalues are much closer to those obtained with PCA_{red} . Note that some other robust methods, as e.g. the PROP method [18] result in eigenvalues which are in close to complete agreement in this example, for both the full and the reduced data set.

6 Simulation study

In order to compare RAPCA with classical PCA we have conducted a small simulation study. We have generated 100 samples of size 100 from the contamination model

$$(1 - \varepsilon)N_p(\mathbf{0}, \Sigma) + \varepsilon N_p(\tilde{\boldsymbol{\mu}}, \tilde{\Sigma})$$

for different values of $p, \varepsilon, \Sigma, \tilde{\boldsymbol{\mu}}$ and $\tilde{\Sigma}$. That is, $100(1 - \varepsilon)$ of the observations were generated from the p -variate gaussian distribution $N_p(\mathbf{0}, \Sigma)$ and 100ε observations were generated from $N_p(\tilde{\boldsymbol{\mu}}, \tilde{\Sigma})$. In Tables 3 and 4 we report the results for the following situations:

1. $p = 4, \Sigma = \text{diag}(4, 3, 2, 1)$
 - (a) sim1: $\varepsilon = 0$ (no contamination).
 - (b) sim2: $\varepsilon = 10\%, \tilde{\boldsymbol{\mu}} = \mathbf{0}, \tilde{\Sigma} = 9\Sigma$ (outliers that may increase the eigenvalues of the covariance matrix).
 - (c) sim3: $\varepsilon = 10\%, \tilde{\boldsymbol{\mu}} = (0, 0, 0, 10)^t, \tilde{\Sigma} = \Sigma$ (outliers that may change the direction of the eigenvectors).
2. $p = 20, \Sigma = \text{diag}(5, 4, 3, 2, 1, 0.15, 0.14, \dots, 0.02, 0.01)$
 - (a) sim4: $\varepsilon = 0$ (no contamination).
 - (b) sim5: $\varepsilon = 10\%, \tilde{\boldsymbol{\mu}} = 10\mathbf{e}_6, \tilde{\Sigma} = \Sigma$ (e.e. contamination in the direction of an eigenvector that corresponds with a very small eigenvalue).
 - (c) sim6: $\varepsilon = 10\%, \tilde{\boldsymbol{\mu}} = 10\mathbf{e}_5, \tilde{\Sigma} = \frac{1}{20}\Sigma$ (i.e. concentrated contamination in the direction of a medium-sized eigenvalue).

For each simulation we summarize the estimation procedure (PCA or RAPCA) as follows:

- for each eigenvector we consider the mean (over the 100 replications) of the angle (in radians) between the true eigenvector and the estimated one. The ideal value is thus 0.
- for each eigenvalue we report its mean value, and its mean squared error (MSE) defined as

$$\text{MSE}(\hat{\lambda}_i) = \frac{1}{100} \sum_{j=1}^{100} (\lambda_i - \hat{\lambda}_i^{(j)})^2$$

where $\hat{\lambda}_i^{(j)}$ is the estimated value of λ_i at the j th replication.

Table 3 reports the results for the three simulations in four dimensions, whereas Table 4 lists the simulation results for the five largest eigenvalues and corresponding eigenvectors of the three simulations in $p = 20$ dimensions. We see that PCA performs somewhat better than RAPCA when there are no outliers, but that it yields unreliable estimates in the presence of contamination.

[Tables 3 and 4 about here]

7 Conclusions and outlook

In this paper we have constructed the RAPCA method for robust principal components. It can deal with situations where there are more variables than objects, and combines numerical accuracy with computation speed. In a data set with outliers, RAPCA is able to detect the latent variables that explain the variability of the good data points, whereas these latent variables are often badly estimated by classical PCA.

In addition to robust principal components and scores, RAPCA also induces a robust covariance matrix S defined as

$$S = \sum_{l=1}^r s_l^2 \mathbf{v}_l \mathbf{v}_l^t.$$

Since S is defined by its spectral decomposition, we see immediately that the set $(\mathbf{v}_l)_l$ corresponds with the eigenvectors of S , and $(s_l^2)_l$ with their eigenvalues. We are currently investigating the performance of this robust covariance matrix when used in calibration methods like PLS or PCR.

Acknowledgements. We are grateful to Pascal Lemberge and Paul Eilers for providing the XRF and the Biscuit data sets, and to Sijmen de Jong for constructive remarks which improved the presentation.

8 Appendix

8.1 Flowchart of the C-R algorithm

The C-R algorithm computes the eigenvectors and eigenvalues as follows. Recall that $\hat{\boldsymbol{\mu}}^R$ is the L^1 -median of the original data points and that $\mathbf{x}_i^{(1)} = \mathbf{x}_i - \hat{\boldsymbol{\mu}}^R$ for all $i = 1, \dots, n$.

1. The first eigenvector is given by

$$\mathbf{v}_1 = \operatorname{argmax}_{\mathbf{a} \in A_1} Q_n(\mathbf{a}^t \mathbf{x}_1^{(1)}, \mathbf{a}^t \mathbf{x}_2^{(1)}, \dots, \mathbf{a}^t \mathbf{x}_n^{(1)})$$

with $A_1 = \{\mathbf{x}_i^{(1)} / \|\mathbf{x}_i^{(1)}\|; i = 1, \dots, n\}$.

2. To guarantee orthogonality, the data points are now projected on the orthogonal complement of \mathbf{v}_1 , i.e.

$$\mathbf{x}_2^{(2)} = (I_{p,p} - \mathbf{v}_1 \mathbf{v}_1^t) \mathbf{x}_i^{(1)}$$

with $I_{p,p}$ the p -dimensional identity matrix.

3. Steps 1 and 2 are repeated until $k \leq r = \text{rank}(X_{n,p} - M^R) \leq \min(n-1, p)$ eigenvectors are found. Suppose that $\mathbf{v}_1, \dots, \mathbf{v}_{l-1}$ have been constructed, then

$$\mathbf{v}_l = \underset{\mathbf{a} \in A_l}{\text{argmax}} \quad Q_n(\mathbf{a}^t \mathbf{x}_1^{(l)}, \mathbf{a}^t \mathbf{x}_2^{(l)}, \dots, \mathbf{a}^t \mathbf{x}_n^{(l)})$$

with $A_l = \{\mathbf{x}_i^{(l)} / \|\mathbf{x}_i^{(l)}\|; i = 1, \dots, n\}$. Moreover,

$$\mathbf{x}_i^{(l+1)} = (I_{p,p} - \sum_{j=1}^l \mathbf{v}_j \mathbf{v}_j^t) \mathbf{x}_i^{(1)} = \mathbf{x}_i^{(l)} - \mathbf{v}_l \mathbf{v}_l^t \mathbf{x}_i^{(1)}$$

Finally, for all $l = 1, \dots, r$ the robust scale s_l^R is defined as

$$s_l^R = Q_n(\mathbf{v}_l^t \mathbf{x}_1^{(l)}, \mathbf{v}_l^t \mathbf{x}_2^{(l)}, \dots, \mathbf{v}_l^t \mathbf{x}_n^{(l)}).$$

Note that for all $l = 1, \dots, r$ and all $i = 1, \dots, n$ the point $\mathbf{x}_i^{(l)}$ belongs to \mathbb{R}^p .

8.2 Flowchart of the R-step approach

1. The first eigenvector \mathbf{v}_1 is constructed as in the C-R algorithm.
2. Now the data are transformed by means of a reflection U_1 such that

$$U_1(\mathbf{v}_1) = \mathbf{e}_1 = (1, 0, \dots, 0)^t \in \mathbb{R}^p$$

i.e. let

$$\mathbf{n}_1 = \frac{\mathbf{e}_1 - \mathbf{v}_1}{\|\mathbf{e}_1 - \mathbf{v}_1\|}$$

and put

$$\begin{aligned} \mathbf{x}_i^{(2)} &= U_1(\mathbf{x}_i^{(1)}) \\ &= \mathbf{x}_i^{(1)} - 2\langle \mathbf{x}_i^{(1)}, \mathbf{n}_1 \rangle \mathbf{n}_1. \end{aligned}$$

3. The data points $\mathbf{x}_i^{(1)}$ are projected onto the orthogonal complement of $U_1(\mathbf{v}_1)$ by omitting their first coordinate. This yields

$$\tilde{\mathbf{x}}_i^{(2)} = (x_{i2}^{(2)}, \dots, x_{ip}^{(2)})^t \in \mathbb{R}^{p-1}.$$

4. The second eigenvector $\tilde{\mathbf{v}}_2$ is now found in the reduced space \mathbb{R}^{p-1} according to the C-R algorithm. This means that

$$\tilde{\mathbf{v}}_2 = \operatorname{argmax}_{\mathbf{a} \in A_2} Q_n(\mathbf{a}^t \tilde{\mathbf{x}}_1^{(2)}, \mathbf{a}^t \tilde{\mathbf{x}}_2^{(2)}, \dots, \mathbf{a}^t \tilde{\mathbf{x}}_n^{(2)})$$

with $A_2 = \left\{ \tilde{\mathbf{x}}_i^{(2)} / \|\tilde{\mathbf{x}}_i^{(2)}\|; i = 1, \dots, n \right\}$. Then $\tilde{\mathbf{v}}_2$ is backtransformed to \mathbb{R}^p using the inverse reflection.

5. Steps 2-4 are repeated until k eigenvectors are found.

Suppose that $\mathbf{v}_1, \dots, \mathbf{v}_{l-1}$ have been constructed. Then the reflection U_{l-1} and the transformed and the projected data points are defined as

$$\begin{aligned} U_{l-1}(\tilde{\mathbf{v}}_{l-1}) &= \tilde{\mathbf{e}}_{l-1} = (1, 0, \dots, 0)^t \in \mathbb{R}^{p-l+2} \\ \mathbf{x}_i^{(l)} &= U_{l-1}(\tilde{\mathbf{x}}_i^{(l-1)}) \in \mathbb{R}^{p-l+2} \\ \tilde{\mathbf{x}}_i^{(l)} &= \left(x_{i,2}^{(l)}, \dots, x_{i,p-l+2}^{(l)} \right) \in \mathbb{R}^{p-l+1}. \end{aligned}$$

Finally

$$\tilde{\mathbf{v}}_l = \operatorname{argmax}_{\mathbf{a} \in A_l} Q_n(\mathbf{a}^t \tilde{\mathbf{x}}_1^{(l)}, \mathbf{a}^t \tilde{\mathbf{x}}_2^{(l)}, \dots, \mathbf{a}^t \tilde{\mathbf{x}}_n^{(l)})$$

with $A_l = \left\{ \tilde{\mathbf{x}}_i^{(l)} / \|\tilde{\mathbf{x}}_i^{(l)}\|; i = 1, \dots, n \right\}$. The eigenvector \mathbf{v}_l is again obtained from $\tilde{\mathbf{v}}_l$ by backtransformation.

References

- [1] G. Li and Z. Chen, J. Amer. Statist. Assoc., 80 (1985) 759-766.
- [2] C. Croux and A. Ruiz-Gazen, in COMPSTAT: Proceedings in Computational Statistics 1996, Physica-Verlag, Heidelberg, 1996, pp. 211-217.
- [3] C. Croux and A. Ruiz-Gazen, High Breakdown Estimators for Principal Components: the Projection-Pursuit Approach Revisited, 2000, submitted for publication.
- [4] S. Devlin, R. Gnanadesikan and J. Kettenring, J., Biometrika, 62 (1975) 531-545.
- [5] N. Locantore, J.S. Marron, D.G. Simpson, N. Tripoli, J.T. Zhang, and K.L. Cohen, Test, 8 (1999) 1-73.

- [6] C. Croux and G. Haesbroeck, *Biometrika*, 87 (2000) 603-618.
- [7] P.J. Rousseeuw, (1984). *J. Amer. Statist. Assoc.*, 79 (1984) 871-880.
- [8] P.J. Rousseeuw and A. Leroy, *Robust Regression and Outlier Detection*, John Wiley, New York, 1987.
- [9] L. Davies, *Ann. Statist.*, 15 (1987) 1269-1292.
- [10] G. Pison, P.J. Rousseeuw, P. Filzmoser, and C. Croux, *Robust Factor Analysis*, to appear in *Journal of Multivariate Analysis* (2002).
- [11] O. Hössjer and C. Croux, *Nonparam. Statist.*, 4 (1995) 293-308.
- [12] P.J. Rousseeuw and C. Croux, *J. Amer. Statist. Assoc.*, 88 (1993) 1273-1283.
- [13] W. Wu, D.L. Massart and S. de Jong, *Chemom. Intell. Lab. Syst* 36 (1997) 165-172.
- [14] B.G. Osborne, T. Fearn, A.R. Miller, and S. Douglas, (1984). *J. Scient. Food Agric.*, 35 (1984) 99-105.
- [15] B.D. Marx, and P.H. Eilers, *Technometrics* 41 (1999) 1-13.
- [16] J.E. Jackson, *A User's Guide to Principal Components*, Wiley Series in Probability and Mathematical Statistics, New York, 1991, pp. 45-46.
- [17] P.J. Rousseeuw and K. Van Driessen, *Technometrics* 41 (1999) 212-223.
- [18] A. Singh and J.M. Nocerino, in J. Einax (Ed.), *Handbook of Environmental Chemistry - Statistical Methods*, Vol 2-G, Springer-Verlag, New York, 1995, pp. 229-277.

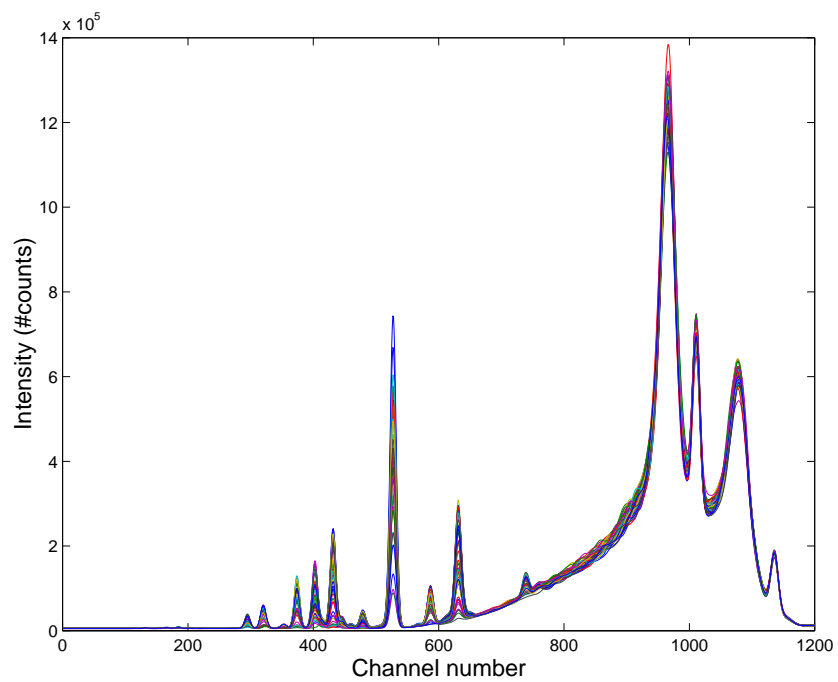


Figure 1: The XRF data set containing 50 XRF spectra of aqueous solutions of metal salts at 1200 channels.

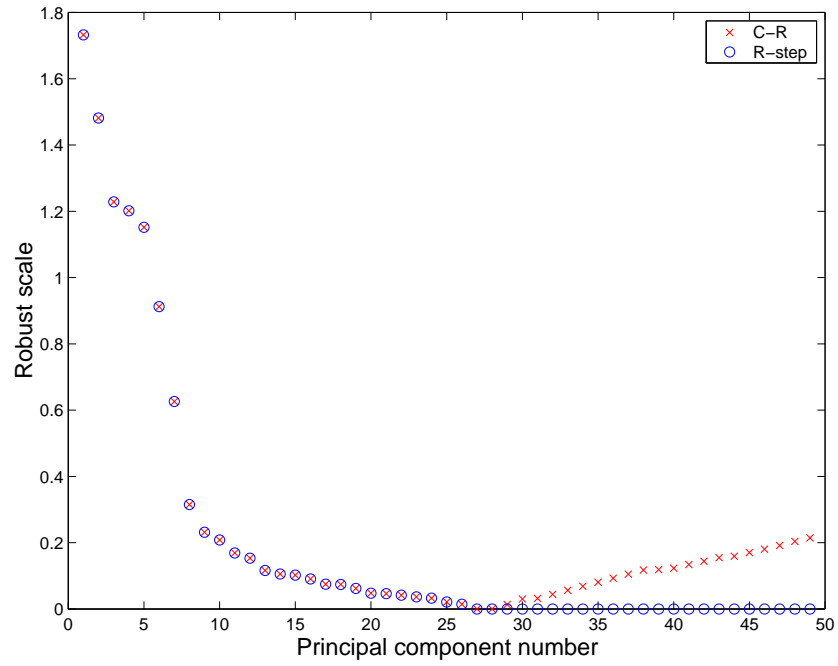


Figure 2: The robust scales of the XRF data set computed with the C-R (x) and the R-step approach (o).

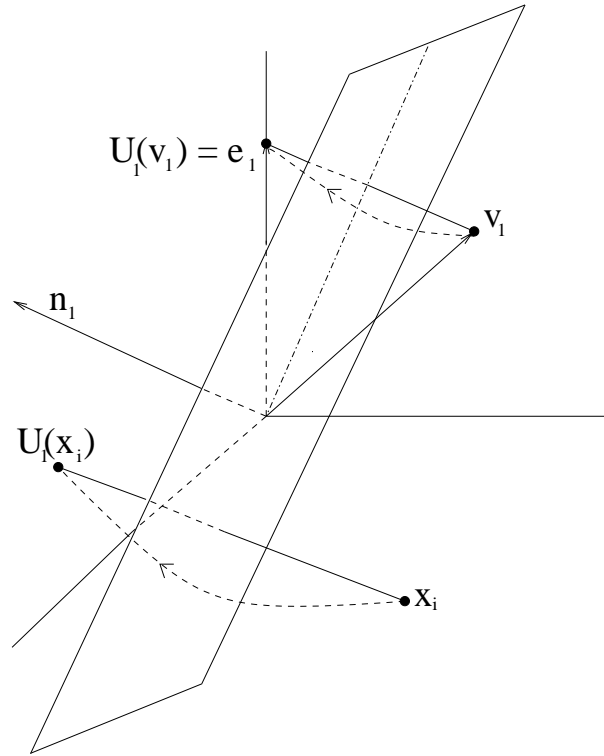


Figure 3: An illustration of the reflection U_1 in three dimensions.

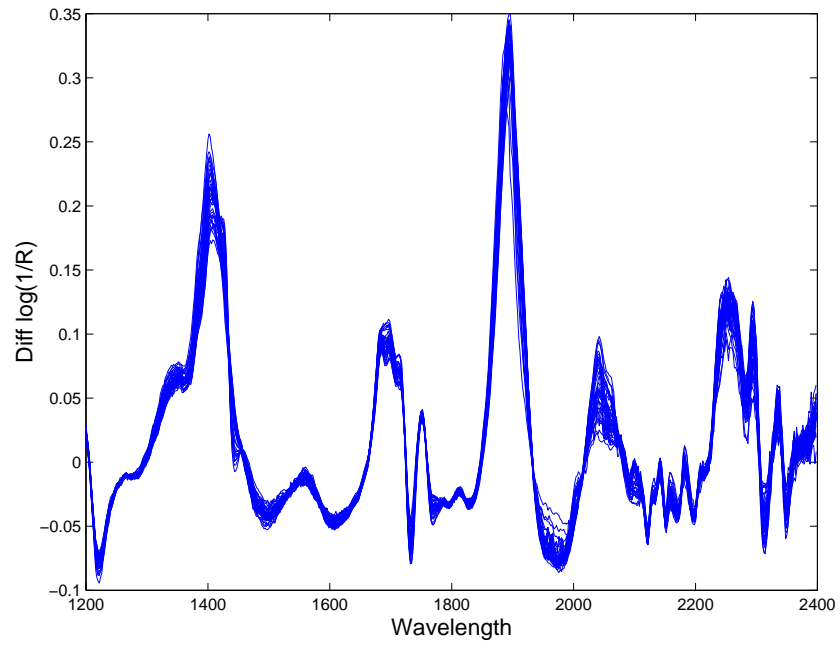


Figure 4: The Biscuit NIR data set.

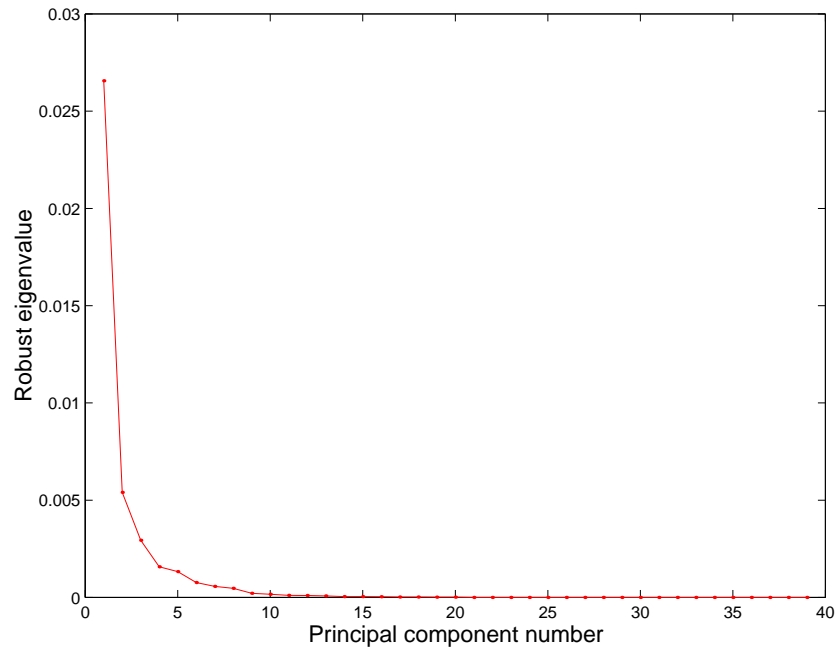


Figure 5: Scree plot of the Biscuit NIR data set.

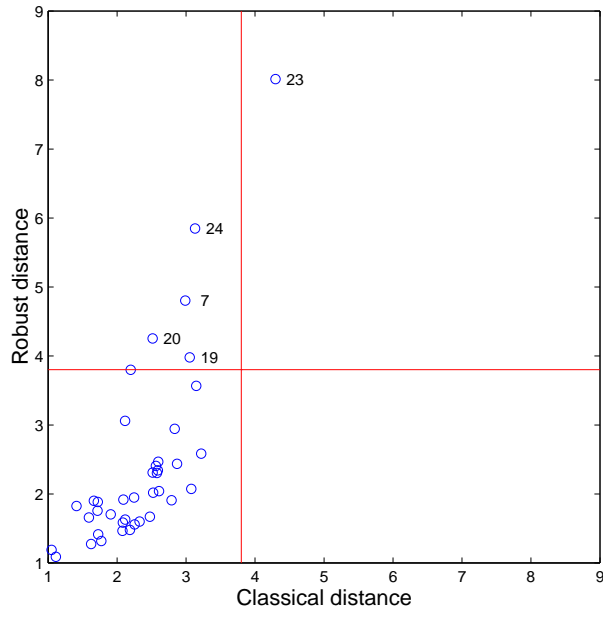


Figure 6: Distance-Distance plot of the Biscuit NIR data set.

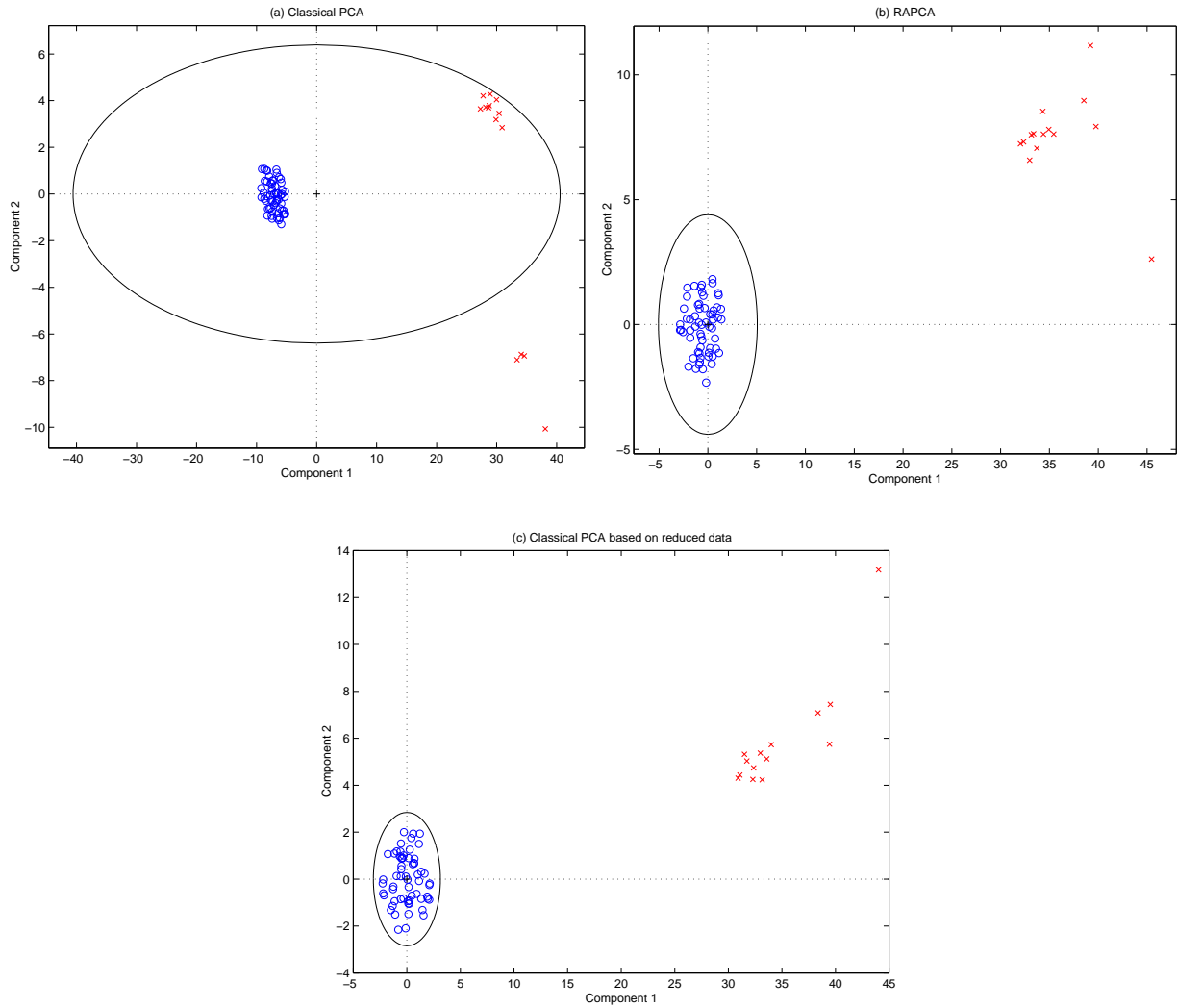


Figure 7: Scores of the Hawkins-Bradu-Kass data based on (a) the classical PCA method, (b) the robust RAPCA method, and (c) the classical PCA method performed on the reduced data set without outliers.

Table 1: Computation times (in seconds) of the C-R algorithm, the R-step approach, and RAPCA applied to the high-dimensional XRF data set, where k is the number of principal components.

k	C-R	R-step approach	RAPCA
5	62.2	361.0	29.3
10	101.5	766.0	30.0
15	142.1	1172.3	30.7
20	186.1	1576.7	31.3
25	227.2	1991.0	32.0
30	267.1	2406.9	32.6
35	307.2	2828.5	33.2
40	349.0	3245.5	33.7
45	388.8	3655.4	34.3

Table 2: Eigenvalues of the Hawkins data set computed with PCA_{full} , $RAPCA_{full}$, PCA_{red} and $RAPCA_{red}$.

	PCA_{full}	$RAPCA_{full}$	PCA_{red}	$RAPCA_{red}$
s_1^2	223.12	3.47	1.33	1.60
s_2^2	5.54	2.63	1.09	1.33
s_3^2	1.69	2.47	0.96	1.24
s_4^2	0.91	0.67	0.30	0.37

Table 3: Simulations results for $p = 4$.

	sim1		sim2		sim3	
	PCA	RAPCA	PCA	RAPCA	PCA	RAPCA
mean v_1	0.31	0.42	0.44	0.43	1.50	0.66
mean v_2	0.41	0.53	0.65	0.52	1.33	0.81
mean v_3	0.31	0.41	0.48	0.47	1.41	1.05
mean v_4	0.16	0.26	0.23	0.25	1.53	1.02
mean $\hat{\lambda}_1$	4.16	4.23	8.00	5.43	10.30	5.01
mean $\hat{\lambda}_2$	2.98	3.18	5.22	3.94	4.06	4.06
mean $\hat{\lambda}_3$	1.95	2.11	3.22	2.67	2.90	3.25
mean $\hat{\lambda}_4$	0.95	1.04	1.61	1.35	1.93	2.37
MSE $\hat{\lambda}_1$	0.31	0.39	18.69	2.61	40.13	1.38
MSE $\hat{\lambda}_2$	0.12	0.18	5.70	1.20	1.36	1.36
MSE $\hat{\lambda}_3$	0.08	0.13	1.82	0.62	0.92	1.73
MSE $\hat{\lambda}_4$	0.02	0.03	0.50	0.19	0.94	2.00

Table 4: Simulations results for $p = 20$.

	sim4		sim5		sim6	
	PCA	RAPCA	PCA	RAPCA	PCA	RAPCA
mean v_1	0.38	0.54	1.45	0.57	1.53	0.72
mean v_2	0.50	0.76	1.30	0.74	1.32	0.98
mean v_3	0.46	0.68	1.35	0.85	1.28	1.09
mean v_4	0.33	0.60	1.39	0.94	1.38	1.23
mean v_5	0.22	0.57	1.49	1.15	1.55	1.23
mean $\hat{\lambda}_1$	5.29	5.12	9.54	5.30	10.04	5.55
mean $\hat{\lambda}_2$	3.93	3.83	5.23	4.07	4.82	4.65
mean $\hat{\lambda}_3$	2.93	2.87	3.79	3.25	3.50	3.81
mean $\hat{\lambda}_4$	1.91	1.92	2.81	2.51	2.56	3.12
mean $\hat{\lambda}_5$	0.93	0.96	1.93	1.78	1.72	2.24
MSE $\hat{\lambda}_1$	0.48	0.53	20.76	0.55	25.47	0.70
MSE $\hat{\lambda}_2$	0.20	0.25	1.91	0.21	1.07	0.64
MSE $\hat{\lambda}_3$	0.12	0.15	0.82	0.24	0.45	0.81
MSE $\hat{\lambda}_4$	0.09	0.12	0.79	0.41	0.42	1.43
MSE $\hat{\lambda}_5$	0.02	0.03	0.93	0.74	0.58	1.71