Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-025-00656-3

REGULAR ARTICLE

# Cellwise robust and sparse principal component analysis

**Pia Pfeiffer**[1] · **Laura Vana-Gür**[1] · **Peter Filzmoser**[1]

Received: 7 November 2024 / Revised: 28 August 2025 / Accepted: 10 September 2025


**Abstract**
A first proposal of a sparse and cellwise robust PCA method is presented. Robustness to single outlying cells in the data matrix is achieved by substituting the squared loss function for the approximation error by a robust version. The integration of a sparsity-inducing $L_1$ or elastic net penalty offers additional modeling flexibility. For the resulting challenging optimization problem, an algorithm based on Riemannian stochastic gradient descent is developed, with the advantage of being scalable to high-dimensional data, both in terms of many variables as well as observations. The resulting method is called SCRAMBLE (Sparse Cellwise Robust Algorithm for Manifold-based Learning and Estimation). Simulations reveal the superiority of this approach in the high-dimenstional setting in comparison to established methods, both in the casewise and cellwise robustness paradigms. Two applications from the field of tribology underline the advantages of a cellwise robust and sparse PCA method.

**Keywords** Cellwise outliers · Robust PCA · Sparse PCA · Manifold learning

**Mathematics Subject Classification** 62H25 · 62G35 · 62R30

✉ Pia Pfeiffer
pia.pfeiffer@tuwien.ac.at

Laura Vana-Gür
laura.vana.guer@tuwien.ac.at

Peter Filzmoser
peter.filzmoser@tuwien.ac.at

[1] Institute of Statistics and Mathematical Methods in Economics, TU Wien, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

Published online: 07 November 2025

## 1 Introduction

The increasing prevalence of large data sets, especially high-dimensional data in the sense of many more variables than observations, motivates the use and development of dimension-reduction techniques. Principal Component Analysis (PCA), dating back to Pearson (1901) and Hotelling (1933), is one of the oldest and most widely applied dimension-reduction techniques. The idea of PCA is to find a low-dimensional representation of the data set in a way that preserves as much variance as possible (e.g. Jolliffe et al. 2003).

Although PCA is a well established method, there are challenges, particularly for high-dimensional data, thus data where the number $p$ of variables can exceed the number $n$ of observations. Among those challenges are sparsity and robustness. The former refers to a PCA solution that only involves a subset of variables, while the latter aims at reducing the effect of outliers for the estimation of the PCs. Outlier-robust PCA has been a very active research field, and thus many different approaches are available in the literature (see, e.g., She et al. 2016, for an overview). Standard robust approaches are designed to downweight entire observations (i.e. rows of the data matrix), even if only single entries (cells) of the observations are contaminated (Maronna et al. 2019). A method that only downweights outlying data cells would have particular advantages if $p \gg n$: Assuming that any data cell has the same chance of being contaminated, even a small amount of contamination can lead to many rowwise outliers, which at some point could even form the majority and cause breakdown of traditional rowwise robust methods (Maronna et al. 2019). Moreover, omitting "clean" data cells for the estimation can lead to a loss of valuable information, particularly in the case $n \ll p$. In this paper we will propose a method which is robust against single outlying cells in the data matrix, thus cellwise robust, and at the same time yields sparsity, thus zero entries in the loadings matrix.

In the following, we give a brief (and possibly incomplete) overview about rowwise robust sparse PCA methods, and cellwise robust approaches, before linking cellwise robustness with sparsity.

*Rowwise robust sparse PCA:* As PCA can be seen as a projection-pursuit (PP) problem, with the task to search for a projection direction that maximizes the variance of the projected observations, Croux and Ruiz-Gazen (2005) introduced a robust procedure by considering a robust variance (scale) estimator. Inspired by the SCoTLASS approach which adds a LASSO penalty on the direction vectors (Jolliffe et al. 2003), this PP approach was also reformulated to include an $L_1$ penalty, resulting in a robust and sparse PCA method (Croux et al. 2013).

The ROBPCA method by Hubert et al. (2005) combines a PP approach with the plug-in method by first projecting the data to a low-dimensional subspace and then applying a robust estimator of the covariance. This method was extended in Hubert et al. (2016), who proposed ROSPCA by integrating an $L_1$-penalty with the ROBPCA algorithm, which also leads to sparse solutions.

Zou et al. (2006) suggest reformulating PCA as a regression problem. Then, sparse loadings can be derived using elastic net (Zou and Hastie 2003) and LASSO (Tibshirani 1996) regression. A robust extension that is based on robust plug-in estimators for the covariance was proposed by Greco and Farcomeni (2016). In the case

$p > n$ they propose to use the unconstrained ROBPCA solution to obtain a plug-in estimator.

*Cellwise robust PCA:* The concept of cellwise outliers has been introduced in Alqallaf et al. (2009), but already De La Torre and Black (2003) and Maronna and Yohai (2008) formulated methods for low-rank approximation of a data matrix by downweighting outlying cells in the residual matrix of the reconstructed data set rather than outlying rows.

A PCA method that cannot only deal with cellwise outliers but also with missing values is MacroPCA (Hubert et al. 2019). This method combines the DetectDeviatingCells (DDC) algorithm of Rousseeuw and Bossche (2018) to detect cellwise outliers with a version of ROBPCA. Another approach to cellwise robust PCA, called cellPCA, was proposed in Centofanti et al. (2024), which combines loss functions to downweight cellwise and rowwise outliers into one objective function. Also cellwise robust covariance estimation as done in Centofanti et al. (2025) can be used as a basis for cellwise robust PCA.

*Cellwise robust sparse PCA:* One of the first papers on robust sparse PCA is Candès et al. (2011). However, sparsity here refers to a sparse residual matrix as the remainder of a robust rank-*k* approximation. In our contribution, we are interested in sparsity for the loadings matrix, as this simplifies the interpretation of the PCs. Although the work in Candès et al. (2011) is highly visible, the method is intended to deal with additive outliers, where a noise matrix is added to the data matrix, and not with outliers in the orthogonal complement to the PCA subspace, see She et al. (2016); Hubert et al. (2005).

A *cellwise robust and sparse PCA* method was proposed in Erichson et al. (2020). We will outline the differences to our proposal in more detail in the next section where we introduce our method on detail. An algorithm for its computation based on manifold learning is presented in Sects. 3, 4 provides a comparison of simulation results with alternative PCA methods. Applications in Sect. 5 demonstrate the usefulness of the method. The final Sect. 6 provides a summary and conclusions.

## 2 Cellwise robust sparse PCA for high-dimensional data

Based on a mean-centered (and possibly scaled) data matrix $\boldsymbol{X}$, with $n$ observations in the rows, and $p$ variables in the columns, the principal components (PCs) are defined by the linear combination $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{V}$ under the constraint that the columns of the $p \times p$ matrix $\boldsymbol{V}$ are normed to length 1 and orthogonal to each other. Since the variances of the columns of $\boldsymbol{Z}$ have to be maximized, the solution for $\boldsymbol{V}$ can be obtained by the spectral decomposition $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{V}}\hat{\boldsymbol{A}}\hat{\boldsymbol{V}}'$, where $\hat{\boldsymbol{\Sigma}}$ is the estimated covariance matrix of $\boldsymbol{X}$, and $\hat{\boldsymbol{A}} = \text{Diag}\,(\hat{a}_1, \ldots, \hat{a}_p)$ is the diagonal matrix with the corresponding estimated eigenvalues, arranged in descending order (Jolliffe et al. 2003). The matrix $\hat{\boldsymbol{V}}$ is also known as loadings matrix, while $\hat{\boldsymbol{Z}} = \boldsymbol{X}\hat{\boldsymbol{V}}$ refers to the PCA score matrix. Traditionally, the sample covariance matrix $\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}'\boldsymbol{X}$ is used for $\hat{\boldsymbol{\Sigma}}$, resulting in eigenvectors $\tilde{\boldsymbol{V}}$, eigenvalues $\tilde{\boldsymbol{A}}$, and classical principal components

$\tilde{\boldsymbol{Z}} = \boldsymbol{X}\tilde{\boldsymbol{V}}$. The identical solution $\tilde{\boldsymbol{V}}$ can be obtained from a singular value decomposition (SVD) of $\boldsymbol{X}$ as $\boldsymbol{X} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{V}}'$ (e.g. Jolliffe et al. 2003).

As the main interest is usually in the first $k$ PCs, where $k < p$, or even $k \ll p$ for high-dimensional data, it is not necessary to compute the whole $p \times p$ matrix $\hat{\boldsymbol{V}}$, but to only focus on the matrix $\hat{\boldsymbol{V}}_k \in \mathbb{R}^{p\times k}$ with the first $k$ columns, to obtain the first $k$ PCs $\hat{\boldsymbol{Z}}_k = \boldsymbol{X}\hat{\boldsymbol{V}}_k$. Especially for $p \gg n$, the approach based on a spectral decomposition of the estimated covariance matrix is numerically not attractive, and thus SVD is commonly employed in this case. This leads to a rank-$k$ approximation $\tilde{\boldsymbol{X}}_k = \tilde{\boldsymbol{U}}_k\tilde{\boldsymbol{D}}_k\tilde{\boldsymbol{V}}_k'$ of $\boldsymbol{X}$, with $\tilde{\boldsymbol{U}}_k \in \mathbb{R}^{n\times k}$, $\tilde{\boldsymbol{V}}_k \in \mathbb{R}^{k\times k}$ and $\tilde{\boldsymbol{D}}_k \in \mathbb{R}^{k\times k}$ with elements $\tilde{d}_{ii} \geq 0$ for $i = 1, \ldots, k$ and $\tilde{d}_{ij} = 0$ otherwise. The rank-$k$ SVD is the best rank-$k$ approximation in the Frobenius norm (Eckart and Young 1936),

$$\tilde{\boldsymbol{V}}_k = \operatorname{argmin}_{\boldsymbol{V}_k} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}_k\boldsymbol{V}_k'\|_F^2 \tag{1}$$

for any $p \times k$ matrix $\boldsymbol{V}_k$ with $\operatorname{rank}(\boldsymbol{V}_k) \leq k$ and $\boldsymbol{V}_k'\boldsymbol{V}_k = \boldsymbol{I}_k$. We can define the residual matrix

$$\boldsymbol{R} = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}_k\boldsymbol{V}_k' \quad \text{with elements} \quad r_{ij}, \text{ for } i = 1, \ldots, n \text{ and } j = 1, \ldots, p, \tag{2}$$

and problem (1) is equivalent to minimizing the sum of all squared residuals by using a matrix $\boldsymbol{V}_k$ as defined above. Our goal is to obtain an estimate of the loadings matrix which is (i) robust against outliers that may appear in any cell of the data matrix, and (ii) is sparse, and thus forces small (absolute) loadings entries to be zero.

As the objective function (1) is based on residual sum-of-squares, it is sensitive to outlying data cells. De La Torre and Black (2003) and Maronna and Yohai (2008) replaced the Frobenius norm in (1) with a more robust loss function. They estimate two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of rank $k$, where $\boldsymbol{B}$ is constrained to be orthonormal, such that $\boldsymbol{A}\boldsymbol{B}'$ approximates $\boldsymbol{X}$. The matrix $\boldsymbol{B}$ has the role of the loadings matrix, which in these approaches is not intended to be sparse. Maronna and Yohai (2008) are not even concerned about identifying $\boldsymbol{A}$ and $\boldsymbol{B}$, as their interest is in low-rank approximation of the data matrix, and not in estimating a loadings matrix.

In our approach we want to combine cellwise robustness with sparsity. Particularly, in the high-dimensional case it is desirable to obtain (many) zeros in the matrix $\boldsymbol{V}_k$, since this simplifies the interpretation of the PCs. Sparsity has also another advantage: Already De La Torre and Black (2003) pointed out that even single outlying cells in the data matrix can propagate to contaminate even entire rows in the residual matrix. However, if the loadings matrix is very sparse, this propagation effect is limited. Erichson et al. (2020) proposed a cellwise robust and sparse PCA method that works as follows. They estimate two rank $k$ matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, such that $\boldsymbol{X}\boldsymbol{A}_1\boldsymbol{A}_2'$ approximates $\boldsymbol{X}$, where $\boldsymbol{A}_2$ is supposed to be orthogonal. Sparsity for $\boldsymbol{A}_1$ is achieved by adding a sparsity inducing penalty on $\boldsymbol{A}_1$ to the objective function. The matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are estimated in an alternating scheme such that the back-rotated principal components $\boldsymbol{X}\boldsymbol{A}_1$ are close to $\boldsymbol{X}$. In order to obtain robustness, they replace the Frobenius norm by the Huber loss function, as this can be formulated by using a matrix $\boldsymbol{S}$ of the same dimension as $\boldsymbol{X}$, which absorbs big cellwise (additive) recon-

struction errors within a Frobenius norm, plus an L1 penalty on $\boldsymbol{S}$ to achieve sparsity on this error matrix, see Equation (17) in Erichson et al. (2020).

In our approach we will stick to the original PCA problem and thus focus on the residual matrix given in Equation (2). Thus, the loadings matrix and the matrix for back-projection will be identical. Further, we will make use of alternative loss functions to achieve more effective downweighting of large reconstruction errors, in combination with robustified starting estimates. Both will be important to obtain robustness against high amounts of contamination. More concretely, the rank-$k$ approximation criterion (1) can be combined with a criterion to obtain sparsity. When using an elastic net penalty, the modified problem formulation is

$$\hat{\boldsymbol{V}}_k = \operatorname{argmin}_{\boldsymbol{V}_k'\boldsymbol{V}_k=\boldsymbol{I}_k} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}_k\boldsymbol{V}_k'\|_F^2 + \sum_{j=1}^{k} \lambda_j(\alpha\|\boldsymbol{v}_j\|_2^2 + (1-\alpha)\|\boldsymbol{v}_j\|_1), \quad (3)$$

where $\boldsymbol{v}_j$ refers to the $j$-th column of $\boldsymbol{V}_k$, $1 \le j \le k < p$, the strength of regularization for each component is controlled by $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)'$, and the elastic net mixing parameter $\alpha$ controls the sparsity (Zou and Hastie 2005).

As the least squares loss is highly susceptible to outlying observations, we propose to substitute it with a robust loss function, similar to the suggestion in Maronna and Yohai (2008), in order to obtain a cellwise robust and sparse PCA method:

$$\hat{\boldsymbol{V}}_k = \operatorname{argmin}_{\boldsymbol{V}_k'\boldsymbol{V}_k=\boldsymbol{I}_k} \frac{1}{np} \sum_{j=1}^{p} \hat{\sigma}_j^2 \sum_{i=1}^{n} \rho\left(\frac{r_{ij}}{\hat{\sigma}_j}\right) + \sum_{j=1}^{k} \lambda_j(\alpha\|\boldsymbol{v}_j\|_2^2 + (1-\alpha)\|\boldsymbol{v}_j\|_1), \quad (4)$$

where $r_{ij}$ are the residuals from (2), and $\hat{\sigma}_j$ is a column-wise estimator of residual scale. The function $\rho$ corresponds to a robust loss function (Maronna et al. 2019), and popular choices are the Huber loss, defined as

$$\rho_H(r) = \begin{cases} r^2 & \text{for } |r| \le b, \\ b|r| & \text{otherwise,} \end{cases} \quad (5)$$

the Tukey loss, defined as

$$\rho_T(r) = \begin{cases} \left(\frac{r}{c}\right)^2 \left(3 - 3\left(\frac{r}{c}\right)^2 + \left(\frac{r}{c}\right)^4\right) & \text{for } |r| \le c, \\ 1 & \text{otherwise,} \end{cases} \quad (6)$$

or a trimmed version of the least squares loss, defined as

$$\rho_{LTS}(r) = \begin{cases} r^2 & \text{for } |r| \le |r|_{(h)}, \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

where $|r|_{(i)}$ refers to the ordered values of the absolute residuals, i.e., $|r|_{(1)} \le \ldots \le |r|_{(n)}$ and $h \in \lfloor n/2, n \rfloor$. The trimming is applied column-wise.

The choice of the loss function will also determine the robustness properties of $\hat{\boldsymbol{V}}_k$, see also Maronna and Yohai (2008) for more detailed discussions. For either of these loss functions, the influence of data points with large scaled residuals is reduced, resulting in a more robust estimate (Maronna et al. 2019). Appropriate parameter choices for the constants $b$ and $c$ in (5) and (6), respectively, are proposed in the literature (Maronna et al. 2019). Throughout our experiments, we will use the default parameters $b = 1.35$, $c = 1.35$, assuming that the regular observations are standard Gaussian and $h = 0.5$, leading to maximum robustness for trimming.

Note that a multiplication with $\hat{\sigma}_j^2$ in the objective function (4) is important, since this guarantees that for the special choice $\rho(r) = r^2$ one obtains the objective function (3) of the non-robust version. The estimation of the residual scale will be discussed in detail in the next section.

For outlier diagnostics, i.e., for distinguishing between regular observations, good and bad leverage points, and orthogonal outliers, a diagnostic plot can be constructed as proposed by Hubert et al. (2005). The score distances (SD, i.e., Mahalanobis-like measure of distance of an observation within the PC space) and orthogonal distances (OD, i.e., the orthogonal distance of an observation to the space spanned by the first $k$ PCs) are computed based on the robust PCA result and plotted together with the cutoff values, also described in detail in Hubert et al. (2005).

## 3 Algorithm

Problem (4) is an optimization problem under constraints. The objective function

$$\mathcal{L}(\boldsymbol{V}) = \frac{1}{np} \sum_{j=1}^{p} \hat{\sigma}_j^2 \sum_{i=1}^{n} \rho\left(\frac{r_{ij}(\boldsymbol{V})}{\hat{\sigma}_j}\right) + \sum_{j=1}^{k} \lambda_j (\alpha \|\boldsymbol{v}_j\|_2^2 + (1-\alpha) \sum_{i=1}^{n} \|v_{ij}\|_1) \quad (8)$$

is minimized under the constraint that $\boldsymbol{V}'\boldsymbol{V} = \boldsymbol{I}_k$, which corresponds to the Stiefel Manifold, defined as the set of orthonormal matrices, i.e., $\mathrm{St}(p,k) = \{\boldsymbol{V} \in \mathbb{R}^{p \times k} : \boldsymbol{V}'\boldsymbol{V} = \boldsymbol{I}_k\}$.

We can therefore cast this problem in the framework of optimization on manifolds. Gradient methods such as the Newton or Conjugate Gradient algorithm on manifolds have already been studied in Edelman et al. (1998), and Bonnabel (2013) have extended SGD (Stochastic Gradient Descent) to the case when the objective function is defined on a Riemannian manifold and derived convergence properties for the algorithm. When the gradient is calculated on the whole dataset, it is referred to as batch gradient descent in the machine learning literature, see, e.g., Goodfellow et al. (2016). When big amounts of data have to be processed, (minibatch) SGD is preferable, this means that the gradient is computed on each sample, or a minibatch of samples. Mathematically, the true gradient of the objective function corresponds to the expectation of the gradient over the data-generating distribution. For batch gradient descent, the true gradient is approximated by the gradient over the whole training set. When there is a large number of observations, however, computing this expec-

tation over the whole dataset is not feasible, therefore it is computed over random subsamples, which are called minibatches (Goodfellow et al. 2016). Therefore, by application of a suitable variant of SGD we can ensure the scalability of the proposed algorithm to datasets containing a very large number of observations, which would not be possible with an approach based on alternating regressions, for example. In the following, we describe the algorithm for batch gradient descent for ease of notation.

### 3.1 Manifold optimization

Given a starting point $\boldsymbol{V}_0 \in \mathrm{St}(p,k)$, subsequent iterations lie on the same manifold. This is accomplished as follows: First, the gradient at step $t$, $\boldsymbol{G}_t := \nabla_{\boldsymbol{V}_t}\mathcal{L}(\boldsymbol{V}_t) = \left(\frac{\partial \mathcal{L}(\boldsymbol{V}_t)}{\partial \boldsymbol{v}_l}\right)_{l=1}^{k}$, is computed, then the gradient is projected on the tangent space of the manifold at the current parameter value, denoted by $\mathcal{T}_{\boldsymbol{V}_t}\mathrm{St}(p,k)$. Let $\boldsymbol{P}_{\boldsymbol{G}_t}$ denote the projection of the gradient, which can be computed as $\boldsymbol{P}_{\boldsymbol{G}_t} = -\gamma_t(\boldsymbol{I}_p - \boldsymbol{V}_t\boldsymbol{V}_t')\boldsymbol{G}_t$, where $\gamma_t$ refers to the step size.

Finally, the gradient step is executed on the manifold. This can be done via an exponential map, $\boldsymbol{V}_{t+1} \leftarrow \exp_{\boldsymbol{V}_t}(-\gamma_t \boldsymbol{P}_{\boldsymbol{G}_t})$, or via a retraction, a first-order approximation of the exponential map, denoted by $\boldsymbol{V}_{t+1} \leftarrow R_{\boldsymbol{V}_t}(-\gamma_t \boldsymbol{P}_{\boldsymbol{G}_t})$. Numerically, the latter is preferable, and we can use $R_{\boldsymbol{V}_t}(\boldsymbol{P}) = \mathrm{qf}(\boldsymbol{P})$, where qf() extracts the orthogonal factor from the QR decomposition. This particular retraction was also studied in Proposition 3 in Bonnabel (2013) and essentially follows the gradient in the Euclidean space and then orthonormalizes the matrix at each step.

For the computation of the gradient, the continuous differentiability of the loss function $\rho$ is a necessary condition. This assumption is fulfilled for the least squares loss without regularization, for the robust loss functions and different penalties, however, it does not hold. While the Huber loss (5) can be approximated by the differentiable Pseudo-Huber loss function (Hartley and Zisserman 2003), $\rho_{PH}(r) = b^2(\sqrt{1+(r/b)^2}-1)$, the treatment of the LTS loss (7) requires more thought. It is easy to see though, that a gradient step in terms of the $\rho_{LTS}$ function can be expressed as a re-weighting step: $\rho_{LTS}$ is continuously differentiable on the set of points for which $|r| \leq |r|_{(h)}$. Let $H_t$ denote this $h$-subset computed based on the current iterate $\boldsymbol{V}_t$, then $\mathcal{L}(\boldsymbol{V}_t, H_t)$ corresponds to the value of the objective function depending on $H_t$. After each gradient step, $H_{t+1}$ is updated using the current iterate $\boldsymbol{V}_{t+1}$ and we get the inequality

$$\mathcal{L}(\boldsymbol{V}_{t+1}, H_{t+1}) \leq \mathcal{L}(\boldsymbol{V}_{t+1}, H_t) \leq \mathcal{L}(\boldsymbol{V}_t, H_t), \tag{9}$$

where the inequality on the right is due to the gradient update (Bonnabel 2013), and the inequality on the left holds because the largest standardized residuals are trimmed. Note that for the stochastic variant of the gradient step, the inequality only holds in expectation.

### 3.2 Initialization

As the loss function (8) is based on the approximation of the data matrix via rank-$k$ SVD, it seems natural to consider the first $k$ singular values as starting points. Despite the desirable properties of Riemannian SGD studied by Bonnabel (2013), however, we cannot hope to get convergence to a global minimum, as the loss function is not convex on $\mathrm{St}(p, k)$. Therefore, it is crucial to choose an initial estimate $\boldsymbol{V}_0$ that is robust in the presence of outliers.

We propose first applying a transformation $g$ to the data matrix and then computing the SVD of $\boldsymbol{Y} = g(\boldsymbol{X})$, resulting in the first $k$ right-singular vectors of $g(\boldsymbol{X})$ as the initial estimate $\boldsymbol{V}_0$. The procedure is inspired by the robust high-dimensional product-moment correlation, studied in Raymaekers and Rousseeuw (2021) where robustness properties of different data transformations are investigated. In the following, we consider one of the following options for the transformation, and later on, compare their results. Both options involve estimators of location $t_j$ and scale $c_j$ of the variables ($j = 1, \ldots, p$). Here we use the median for location and the Qn estimator (Rousseeuw and Croux 1993) for scale.

1. Rank transformation: The elements of the transformed data matrix $\boldsymbol{Y}$ are given by $y_{ij} = g(x_{ij}) = {}^1\!/_n(\mathrm{rank}_i(x_{ij}) - 0.5) \cdot c_j + t_j$.
2. Wrapping transformation: Denote $z_{ij} = \frac{x_{ij} - t_j}{c_j}$, for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, as the column-wise robustly standardized data. The elements of the transformed data matrix $\boldsymbol{Y}$ are given by $y_{ij} = g(x_{ij}) = \psi_{b,c}(z_{ij}) \cdot c_j + t_j$. The $\psi$-function is given by

$$\psi_{b,c}(z) = \begin{cases} z & \text{if } 0 \leq |z| \leq b, \\ q_1 \tanh\left(q_2(c - |z|)\right)\mathrm{sign}(z) & \text{if } b \leq |z| \leq c, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where the values $q_1$ and $q_2$ can be derived for any combination of $0 < b < c$ (Raymaekers and Rousseeuw 2021). We use the default values $b = 1.5$ and $c = 4$ as proposed in Raymaekers and Rousseeuw (2021) and implemented in the function `wrap()` of the R package **cellWise** (Raymaekers and Rousseeuw 2023a).

The choice of transformation is discussed in Raymaekers and Rousseeuw (2021), and depends on the dataset at hand. The application of a rank transformation is straightforward, computationally attractive for large datasets and does not require additional parameter choices. Wrapping can be tuned for a better compromise between robustness and efficiency, requiring additional computational resources.

### 3.3 Residual scale

Based on an initial estimate $\boldsymbol{V}_0$, we can compute the residuals $\boldsymbol{R}(\boldsymbol{V}_0) = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}_0\boldsymbol{V}_0'$ with the elements $r_{ij}(\boldsymbol{V}_0)$. The objective function (8) needs an estimate of the residual scale, $\hat{\sigma}_j$, for the $j$-th column of this matrix ($j = 1, \ldots, p$). Minimizing the objective function then yields an updated estimate of $\boldsymbol{V}$, and thus also new residuals,

from which the residual scale needs to be re-estimated. For estimating this residual scale, Maronna and Yohai (2008) suggest using an M estimator of scale. As the proposed algorithm requires repeated computation of the residual scale, it is desirable to choose an estimator with a high breakdown point which is also easy to compute. We therefore propose to use the simple least median of squares estimator given by $\hat{\sigma}_j = \text{median}_i |r_{ij}|$ for this purpose.

### 3.4 Sparsity inducing penalties

When the elastic net parameter in (8) is set to $\alpha = 0$, we get an $L_1$-penalty, a popular choice for inducing sparsity in the PCA loadings (Croux et al. 2013; Hubert et al. 2016). While the $L_1$-norm is not continuously differentiable, it can be approximated by a differentiable function, which converges towards the $L_1$-norm. In our implementation we use $|v_{jl}| = v_{jl}\text{sign}(v_{jl}) = \lim_{c\to\infty} v_{jl}\tanh(c \cdot v_{jl})$, as discussed by Öllerer et al. (2015). The more the constant $c$ is increased, the better the absolute value function is approximated. In our implementation, we set $c = 1000$. Due to this approximation and the nature of gradient-based updates, this procedure does not lead to true sparsity in the loadings but only shrinks the elements of the loadings matrix close to zero. In order to get truly sparse results, we propose to threshold the loadings in the following way: First, we track the relative change in each of the elements of the loadings matrix between the iterations $\boldsymbol{V}_t$ and $\boldsymbol{V}_{t+1}$. Then, the threshold value is computed as the average change of all elements during the last $M$ iterations plus two standard deviations. If the absolute value of an entry of $\hat{\boldsymbol{V}}$ is lower than the threshold, it is set to 0. In the implementation, we use $M = 10$, which we found to be a reasonable choice in our experiments, as increasing $M$ above this value brought minimal improvement.

The complete algorithm is summarized in Algorithm 1.

1: Compute transformations $\boldsymbol{Y} \leftarrow g(\boldsymbol{X})$
2: Initialize $\boldsymbol{V}_0$ as first $k$ right-singular vectors of $\boldsymbol{Y}$
3: **if** $\rho = \rho_{LTS}$ **then**
4: Initialize the $h$-subset $H_0$ based on $\boldsymbol{V}_0$ ▷ for LTS loss, initialize $H_0$
5: **else**
6: $H_0 \leftarrow \{1, \ldots, n\}$ ▷ otherwise, $h$-subset consists of all samples
7: **end if**
8: **while** true **do**
9: $\boldsymbol{G}_t \leftarrow \nabla_{\boldsymbol{V}} \mathcal{L}(\boldsymbol{V}_t, H_t)$ ▷ compute gradient
10: $\boldsymbol{P}_{\boldsymbol{G}_t} \leftarrow \text{proj}_{\mathcal{T}_{\boldsymbol{V}_t}\text{St}(p,k)}(\boldsymbol{G}_t)$ ▷ projection of gradient onto tangent space
11: $\boldsymbol{V}_{t+1} \leftarrow R_{\boldsymbol{V}_t}(-\gamma_t \boldsymbol{P}_{\boldsymbol{G}_t})$ ▷ gradient step via retraction
12: **if** $\rho = \rho_{LTS}$ **then**
13: update $H_{t+1}$ based on $\boldsymbol{V}_{t+1}$ ▷ for LTS loss, update $h$-subset
14: **else**
15: $H_t = H_0$
16: **end if**
17: $d_{t+1} \leftarrow \|\boldsymbol{V}_{t+1} - \boldsymbol{V}_t\| / \|\boldsymbol{V}_t\|$ ▷ track relative change
18: $t+ = 1$
19: **if** $\|\mathcal{L}(\boldsymbol{V}_{t+1}, H_{t+1}) - \mathcal{L}(\boldsymbol{V}_t, H_t)\| < \delta$ **then**
20: break
21: **end if**
22: **end while**
23: $\bar{t} \leftarrow \text{avg}[d_m]_{m=i}^{i-M+1} + 2\text{sd}[d_m]_{m=i}^{i-M+1}$ ▷ compute based on last $M$ iterations
24: $\hat{\boldsymbol{V}} \leftarrow [v_{jl} \text{ if } |v_{jl}| > \bar{t}, 0 \text{ otherwise}]_{jl}$ ▷ thresholding
25: $\hat{\boldsymbol{Z}} \leftarrow \boldsymbol{X}\hat{\boldsymbol{V}}$ ▷ compute robust scores
26: $\hat{\boldsymbol{a}} \leftarrow \text{Qn}^2(\boldsymbol{X}\hat{\boldsymbol{V}})$ ▷ compute robust variances of the scores

**Algorithm 1** Robust and Sparse PCA via Manifold Optimization

Line 25 of Algorithm 1 returns the principal components, and line 26 their variances, here estimated per component with the Qn-scale estimator (Rousseeuw and Croux 1993).

### 3.5 Selection of sparsity parameter

While the elastic net mixing parameter $\alpha$ in (8) is set in advance by the user, the sparsity parameter $\lambda$ is chosen depending on the data. Similarly to Croux et al. (2013), we choose the tradeoff-product criterion (TPO) that leads to a compromise between explained variance and sparsity in the loadings. In the following, we denote the columns of the estimate $\hat{\boldsymbol{V}}$ as $\hat{\boldsymbol{v}}_l$, for $l = 1, \ldots, k$. The original criterion

$$\text{TPO} = \sum_{l=1}^{k} \text{Qn}^2(\boldsymbol{X}\hat{\boldsymbol{v}}_l) \cdot \left(1 - \frac{\#\{\hat{\boldsymbol{v}}_l \neq 0\}}{p}\right)$$

where $\#\{\hat{\boldsymbol{v}}_l \neq 0\}$ returns the number of non-zero components in $\hat{\boldsymbol{v}}_l$, can be adapted for non-sparse regularization by including the elastic net parameters $\alpha$:

$$\text{TPO} = \sum_{l=1}^{k} \text{Qn}^2(\boldsymbol{X}\hat{\boldsymbol{v}}_l) \cdot \left(1 - \alpha \frac{\#\{\hat{\boldsymbol{v}}_l \neq 0\}}{p}\right). \tag{11}$$

The maximum of this score function is now determined using Bayesian optimization: The advantage of this procedure is that, contrary to cross-validation in combination with grid search, the information from previous function evaluations can be exploited. This way, a bigger search space can be covered. For the basic algorithm, it is assumed that there is a budget of a total of $N$ function evaluations. A Gaussian prior is placed on the score function, then its value is observed at $n_0$ points. Until $N$ iterations are reached, the following steps are repeated: (i) update the posterior probability distribution on the score function, (ii) determine the maximum of the acquisition function, and (iii) observe the score value at this parameter configuration (see, e.g., Snoek et al. 2012). We used the implementation in the R package **ParBayesianOptimization** (Wilson 2022) with the commonly used expected improvement as the acquisition function and the tradeoff product optimization (TPO) as the score function.

The proposed algorithm is called SCRAMBLE (Sparse Cellwise Robust Algorithm for Manifold-based Learning and Estimation) and implemented in R (R Core Team 2024), in package **scramble**, available in the supplement.

## 4 Simulations

A simulation study was conducted to evaluate the performance of the proposed method using different loss functions in comparison with other approaches, namely Robust and Sparse PCA (ROSPCA) (Hubert et al. 2016) for the rowwise contamination model, and MacroPCA (Hubert et al. 2019) for the cellwise contamination model, although this method cannot induce sparsity.

### 4.1 Simulation settings

Similar to Croux et al. (2013), Hubert et al. (2016, 2019), several different contamination settings are considered. The casewise Tukey-Huber contamination model can be formalized as

$$X_1 = (1 - B)X_2 + BX_3, \tag{12}$$

where $X_1$, $X_2$ and $X_3$ are $p$-dimensional vectors, $X_2 \sim F$ with $F$ corresponding to the model distribution, and $X_3 \sim G$ with $G$ corresponding to the outlier-generating distribution. $B \sim Bin(1, \varepsilon)$ can be interpreted as a *contamination indicator* (Alqallaf et al. 2009). In the multivariate setting, this model has been criticized for only allowing rowwise contamination, when in reality – especially in high-dimensional settings – it is likely that only a few columns in many rows are affected by outliers.

For this cellwise contamination framework, Alqallaf et al. (2009) have formalized the independent contamination model as

$$X_1 = (\boldsymbol{I} - \boldsymbol{B})X_2 + \boldsymbol{B}X_3, \tag{13}$$

where $\boldsymbol{B} = diag(B_1, B_2, \ldots, B_p)$ are independent $B_i \sim Bin(1, \varepsilon_i)$.

A low- and a high-dimensional simulation setting is considered, and they are adapted from Croux et al. (2013) and Hubert et al. (2016). Clean data $\boldsymbol{X}$ are generated from a multivariate normal distribution $\mathcal{N}_p(\boldsymbol{0}, \boldsymbol{\Sigma})$ with zero means and covariance $\boldsymbol{\Sigma} = \boldsymbol{C}^{1/2}\boldsymbol{A}\boldsymbol{C}^{1/2}$ given as follows :

1. Low-dimensional, order 2: $p = 10, n = 50$ observations

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{S}^1_{4\times 4} & \boldsymbol{0}_{4\times 4} & \boldsymbol{0}_{4\times 2} \\ \boldsymbol{0}_{4\times 4} & \boldsymbol{S}^2_{4\times 4} & \boldsymbol{0}_{4\times 2} \\ \boldsymbol{0}_{2\times 4} & \boldsymbol{0}_{2\times 4} & \boldsymbol{I}_{2\times 2} \end{bmatrix},$$

where

$$\boldsymbol{S}^1_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0.9 & \text{otherwise,} \end{cases}$$

and

$$\boldsymbol{S}^2_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0.7 & \text{otherwise.} \end{cases}$$

The matrix $\boldsymbol{C}$ assigns the same scale to all variables of the same group, and it is given as $\boldsymbol{C} = \text{diag}(\boldsymbol{100}_4, \boldsymbol{25}_4, \boldsymbol{4}_2)$, where $\boldsymbol{a}_b$ is a vector with $b$ replicates of the number $a$. The true loadings of the first component in this setting are $\boldsymbol{v}_1 = {}^1\!/_2(\boldsymbol{1}_4, 0, \ldots, 0)$, and of the second component $\boldsymbol{v}_2 = {}^1\!/_2(\boldsymbol{0}_4, \boldsymbol{1}_4, 0, \ldots, 0)$.

2. High-dimensional, order 2: $p = 500, n = 100$ observations

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{S}^1_{20\times 20} & \boldsymbol{0}_{20\times 20} & \boldsymbol{0}_{20\times 460} \\ \boldsymbol{0}_{20\times 20} & \boldsymbol{S}^2_{20\times 20} & \boldsymbol{0}_{20\times 460} \\ \boldsymbol{0}_{460\times 20} & \boldsymbol{0}_{460\times 20} & \boldsymbol{I}_{460\times 460} \end{bmatrix},$$

where $\boldsymbol{S}^1$ and $\boldsymbol{S}^2$ are defined as before. $\boldsymbol{C}$ is given as $\boldsymbol{C} = \text{diag}(\boldsymbol{100}_{20}, \boldsymbol{25}_{20}, \boldsymbol{4}_{460})$. Similarly, the true loadings for the first 2 components correspond to the vectors $\boldsymbol{v}_1 = {}^1\!/_{\sqrt{20}}(\boldsymbol{1}_{20}, 0, \ldots, 0)$ and $\boldsymbol{v}_2 = {}^1\!/_{\sqrt{20}}(\boldsymbol{0}_{20}, \boldsymbol{1}_{20}, 0, \ldots, 0)$.

Casewise contaminated data are generated by replacing $\varepsilon\%$ of rows with data generated from a $p$-variate normal distribution $\mathcal{N}(\boldsymbol{\mu}_{out}, \boldsymbol{I}_p)$, with $\boldsymbol{\mu}_{out} = (2, 4, 2, 4, 0, -1, 1, 0, 1, -1, \ldots, 1, 0, 1, -1)$, corresponding to the $\boldsymbol{\mu}_{out}$ vector suggested by Croux et al. (2013) and also used by Hubert et al. (2016).

For the cellwise contamination setting, the outlying cells are generated as follows. We first randomly select $\varepsilon\%$ cells in each column of the data matrix, and then focus on the rows with at least one selected cell. The cell(s) is (are) replaced by a multiple of the last eigenvector of $\boldsymbol{\Sigma}$, restricted to the rows/columns given by the indexes of

those cells. Thus, these cells are outlying in the given subspace and represent structural outliers (Agostinelli et al. 2015; Rousseeuw and Bossche 2018). To generate these outlying cells, we used the function `generateData(outlierType = "cellwiseStructured")` of the R package **cellWise** (Raymaekers and Rousseeuw 2023a), where the parameter `gamma` for the shift of the outliers in the direction of the orthogonal complement with respect to the center was chosen as 2 or 5, respectively, to evaluate the sensitivity of the different methods to this type of contamination. Additionally, sparsity is assumed for all approaches and the corresponding parameters are set accordingly ($\alpha = 0$ for SCRAMBLE).

### 4.2 Performance measures

To measure the accuracy of the algorithm, we use the *principal angles* between subspaces, which are defined as follows: Let $\boldsymbol{V}$ and $\hat{\boldsymbol{V}}$ be orthonormal bases for the subspaces $\mathcal{V}$ and $\hat{\mathcal{V}}$, and assume that $\dim(\mathcal{V}) \leq \dim(\hat{\mathcal{V}})$. Then, the principal angle $\theta$ can be computed as $\theta = \sin^{-1}(\sigma_{\max}((\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}')\hat{\boldsymbol{V}}))$, where $\sigma_{\max}(\cdot)$ yields the largest singular value of the projected matrix (Björck and Golub 1973). We report the principal angle scaled to lie in [0, 1], thus divided by $\pi/2$, as implemented in the function `angle()` in the R package **rospca** (Reynkens 2018) and described in Hubert et al. (2005). Values close to zero imply that the subspaces are very similar.

The correct level of sparsity is evaluated by the true-positive rate (TPR) and true-negative rate (TNR), defined as

$$\mathrm{TPR}(\boldsymbol{V}, \hat{\boldsymbol{V}}) = \frac{\#\{(j,k) : v_{jk} \neq 0 \text{ and } \hat{v}_{jk} \neq 0\}}{\#\{(j,k) : v_{jk} \neq 0\}} \tag{14}$$

$$\mathrm{TNR}(\boldsymbol{V}, \hat{\boldsymbol{V}}) = \frac{\#\{(j,k) : v_{jk} = 0 \text{ and } \hat{v}_{jk} = 0\}}{\#\{(j,k) : v_{jk} = 0\}}, \tag{15}$$

where $v_{jk}$ and $\hat{v}_{jk}$ are the corresponding elements of $\boldsymbol{V}$ and $\hat{\boldsymbol{V}}$, respectively. TNR and TPR correspond to the rate of correctly identified non-zero elements and zero elements, respectively, in the loadings.

The scaled principal angle corresponds to the accuracy of the subspace estimation and should be controlled at a low level even in the presence of outliers. The TPR and TNR depict how stable the correct estimation of the sparsity is; these performance measures should be close to 1.

### 4.3 Simulation results

The proposed method is compared to ROSPCA (Hubert et al. 2016) in the casewise contamination setting, using the implementation in the R package **rospca** (Reynkens 2018), and MacroPCA as implemented in the R package **cellWise** (Raymaekers and Rousseeuw 2023a) in the cellwise contamination setting. While the latter does not enforce sparsity in the loadings, it is the only other cellwise robust PCA method that software is readily available for, and a comparison in terms of accuracy of subspace

estimation should be insightful. In both settings, the proposed method is also compared to the method of Erichson et al. (2020), here denoted by SPCA, as implemented in the package **sparsepca** (Erichson et al. 2018).

We provide results on two simulation exercises, one investigating the runtime of several methods, and a second one to compare the performance of different algorithms in terms of subspace estimation and sparsity detection. Each simulation setting is repeated 100 times, and we report averages of the corresponding performance measures for these 100 replications.

#### 4.3.1 Runtime

In addition to the algorithms ROSPCA, MacroPCA, and SPCA, we also include the PP algorithm from the **pcaPP** package (Filzmoser et al. 2022) as another sparse and robust PCA method for high-dimensional data in the comparison. However, this method is not included in the performance study, as the detailed comparisons to ROSPCA in Hubert et al. (2016) resulted in worse performance than the latter. For SCRAMBLE, the wrapping transformation and the Huber loss function were used. Here we use Setting 1 described above, keeping the sparsity parameters for all methods fixed and varying $p$ and $n$.

The average computation time for the 100 simulation replications is presented in Fig. 1. For the left panel, the number of variables is fixed at $p = 10$ and the number $n$ of observations increased, while for the right panel, the number of observations is fixed at $n = 50$ and the number of variables $p$ increased. The tuning of hyperparameters was excluded from the timing.

Figure 1 shows in most situations a higher runtime of the proposed method (SCRAMBLE). This is due to some of the other algorithms, such as PP, or core parts of MacroPCA, being implemented in C++. ROSPCA relies on a combination of the PP approach and robust covariance estimation in the resulting subspace, which per-

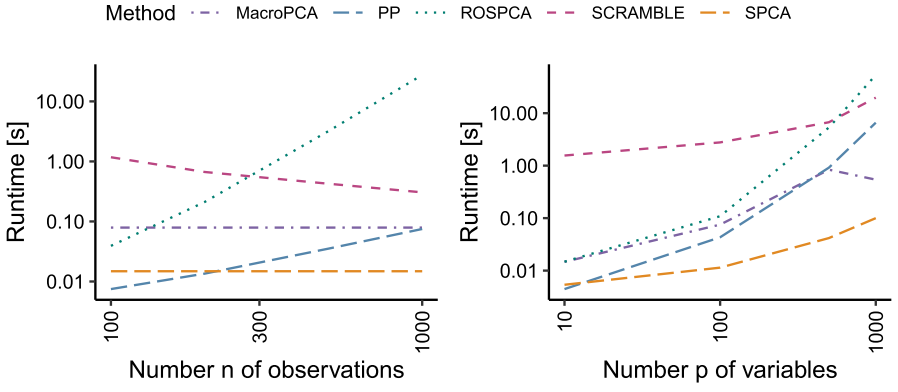


**Fig. 1** Comparison of runtime for the different methods. Shown are average runtimes in seconds for 100 replications. Left: The number of variables is fixed at $p = 10$, the number of observations $n$ is increased from 100 to 1000. Right: The number of observations is fixed at $n = 50$, and the number of variables $p$ is increased from 10 to 1000. For SCRAMBLE, the wrapping transformation and the Huber loss function were used

forms well with moderate dimensions but rapidly increases with the number of observations. Still, with growing $n$ and $p$, the advantage of the proposed method becomes apparent. If the number of observations $n$ increases, the runtime even decreases, as the initial estimate is better, leading to faster convergence. In addition, very large $n$ can also be handled by an appropriate SGD variant, as the data can be processed in batches of suitable sizes, a clear advantage in comparison to the repeated subset evaluations that need to be done for the ROSPCA algorithm, for example. When the number of variables $p$ grows, the approach based on gradient descent also scales better, as several directions can be computed at once and no repeated cycling of candidate directions is necessary as for PP. While the runtime of MacroPCA as another cellwise robust PCA method is appealing, it is not able to include a regularization and produce sparsity in the loadings. The SPCA method has the overall best performance regarding runtime.

The computational complexity of the proposed algorithm depends on three steps: 1. the data transformation, 2. the SVD for the starting value, and 3. the gradient algorithm. For the rank transformation, this results in a complexity of $O(n\log(n) + np\min(n,p) + inp + pk^2)$, and for the wrapping transformation in $O(np + np\min(n,p) + inp + pk^2)$, where $i$ refers to the number of iterations and $k$ to the number of estimated components (refer to Raymaekers and Rousseeuw (2021) for the complexity of the wrapping transformation and Cunningham and Ghahramani (2015) for a discussion of complexity of Riemannian gradient descent). The learning rate was set to $\gamma = 0.001$ with a learning rate decay of 0.99 for batch gradient descent. The simulations were run on an Apple M1 MacBook Pro 2020 (16 GB).

#### 4.3.2 Performance

The robustness of the methods for an increasing proportion of casewise and cellwise outliers is studied. For the proposed method, performance measures of different combinations of the initial transformation and loss function are evaluated and compared to a suitable alternative method.

Figure 2 presents the results for casewise contamination. The performance results for the low-dimensional setting ($p = 10$ and $n = 50$) are shown in the plots on the left side, while those for the high-dimensional setting ($p = 500$ and $n = 100$) are on the right-hand side. In both cases, $k = 2$ is fixed, and ROSPCA and SPCA are compared to different versions of SCRAMBLE, with different loss functions $\rho$ (Huber, Tukey, LTS), see Sect. 2, and different transformations (rank, wrapping) for the initialization, see Sect. 3.2.

In the low-dimensional setting, the SCRAMBLE method clearly performs better than ROSPCA in all performance measures, especially in estimating the sparsity, described by the TNR. The increasing contamination has only little effect on the outcome. Also the different versions of SCRAMBLE show very similar results. SPCA is comparable to SCRAMBLE. For the high-dimensional setting we can see more effects. In the uncontaminated case, SCRAMBLE outperforms ROSPCA, particularly for the TNR. The overall performance of SPCA is very good but SCRAMBLE in combination with the wrapping transformation leads to the overall best results. Interestingly, the TNR improves for ROSPCA with increasing contamination, and

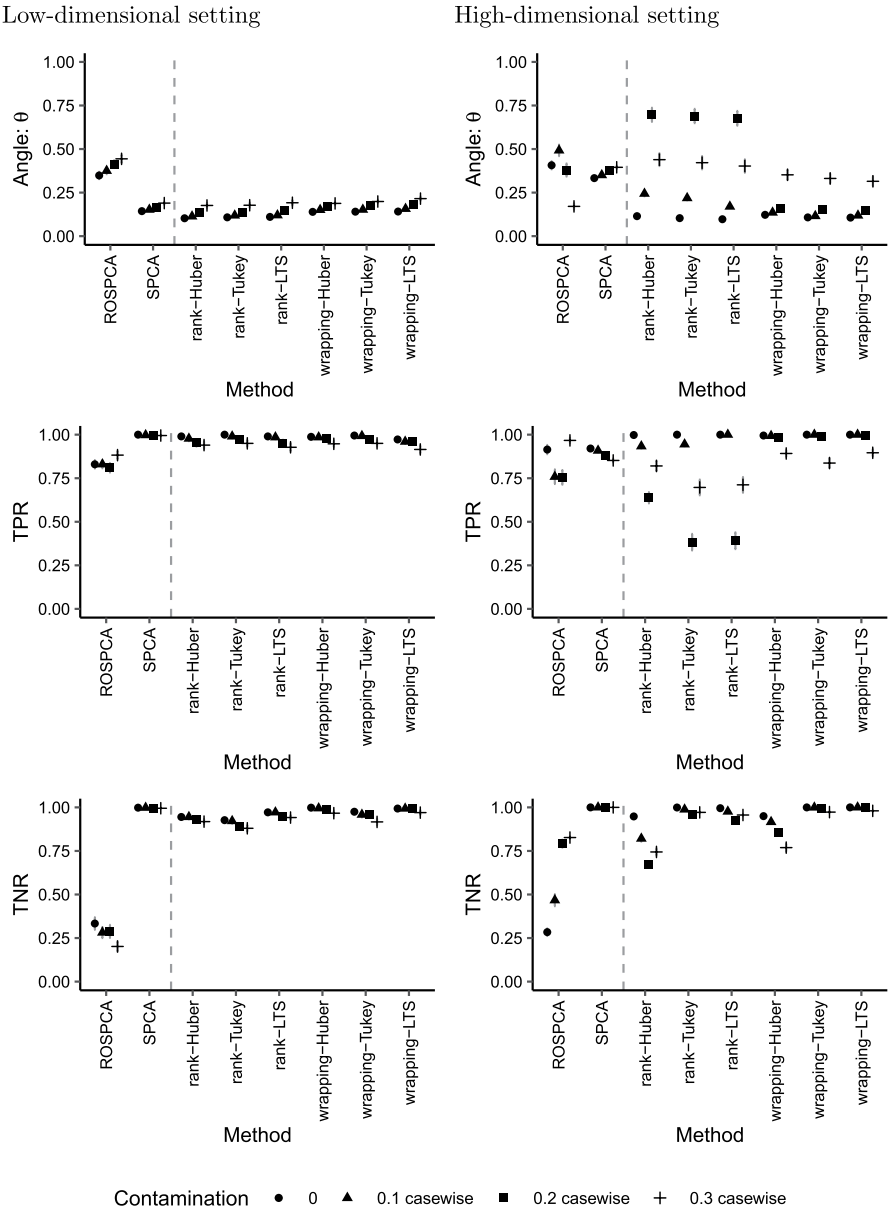


**Fig. 2** Comparison of performance measures for the methods ROSPCA, SPCA and SCRAMBLE (six combinations of loss functions and transformations) for the casewise setting. Shown are averages of the performance measures for 100 replications. Left: low-dimensional ($p = 10$, $n = 50$); right: high-dimensional ($p = 500$, $n = 100$). The different point shapes correspond to different contamination levels: $\epsilon = 0\%, 10\%, 20\%, 30\%$

the same holds for SCRAMBLE based on the rank transformation. However, here an angle of about 0.75 for 20% contamination already suggests a solution very different from the target. The best results are achieved by SCRAMBLE initialized with the wrapping transformation, for the LTS and the Tukey loss function.

The results for the cellwise contamination setting with `gamma` set to 2 are shown in Fig. 3, with the low-dimensional setting on the left and the high-dimensional setting on the right. Here we compare SCRAMBLE to MacroPCA as an alternative algorithm for cellwise robust PCA, although this method does not allow for sparsity. Consequently, MacroPCA fails to accurately predict the sparsity (resulting in a TPR of 1 and a TNR of 0) but yields very good results for the principal angle in the low-dimensional setting. Note that we selected the same proportions of contamination as for the casewise setting. However, in the cellwise setting, these amounts contaminate many more rows of the data matrix, or even all rows (Raymaekers and Rousseeuw 2024), leading to a faster decrease in performance among all metrics. We also compare to SPCA which is an alternative cellwise robust and sparse algorithm. In the low-dimensional setting its performance is similar to SCRAMBLE. For the high-dimensional setting, however, the proposed algorithm yields the best overall results. With increasing contamination, especially the performance of SPCA gets worse quickly. SCRAMBLE can deal with high-dimensional settings as it processes the loss function cellwise, and thus both more rows or more columns yield more training set observations. As expected, there is a certain decrease in performance for increased contamination. In this setting, rank-based initialization is more robust than initialization with the wrapping transformation with default values (Raymaekers and Rousseeuw 2021). The overall best results are in combination with the LTS or Tukey loss function.

The results for the cellwise contamination setting with `gamma` set to 5 are shown in Fig. 4. It can be observed that MacroPCA still yields very good results in terms of robustness, but does not provide sparsity. When we compare the two sparse methods, the results show a clear advantage for SCRAMBLE in comparison to SPCA. When the contamination is set to 10%, SPCA fails to obtain the correct angle and sparsity. The performance of SCRAMBLE also decreases with higher contamination but it can tolerate higher amounts than the alternative. Overall, SPCA is competitive in the casewise and low-dimension scenarios but SCRAMBLE leads to better results in high-dimensional cellwise contaminated settings, especially when the contamination is shifted from the center of the data.

## 5 Illustration on real data

The usefulness of the approach will be demonstrated on two datasets from tribology, the study of friction, wear, and lubrication. The presented data originate from chemical analyses and tribological experiments performed on automotive engine oils after they have been subjected to a varying duration of artificial alteration in the laboratory, see Dörr et al. (2019) and Besser et al. (2019) for a description of the alteration methods. FTIR (Fourier-transform infrared) spectra consist of absorption values that are measured over about 2000 wavenumbers, with distinctive peaks associated with

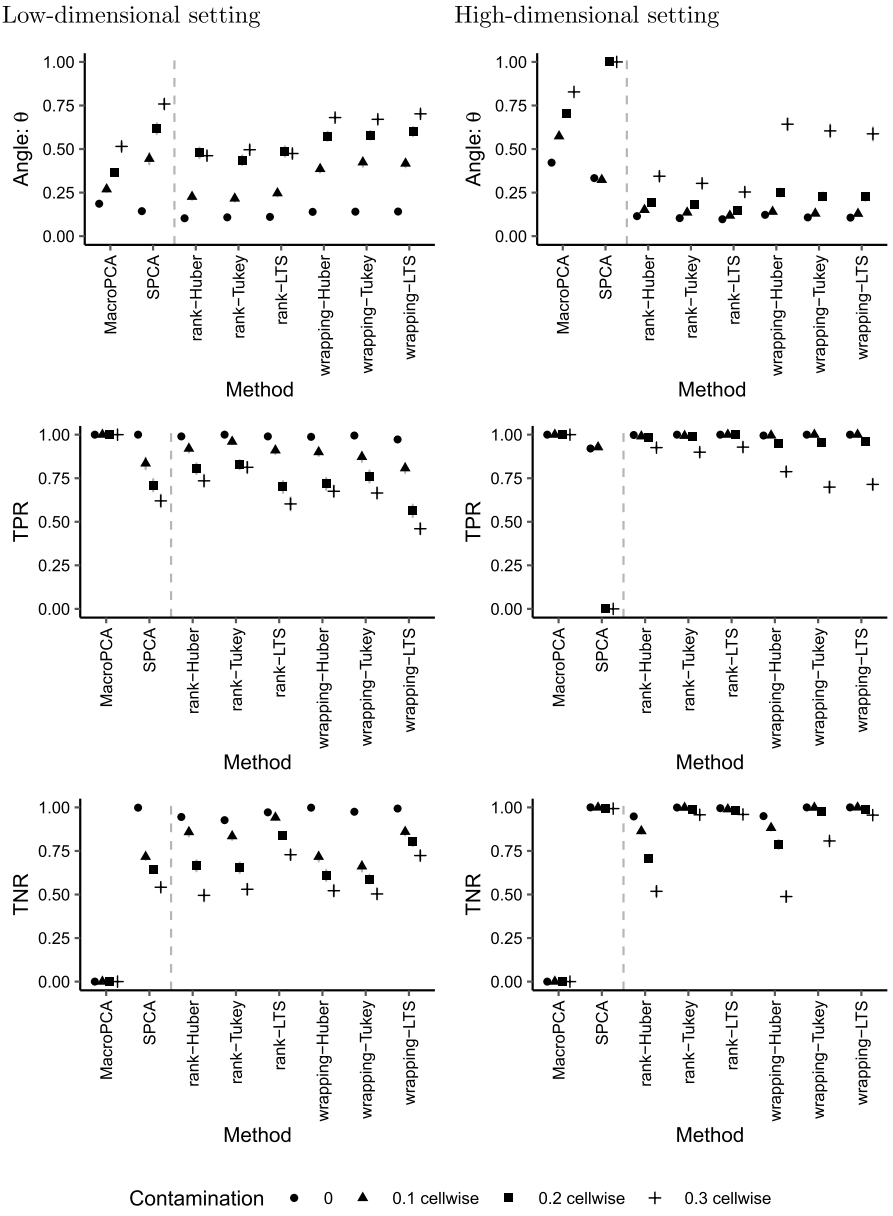


**Fig. 3** Comparison of performance measures for the methods MacroPCA, SPCA and SCRAMBLE (six combinations of loss functions and transformations) for the cellwise setting. Shown are averages of the performance measures for 100 replications for `gamma` = 2. Left: low-dimensional ($p = 10$, $n = 50$); right: high-dimensional ($p = 500$, $n = 100$). The different point shapes correspond to different contamination levels: $\epsilon = 0\%, 10\%, 20\%, 30\%$

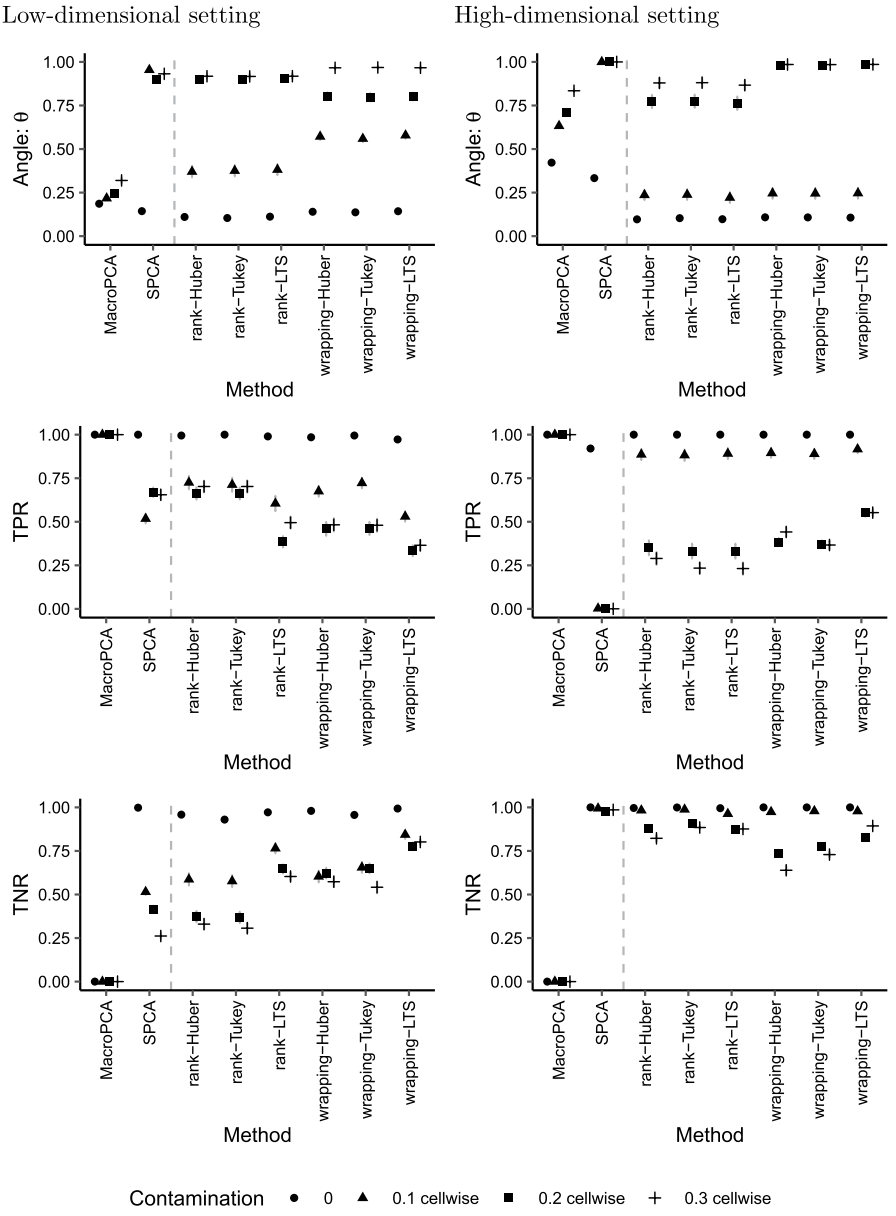


**Fig. 4** Comparison of performance measures for the methods MacroPCA, SPCA and SCRAMBLE (six combinations of loss functions and transformations) for the cellwise setting. Shown are averages of the performance measures for 100 replications for `gamma` = 5. Left: low-dimensional ($p = 10$, $n = 50$); right: high-dimensional ($p = 500$, $n = 100$). The different point shapes correspond to different contamination levels: $\epsilon = 0\%, 10\%, 20\%, 30\%$

certain oil attributes. For this data structure, the sparsity assumption can be justified: It can be assumed that only a small set of wavenumbers is sufficient to explain most of the variability in the dataset (Pfeiffer et al. 2022). Another aspect is lubrication performance, which can be measured on an Schwing-Reib-Verschleiss (SRV) tribometer experiment (a steel ball sliding against a steel disk with the lubricant of interest in between, see Dörr et al. (2019) for a more detailed description of the experiment). One part of the resulting data consists of optical images (taken under the microscope) of the wear scar areas, which yield data matrices with $n \ll p$ when vectorized.

As both types of data are produced in the laboratory, we can expect outliers to be present in the datasets due to possibly high variability following the alteration process and experimental effects. In addition, the experiments are often costly and time-consuming, resulting in much fewer observations than variables and wide data matrices. Thus, dimension reduction is necessary before applying any further analysis. We demonstrate in the following how the proposed method can be applied to perform this dimension reduction via robust PCA and, in addition, yield sparse loadings when appropriate.

### 5.1 FTIR spectra

The presented dataset consists of $n = 50$ FTIR spectra of 10 automotive engine oils. The fresh oils were subjected to a small-scale alteration in the laboratory as described in Dörr et al. (2019). During the alteration, samples were taken regularly and spectra were recorded, each containing the absorbance at $p = 1668$ wavenumbers. The task at hand is to understand which variables contribute most to the variability in the dataset, and often classical PCA is applied, see, for example, Besser et al. (2013). To make it more challenging, we contaminate the dataset with 6 observations originating from a large-scale alteration (Besser et al. 2019) to imitate a scenario when the origin of a sample may not be as clear as in the laboratory setting. Our aim is to identify observations that are different from the majority of the data, and also understand why they are outlying. As mentioned before, sparsity in the PCA loadings is desirable in this setting to enhance interpretability.

The resulting dataset consists of $n = 56$ observations and $p = 1668$ variables. As spectral data are already on the same scale, the data are not scaled, but only column-wise centered with the median before applying the methods. We compare the results from ROSPCA (Hubert et al. 2016) and the proposed SCRAMBLE method with the rank-based data transformation for the starting value and the Huber loss function. Due to the high number of variables in the dataset, the rank-based transformation is advisable. The different loss functions performed similarly, therefore we only show the results for the Huber loss. For both methods, the number of principal components is determined via the cumulative proportion of explained variance. For ROSPCA, this results in $k_{\mathrm{ROSPCA}} = 10$, for SCRAMBLE in $k_{\mathrm{SCRAMBLE}} = 7$ components. Then, hyperparameter optimization for the sparsity parameters is done for both algorithms. For SCRAMBLE, this is done using Bayesian optimization and the TPO criterion; for RSPCA, the grid search and BIC type criterion proposed by Hubert et al. (2016) is used. Figure 5 shows the original FTIR spectra in gray, together with the first (left plot) and second (right plot) loadings vectors of both methods. We find that

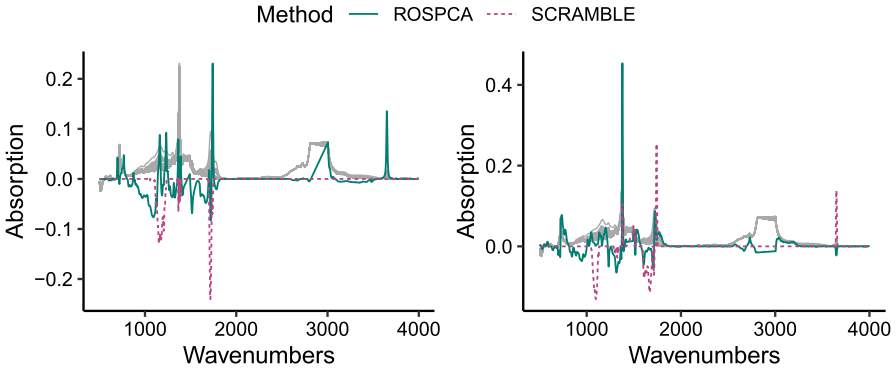


**Fig. 5** FTIR spectra of the original data, shown in gray, together with the first (left plot) and second (right plot) loadings vectors of each method

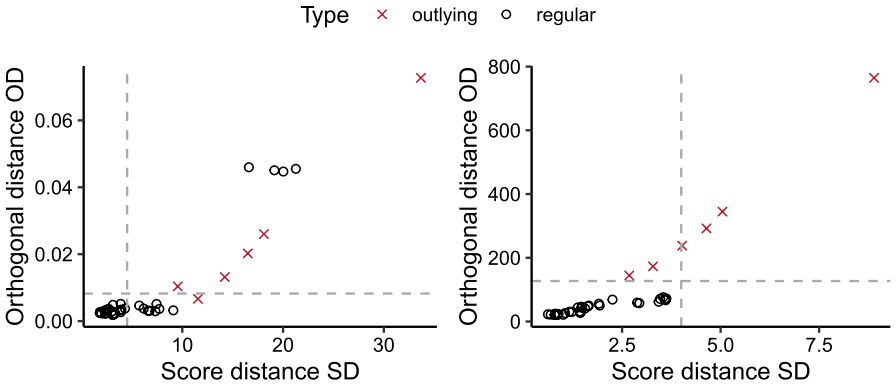


**Fig. 6** Score distance versus orthogonal distance for the FTIR spectra. Results for ROSPCA (left) and SCRAMBLE (right)

SCRAMBLE leads to more sparsity, and therefore to results that are easier to interpret. Non-zero loadings refer to wavelengths and thus to chemical compounds that are associated with underlying chemical processes during oil alteration. This information is highly relevant for practitioners, and we can find variables associated with selected variables such as oxidation, conventionally evaluated at wavenumber 1720 $cm^{-1}$, or phenolic antioxidants at 3650 $cm^{-1}$ (Besser et al. 2019).

Figure 6 shows PCA diagnostic plots based on the score distance SD and orthogonal distance OD, with the outlier cutoff values as dashed lines, see Hubert et al. (2005). A high value of the OD means that the observations are far away from the estimated PCA subspace. The left plot presents the results for ROSPCA, and here almost all outliers, i.e., the observations from large-scale alteration, are identified with high OD values. However, also four regular observations yield high OD values. The right plot for the SCRAMBLE results corresponds to what we would expect.

In order to investigate the differences between ROSPCA and SCRAMBLE in more detail, we show plots of the standardized residuals in Fig. 7 for a selected

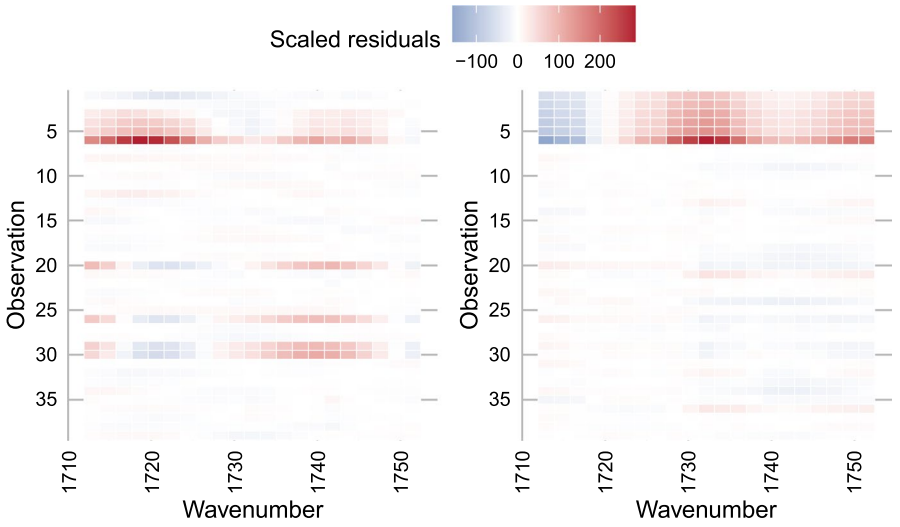


**Fig. 7** Residual plots for a selected range of wavenumbers and for a subset of the observations. Left: ROSPCA residuals; right: SCRAMBLE residuals

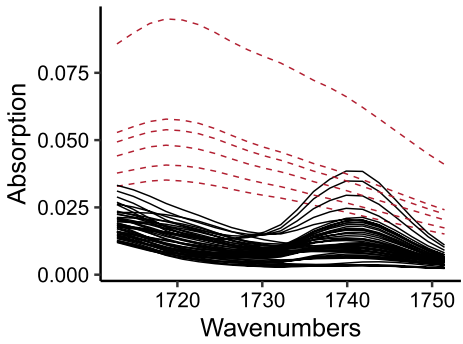


**Fig. 8** Zoomed-in view of the FTIR spectra from Fig. 5 for the selected wavelength range shown in Fig. 7. The red dashed lines are outliers with large-scale alteration

wavenumber range and a subset of the observations. In fact, this wavenumber range mainly determines PC1 of SCRAMBLE (negative) and ROSPCA (positive), and refers to oxidation effects (see Fig. 5). The left plot is for ROSPCA, the right plot is for SCRAMBLE, and each tile represents one element in the scaled residual matrix, with color according to the legend. The residuals were standardized robustly using the median and mad of the residual matrix. Note that the residuals scale is very small leading to very large scaled residuals for some cells. The first six rows correspond to the FTIR spectra of oils from a different alteration process, and SCRAMBLE clearly reconstructs these worst, meaning they are not as influential to the fit of the PCA subspace. In ROSPCA, on the other hand, only observation 6 is the only clearly visible observation out of the outlier subset, and four further observations also show larger residuals. A look at the original spectra with a zoomed-in view into this wavenumber range in Fig. 8 explains this behavior: There, the outliers (first 6 rows) are shown by red dashed curves, and only one (observation 6) is clearly further away, while

the other outliers partially overlap with regular observations. This overlap around wavenumber 1740 $cm^{-1}$, thus in a very restricted range, is the reason why four regular observations are falsely classified as outlying by the ROSPCA algorithm. The affected wavenumbers lie in the absorption band of oxidation products, ranging from 1860–1660 $cm^{-1}$, and are of interest in conventional analysis of FTIR spectra (Besser et al. 2019; Pfeiffer et al. 2022). A cellwise robust method can assist the practitioner in finding and understanding the differences between outlying and regular observations.

In summary, we can see the benefit of a cellwise robust method in contrast to only casewise robust estimation. As the differences only show at certain peaks in the spectra, a cellwise robust method does not lose as much information as a casewise method, resulting in a better fit with fewer components. In addition, we can also identify the variables which contribute most to the outlyingness.

### 5.2 Tribology: wear scar images

In this example we are interested in the question whether the degradation visible in wear scar images can be used to predict the alteration duration of the oils. As the image information is high-dimensional, we will demonstrate the usefulness of SCRAMBLE in a PCR (Principal Component Regression) setting, thus reducing the dimensionality of the image information before performing regression.

In Pfeiffer and Filzmoser (2023), image features were derived from a similar image dataset, before robust regression methods were applied. For this demonstration, we derive robust features via SCRAMBLE directly from the vectorized images before applying a robust regression on the resulting principal components. We do not use a sparsity-inducing regularization, as this has been found to not provide an advantage for image data (Pfeiffer and Filzmoser 2023). In the given setting, $n = 220$ gray-scale images of size $64 \times 64$, resulting in vectors of size $64^2 = 4096$, together with a response variable containing the alteration duration (in hours) of the lubricant used in the SRV experiment, are available. After the removal of all constant columns, $p = 3025$ columns are left. As the response variable, the alteration duration is given in hours, which is square-root transformed before estimating the model.

We compare classical PCA via SVD to the SCRAMBLE algorithm with rank-based preprocessing and the Huber loss. Therefore, the dataset is randomly split into a 70% training, a 20% validation, and a 10% test set. The principal components are estimated on the training set, and then the optimal number of components is evaluated for the validation set via the mean squared error of prediction (MSEP) using least-squares regression for the classic estimation and robust regression (the function `lmrob()` from the **robustbase** R package (Maechler et al. 2024)) for robust estimation. Finally, the MSEP is computed for the test set. The first three estimated loadings are shown in Fig. 9 for classical PCA and in Fig. 10 for robust PCA using the SCRAMBLE algorithm. We can observe that the first loadings look quite similar, while the order is different. Both methods distinguish between the border and the interior of the wear scar, as well as the overall contributions. This is also how we would interpret the principal components, which is possible even though sparsity

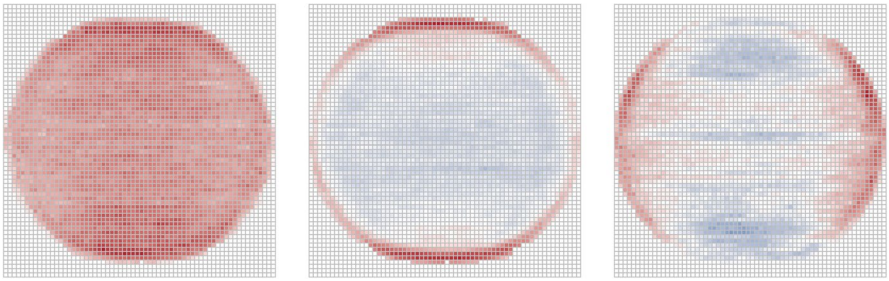

**Fig. 9** Classical loadings back-transformed to the image space

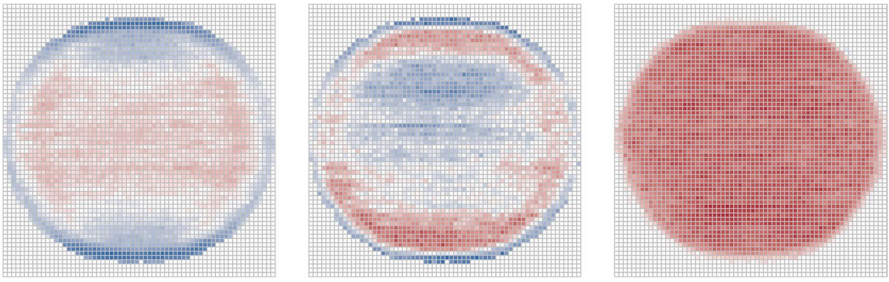

**Fig. 10** Robust loadings obtained from SCRAMBLE back-transformed to the image space

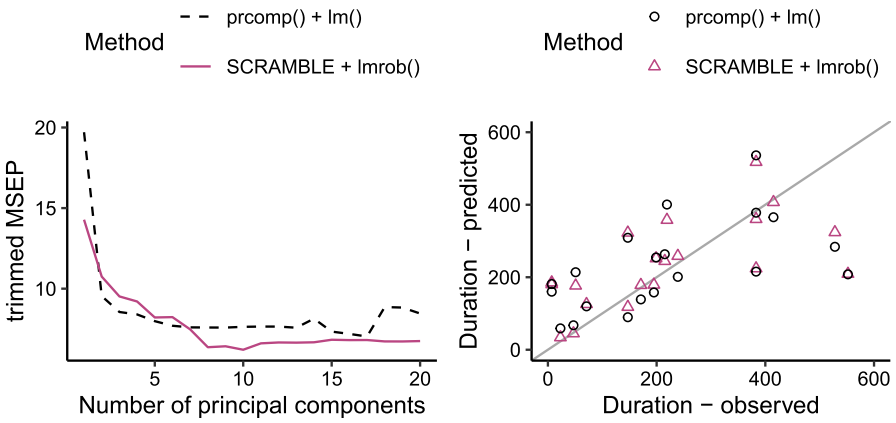


**Fig. 11** Left: Selection of best number of PCs based on 10% trimmed MSEP computed on the validation set. Right: Observed versus predicted alteration duration for the classical and robust approach

has not been used here. Note that the pixels on the boundary have not been used as predictor information.

The 10% trimmed MSEP for the validation set based on different numbers of components is shown on the left side in Fig. 11 for both classical and robust PCR. For classical PCR, the error starts at a higher level, possibly indicating that the directions

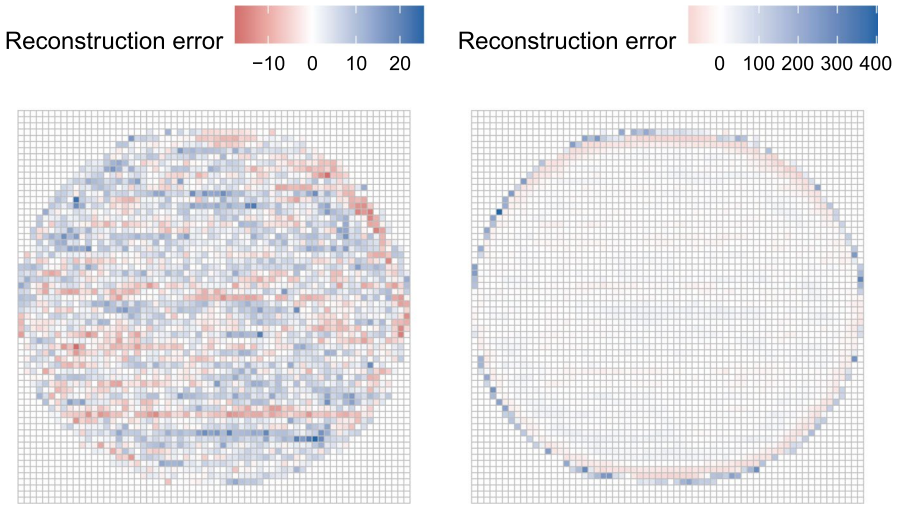


**Fig. 12** Reconstruction errors per variable, visualized in the image dimensions. Left: PCA via classical SVD, right: PCA via SCRAMBLE. The number of components corresponds to the number leading to the minimum trimmed MSEP for PCR

of the first few components are influenced by outliers, and thus they are not as effective for prediction.

For the optimal number of components, we select that number yielding the smallest trimmed MSEP, resulting in $k_{\text{classical}} = 17$ and $k_{\text{SCRAMBLE}} = 10$ components. Thus, the robust method leads to a smaller number of components, and also to a smaller prediction error.

In the right plot of Fig. 11, the observed and predicted values of the alteration duration are shown for both methods for the test set observations. The predictions for higher values of duration for both methods are worse than for values in the beginning or middle of the duration range. In total, the model based on robust PCR performs better, with a trimmed MSEP of 13.76 for the test set observations, while classical PCR leads to an error of 17.88 (Fig. 11).

Using this number of components, we can also reconstruct the data matrices and analyze the reconstruction errors. In Fig. 12, the reconstruction errors per variable (pixel) are visualized for both the classical PCA (on the left) and SCRAMBLE (right). While for the classical method, no structure is left in these residuals, it is clearly visible that for the robust method, the border of the wear scars is not reconstructed well. This also makes sense because the borders of the balls in the wear scar images are not identical. In fact, the size of the balls in the image can slightly change due to the nature of the experiment, as the balls are placed manually under a microscope, but also due to preprocessing and cutting the images. Obviously, for the cellwise robust procedure, this change in size is not relevant for prediction, whereas the classical model takes this variability into account.

While the results for classical and robust PCR are not very different, the example still illustrates that the robust method is able to perform better for the majority of the data (in the middle of the duration range), while the classical predictions are influenced

by more extreme values. Furthermore, we can use robust diagnostics to get further insight into why the prediction quality between certain values of the response differs.

## 6 Discussion and summary

In recent years, cellwise robust methods are becoming increasingly important. This is mainly due to the increased occurrence of high-dimensional data, as a result of modern measurement methods and devices. With high-dimensional data it becomes more likely that an observation contains outliers in single variables, and traditional rowwise (casewise) methods would no longer work if the majority of observations are contaminated. This is also an issue for principal component analysis (PCA), where rowwise robust methods could fail in the presence of many cellwise outliers.

One could think of several different approaches to obtain a cellwise robust PCA method. A first idea could be the identification of cellwise outliers and the replacement of those cells by values which would be expected according to some distributional assumptions (Rousseeuw and Bossche 2018). With the cleaned data matrix one could proceed with classical PCA. Even in the casewise robust setting, the approach to detect and correct outlying observations prior to classical PCA would be a way to obtain a casewise robust PCA version. However, outlier detection assumes an underlying model, usually a multivariate normal distribution, and outlier detection/correction identifies/corrects observations or cells according to this model assumption. In cellwise or casewise robust PCA, on the other hand, we are not limited to this distribution. Particularly for PCA based on low-rank approximation, the interest is rather in a robust data reconstruction, where the error loss function utilizes information from the single variables rather than from the joint multivariate distribution, see Equation (4).

Another approach to cellwise robust PCA is to use a plug-in estimator for the covariance to determine the principal components, for example, the cellwise robust Minimum Covariance Determinant (MCD) estimator (Raymaekers and Rousseeuw 2023b). This is the cellwise equivalent to a rowwise robust PCA version where a rowwise robust covariance estimator, such as the MCD (Rousseeuw 1985) is plugged in. While such a procedure is straightforward to implement, it might not be so clear how to include sparsity.

Sparsity, or the natural requirement of interpretability of the principal components is especially important in a high-dimensional setting. For that reason, sparse (Jolliffe et al. 2003) and sparse robust (Croux et al. 2013) PCA versions were introduced which maximize the variance of the components subject to an $L_1$ penalty on the loadings vectors. The gain in sparsity or explainability leads to a loss in explained variance, and this compromise can be formalized by an appropriate objective function (Croux et al. 2013).

We have introduced a cellwise robust and sparse PCA method using low-rank matrix approximation. The objective function can be formulated in a very natural way (Maronna and Yohai 2008; Croux et al. 2013), and it combines a robust loss function for the reconstruction error of all cells of the data matrix with an elastic net penalty on the loadings. The specific choice of the loss function determines the robustness properties of the PCA solution (Maronna and Yohai 2008). Both the robust

loss function and the incorporation of an $L_1$ or elastic net penalty leads to computational challenges. We have developed an algorithm based on manifold learning to optimize the objective function. The $L_1$ penalty was incorporated by the use of a sparsity-inducing penalty, allowing for an approximation by a differentiable function. The choice of appropriate starting values is important, and we compared different approaches. Overall, the algorithm leads to an efficient computation, even for high dimensions $p$ and many observations $n$. Simulations have demonstrated that the resulting method, called SCRAMBLE (Sparse Cellwise Robust Algorithm for Manifold-based Learning and Estimation), has superior properties when compared to alternative robust PCA approaches, both in the casewise and cellwise case for high-dimensional settings.

We applied the suggested method to two real data examples from tribology and compared the performance with existing estimators, illustrating the usefulness of a cellwise robust and sparse PCA method.

Possible extensions to groupwise PCA or robust data imputation are possible via a modification of the objective function (8). Furthermore, theoretical robustness properties like the influence function and breakdown point (Maronna et al. 2019) would be interesting topics for future research.|

## 7 Supplementary material

The proposed procedure is implemented in the R package `scramble`, which is available from https://github.com/piapfeiffer/scramble. Replication files are available from https://github.com/piapfeiffer/scramble_replication_files. The data used in the examples is provided in the repository with the replication files.

**Acknowledgements** The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

**Funding** Open access funding provided by TU Wien (TUW). This work was funded by the Austrian COMET-Program (project InTribology2, No. 906860) via the Austrian Research Promotion Agency (FFG) and the federal states of Niederösterreich and Vorarlberg and was carried out at the Austrian Excellence Centre of Tribology (AC2T research GmbH) and the TU Wien. Laura Vana-Gür acknowledges funding from the Austrian Science Fund (FWF) for the project "High-dimensional statistical learning: New methods to advance economic and sustainability policies" (ZK 35), jointly carried out by WU Vienna University of Economics and Business, Paris Lodron University Salzburg, TU Wien, and the Austrian Institute of Economic Research (WIFO).

**Data availability** The scramble R package is available on https://github.com/piapfeiffer/scramble. Therep lication files are available on https://github.com/piapfeiffer/scramble_replication_files including the datas ets used in this study and the code to generate the simulated data.

### Declarations

**Conflict of interest** The authors have no conflict of interest to declare that are relevant to the content of this article.

**Ethical approval** This study uses simulated data and does not involve human participants.

**Informed consent** Not applicable.



## References

Agostinelli C, Leung A, Yohai VJ, Zamar RH (2015) Robust estimation of multivariate location and scatter in the presence of cellwise and casewise contamination. TEST 24:441–461. https://doi.org/10.1007/s11749-015-0450-6

Alqallaf F, Aelst S, Yohai VJ, Zamar RH (2009) Propagation of outliers in multivariate data. Ann Stat. https://doi.org/10.1214/07-AOS588

Besser C, Dörr N, Novotny-Farkas F, Varmuza K, Allmaier G (2013) Comparison of engine oil degradation observed in laboratory alteration and in the engine by chemometric data evaluation. Tribol Int 65:37–47. https://doi.org/10.1016/j.triboint.2013.01.006

Besser C, Agocs A, Ronai B, Ristic A, Repka M, Jankes E, McAleese C, Dörr N (2019) Generation of engine oils with defined degree of degradation by means of a large scale artificial alteration method. Tribol Int 132:39–49. https://doi.org/10.1016/j.triboint.2018.12.003

Björck Å, Golub GH (1973) Numerical methods for computing angles between linear subspaces. Math Comput 27(123):579–594. https://doi.org/10.2307/2005662

Bonnabel S (2013) Stochastic gradient descent on riemannian manifolds. IEEE Trans Autom Control 58(9):2217–2229. https://doi.org/10.1109/TAC.2013.2254619

Candès EJ, Li X, Ma Y, Wright J (2011) Robust principal component analysis? J ACM (JACM) 58(3):1–37

Centofanti F, Hubert M, Rousseeuw PJ (2024) Robust principal components by casewise and cellwise weighting. arXiv preprint arXiv:2408.13596

Centofanti F, Hubert M, Rousseeuw PJ (2025) Cellwise and casewise robust covariance in high dimensions. arXiv preprint arXiv:2505.19925

Croux C, Ruiz-Gazen A (2005) High breakdown estimators for principal components: the projection-pursuit approach revisited. J Multivar Anal 95(1):206–226. https://doi.org/10.1016/j.jmva.2004.08.002

Croux C, Filzmoser P, Fritz H (2013) Robust sparse principal component analysis. Technometrics 55(2):202–214. https://doi.org/10.1080/00401706.2012.727746

Cunningham JP, Ghahramani Z (2015) Linear dimensionality reduction: survey, insights, and generalizations. J Mach Learn Res 16(1):2859–2900 (https://jmlr.org/papers/v16/cunningham15a.html)

De La Torre F, Black MJ (2003) A framework for robust subspace learning. Int J Comput Vis 54:117–142

Dörr N, Brenner J, Ristić A, Ronai B, Besser C, Pejaković V, Frauscher M (2019) Correlation between engine oil degradation, tribochemistry, and tribological behavior with focus on zddp deterioration. Tribol Lett 67:1–17. https://doi.org/10.1007/s11249-019-1176-5

Eckart C, Young G (1936) The approximation of one matrix by another of lower rank. Psychometrika 1(3):211–218. https://doi.org/10.1007/BF02288367

Edelman A, Arias TA, Smith ST (1998) The geometry of algorithms with orthogonality constraints. SIAM J Matrix Anal Appl 20(2):303–353. https://doi.org/10.1137/S0895479895290954

Erichson NB, Zheng P, Manohar K, Brunton SL, Kutz JN, Aravkin AY (2020) Sparse principal component analysis via variable projection. SIAM J Appl Math 80(2):977–1002

Erichson NB, Zheng P, Aravkin S (2018) **sparsepca**: sparse principal component analysis (SPCA). https://CRAN.R-project.org/package=sparsepca. R package version 0.1.2

Filzmoser P, Fritz H, Kalcher K (2022) **pcaPP**: robust PCA by Projection Pursuit. https://CRAN.R-project.org/package=pcaPP. R package version 2.0-3

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. http://www.deeplearningbook.org

Greco L, Farcomeni A (2016) A plug-in approach to sparse and robust principal component analysis. TEST 25:449–481. https://doi.org/10.1007/s11749-015-0464-0
Hartley R, Zisserman A (2003) Multiple view geometry in computer vision. Cambridge University Press
Hotelling H (1933) Analysis of a complex of statistical variables into principal components. J Educ Psychol 24(6):417. https://doi.org/10.1037/h0071325
Hubert M, Rousseeuw PJ, Branden KV (2005) ROBPCA: a new approach to robust principal component analysis. Technometrics 47(1):64–79. https://doi.org/10.1198/004017004000000563
Hubert M, Reynkens T, Schmitt E, Verdonck T (2016) Sparse PCA for high-dimensional data with outliers. Technometrics 58:424–434, 10. https://doi.org/10.1080/00401706.2015.1093962
Hubert M, Rousseeuw PJ, Bossche W (2019) MacroPCA: an all-in-one pca method allowing for missing values as well as cellwise and rowwise outliers. Technometrics 61(4):459–473. https://doi.org/10.1080/00401706.2018.1562989
Jolliffe IT, Trendafilov NT, Uddin M (2003) A modified principal component technique based on the LASSO. J Comput Graph Stat 12(3):531–547. https://doi.org/10.1198/1061860032148
Maechler M, Rousseeuw P, Croux C, Todorov V, Ruckstuhl A, Salibian-Barrera M, Verbeke T, Koller M, Conceicao ELT, Anna di Palma M (2024) robustbase: basic robust statistics. http://robustbase.r-forge.r-project.org/. R package version 0.99-2
Maronna RA, Yohai VJ (2008) Robust low-rank approximation of data matrices with elementwise contamination. Technometrics. https://doi.org/10.1198/004017008000000190
Maronna RA, Martin RD, Yohai VJ, Salibián-Barrera M (2019) Robust statistics: theory and methods (with R). John Wiley & Sons
Öllerer V, Croux C, Alfons A (2015) The influence function of penalized regression estimators. Statistics 49(4):741–765. https://doi.org/10.1080/02331888.2014.922563
Pearson K (1901) On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 2(11):559–572
Pfeiffer P, Filzmoser P (2023) Robust statistical methods for high-dimensional data, with applications in tribology. Anal Chim Acta 1279:341762. https://doi.org/10.1016/j.aca.2023.341762
Pfeiffer P, Ronai B, Vorlaufer G, Dörr N, Filzmoser P (2022) Weighted lasso variable selection for the analysis of FTIR spectra applied to the prediction of engine oil degradation. Chemom Intell Lab Syst 228:104617. https://doi.org/10.1016/j.chemolab.2022.104617
Raymaekers J, Rousseeuw P (2023) cellWise: analyzing data with cellwise outliers. https://CRAN.R-project.org/package=cellWise. R package version 2.5.3
Raymaekers J, Rousseeuw PJ (2021) Fast robust correlation for high-dimensional data. Technometrics 63(2):184–198. https://doi.org/10.1080/00401706.2019.1677270
Raymaekers J, Rousseeuw PJ (2023) The cellwise minimum covariance determinant estimator. J Am Stat Assoc. https://doi.org/10.1080/01621459.2023.2267777
Raymaekers J, Rousseeuw PJ (2024) Challenges of cellwise outliers. Econ Stat. https://doi.org/10.1016/j.ecosta.2024.02.002
R Core Team (2024) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
Reynkens T (2018) **rospca**: robust Sparse PCA using the ROSPCA Algorithm. https://CRAN.R-project.org/package=rospca. R package version 1.0.4
Rousseeuw PJ (1985) Multivariate estimation with high breakdown point. Math Stat Appl 8(283–297):37
Rousseeuw PJ, Bossche W (2018) Detecting deviating data cells. Technometrics 60(2):135–145. https://doi.org/10.1080/00401706.2017.1340909
Rousseeuw PJ, Croux C (1993) Alternatives to the median absolute deviation. J Am Stat Assoc 88(424):1273–1283. https://doi.org/10.1080/01621459.1993.10476408
She Y, Li S, Dapeng W (2016) Robust orthogonal complement principal component analysis. J Am Stat Assoc 111(514):763–771. https://doi.org/10.1080/01621459.2015.1042107
Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst 25
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Stat Methodol 58(1):267–288
Wilson S (2022) ParBayesianOptimization: parallel Bayesian optimization of hyperparameters https://CRAN.R-project.org/package=ParBayesianOptimization. R package version 1.2.6
Zou H, Hastie T (2003) Regression shrinkage and selection via the elastic net, with applications to microarrays. J R Stat Soc Ser B Stat Methodol 67:301–20
Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B Stat Methodol 67(2):301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Graph Stat 15(2):265–286. https://doi.org/10.1198/106186006X113430