Erasmus University Rotterdam

Erasmus School of Economics

MacroPCLR: A New Principal Component Regression Method Using MacroPCA that Accounts for Missing Values as Well as Rowwise and Cellwise Outliers

Author: Torsten Ruiter

Student Number: 512473

Supervised by Dr. Aurore Archimbaud

Second Assessor: Dr. Hong Deng

July 3, 2022

Abstract

Principal component regression (PCR) combines principal component analysis (PCA) with a regression. This is useful for forecasting with a high-dimensional dataset which is an increasingly demanded task. However, Classical PCR cannot deal with missing values and is sensitive to outliers. Therefore, a new PCR method is introduced in this paper called MacroPCLR. This method is based on the MacroPCA method introduced in Hubert et al. (2019). MacroPCA is a new PCA method that can deal with missing values and is robust to rowwise as well as cellwise outliers. By using this method as a basis, MacroPCLR also inherits these features. This new MacroPCLR method is compared to other PCR methods for several contaminated datasets. The comparison is performed by making use of the mean squared error (MSE). Additionally, the PCA methods that these PCR methods inherit were also compared by the MSE. Results show that of the PCA methods, MacroPCA performs the best when any outliers are present. If only missing values are present, it is slightly better to use Iterative Classical PCA. Concerning the PCR methods, it was shown that MacroPCLR outperforms other PCR method, regardless of the contamination structure of the data.

Contents

1 Introduction

The Linear regression (LR) model is one of the cornerstones of machine learning and is applicable in many disciplines. These models are used to measure the linear relationship between one dependent variable and one or more explanatory variables. The standard notation of the LR model can be seen below:

$$y_i = \beta_1 + \sum_{j=2}^{k} \beta_j x_{ji} + \epsilon_i \text{ for } i = 1, ..., n. \tag{1}$$

Here, every i represents a single observation. The dependent variable is denoted by y and the explanatory variables are denoted by the x_j. β_j denotes the coefficient of the j-th explanatory variable and ϵ represents the error terms. n is the number of observations and k is the number of explanatory variables. For a more in depth explanation consult Heij et al. (2004) (Section 3.1). The more simple matrix form of this model is as follows:

$$y = X\beta + \epsilon. \tag{2}$$

The resulting output of a LR model can be used for forecasting. Well constructed LR models can thus have numerous practical applications. For instance, the LR model developed in Ramanathan (2012) is able to forecast the demand for soda drinks. This model allows soda drink companies to make better managerial decisions and potentially increases company profits. However, present day databases are often high-dimensional according to Agarwal et al. (2021). There are two main problems with high-dimensional data in the LR framework according to Abdi (2010). The first problem is that this makes it more likely that the number of explanatory variables exceeds the number of observations. The second problem is that it increases the chance that different explanatory variables are correlated, which would result in multicollinearity. Both problems cause the methods used to estimate the model in equation (2), to break down.

A solution to these problems is to first apply principal component analysis (PCA) to the matrix of explanatory variables X before performing the actual regression. This solution is suggested by Perez (2017) as well as Abdi (2010). PCA is a dimension reduction method that constructs new variables, principal components (PCs), from the original data. These PCs are linear combinations of the original variables and are constructed in such a way that they are orthogonal to each other. They are also constructed with the objective to explain as much variance as possible, while limiting the number of PCs. This means that applying PCA on a high-dimensional dataset will result in a small number of uncorrelated variables that still instantiate much variability of the original data. For a more in depth explanation and some examples see Abdi and Williams (2010).

Applying PCA on the data matrix X and consequently performing a linear regression is called principal component regression (PCR). The most simple version of PCR is called Classical Principal Component Regression (CPCR), which combines Classical Principal Component Analysis (CPCA) with a LR. CPCR is neatly described in Agarwal et al. (2021). In CPCR, the PCA stage and the regression stage are not interconnected; for this reason Hubert and Verboven (2003) identify two stages within PCR. Namely, the PCA stage and the regression stage.

While CPCR is an elegant and simple procedure, there are some drawbacks to it. For the majority, these drawbacks are located in the PCA stage due to the use of CPCA. According to Hubert et al. (2019), the drawbacks of CPCA are its inability to handle missing values and its sensitivity to outliers. The latter drawback means that the method is not robust. More precisely, if you want to apply strictly CPCA on a dataset containing missing values and outliers this has the following consequences. First, all rows containing at least one missing value have to be removed. This will likely leave the user with either an insufficient sample size, or a biased sample. Second, the sensitivity to outliers means that even a single outlier will cause large biases. In theory, these drawbacks can thus be fixed by replacing CPCA in the PCA stage of CPCR with a PCA method that does account for missing values and outliers. The goal of this research is to find out if this premise holds in practice. The seemingly best candidate for replacing CPCA is the MacroPCA method from Hubert et al. (2019) as it accounts for missing values as well as rowwise and cellwise outliers. The new method that is created by embodying this combination will be called MacroPCLR.

Therefore, the main research question of this paper is the following: *How does the forecasting accuracy of MacroPCLR perform compared to other PCR methods in the presence of outliers and missing values?*

The answer to this question should be of interest to anyone trying to implement PCR or a LR on a high-dimensional dataset. To help answer the main research question two sub-questions are introduced and discussed.

The first sub-question is: *How can outliers and missing values be appropriately dealt with within PCR?* As mentioned earlier, the presence of outliers and missing values creates certain drawbacks for PCR. Therefore, it is useful to investigate how one can deal with these problems appropriately.

The second sub-question is: *Which evaluation criteria most accurately assesses the forecasting performance of PCR models in the presence of outliers and missing values?*

The aim of this research is to compare several PCR models, where the PCA stage of each model differs. To make this comparison one or more evaluation criteria have to be selected. The best

evaluation criterion to use will most accurately translate the results of comparing these different methods.

To answer the research question, some other PCA methods in the literature will be selected. This way, the difference in forecasting performance when implementing MacroPCA can be assessed. However, it is preferable that these PCA methods fix at least one of the drawbacks of CPCA. This is because these methods might perform better in certain contamination structures. A contamination structure is a certain combination of missing values and outliers. After these methods have been deduced from the literature, they have to be appropriately compared. To do this, one or more evaluation criteria have to be selected. These criteria will be based on a discussion of existing evaluation criteria in the literature. The chosen PCA methods are the Iterative Classical Principal Component Analysis (ICPCA) and Robust Principal Component Analysis accounting for Missing values (MROBPCA). The selected evaluation criterion is the mean squared error (MSE).

After other PCA methods have been chosen and the evaluation criteria have been found, a simulation will be executed to create the data that will be used to compare the PCA methods and the newly developed PCR methods.

We found that MacroPCA is the best PCA method to use in all scenarios where outliers are present, however if only missing values are present in the data one should use Iterative Classical Principal Component Analysis. We also found that the newly developed MacroPCLR method trumps other PCR methods in the presence of missing values as well as rowwise and cellwise outliers.

Thus, this paper contributes to the existing literature by developing a new PCR method called MacroPCLR which accounts for missing values as well as rowwise and cellwise outliers. Additionally, MacroPCLR is compared to other PCR methods which gives more insight into its capabilities.

This paper consist of 6 sections and an appendix. In section 2 several PCA methods in the literature that deal with missing values and outliers are mentioned. Additionally, potential evaluation criteria that can be found in the literature are discussed. In section 3, the methods used to conduct our research are explained and the setup of our research is specified further. In section 4, the results of our research are presented. In section 5 we draw some conclusions from our research and explain them. Finally, in section 6 we present a short discussion of results and its implications.

2 Literature Review

In this section, a review of the existing literature based on PCR with the presence of outliers and missing values takes place, by focusing on the PCA stage of PCR.

2.1 Outliers and Missing Values

Outliers and missing values both have an undesirable impact on the outcome of PCR. However, as the nature of their impact differs they should be treated in a slightly different manner.

Missing Values

The property that defines missing values is binary, a measurement is either missing or not. However, there are several classifications that can be assigned to missing values. An overview of these classifications is given in Acock (2005). As MacroPCA was developed under the assumption of Missingness Completely At Random (MCAR), we will focus on this type of missing value as well. MCAR means that the missing values present in the data are randomly distributed. More precisely, if one were to assign a certain number of missing values to the data, each cell would be equally likely to be selected.

If one insists on using CPCA in the presence of missing values, all rows with at least one missing value require removal according to Nelson et al. (1996). Therefore, other methods had to be developed to deal with missing values. The most simple method, which even allows the use of CPCA is called mean imputation. Here missing values are imputed by the mean of the variable they belong to. However, this method will likely give biased results according to Severson et al. (2017) and is thus undesirable. To overcome this bias Nelson et al. (1996) and Kiers (1997) developed an iterative procedure to deal with missing values called ICPCA. The method proposed in Grung and Manne (1998), deals with missing values by using least-squares to minimize a loss function. Oba et al. (2003) uses Bayesian statistics to account for overfitting of missing value estimations.

In conclusion, there is an abundance of methods that deal with missing values within PCA. Therefore, Severson et al. (2017) suggests to consider the type of missing values before choosing which PCA method to use. Since this will affect which method will yield the most accurate results.

Outliers

Grubbs (1969) defines an outlier as "An observation that deviates markedly from other members of the sample in which it occurs." According to Hubert et al. (2019), one can identify two types of outliers; rowwise and cellwise outliers. A cellwise outlier is a cell which holds a value that is considered outlying within the corresponding column. A rowwise outlier is a row that contains

many cellwise outliers. While rowwise outliers have been under investigation since the 1960s according to Maronna et al. (2019a), cellwise outliers were first properly considered in Alqallaf et al. (2009). Therefore, many PCA methods that are robust to rowwise outliers have been developed, while PCA methods robust to cellwise outliers are lacking. Following are some instances of PCA methods that are robust to rowwise outliers. Croux and Haesbroeck (2000) proposed to use the eigenvectors and eigenvalues of robust estimates of the covariance matrix of the data. However, according to Hubert et al. (2005) this method cannot deal with high-dimensional data. Li and Chen (1985) introduced a robust PCA method that can deal with high-dimensional by incorporating projection pursuits. A disadvantage of the projection-pursuit method is its inaccuracy for high-dimensional data. Therefore, Hubert et al. (2005) combined the methods from Croux and Haesbroeck (2000) and Li and Chen (1985) to develop the ROBPCA method. This method can deal with high-dimensional data while remaining accurate. However, in the light of missing values, this method faces the same problems as CPCA. Consequently, Serneels and Verdonck (2008) developed an algorithm which inherits a robust PCA method and uses robust location and scale estimators to handle missing values. This method is called MROBPCA.
All aforementioned robust PCA methods are only robust to rowwise outliers and might break down in the presence of cellwise outliers. PCA methods that are also robust to cellwise outliers are scarce due to its late habitation in the literature. One such method is MacroPCA, which was introduced in Hubert et al. (2019) and also accounts for missing values and rowwise outliers. This method uses one-step M-estimators to identify and impute cellwise outliers.

In conclusion, PCA methods that are robust to rowwise outliers have received significantly more attention than PCA methods robust to cellwise outliers. Therefore, these methods are more optimised and come in many forms. Nonetheless, MacroPCA is an excellent PCA method that accounts for cellwise outliers, rowwise outliers and missing values.

2.2 Evaluation Criteria

As this paper follows the research of Hubert et al. (2019) for a large part, it makes sense to inherit their evaluation criterion. This is a variation of the mean squared error (MSE), the precise definition will be covered in the next chapter. This criterion is used in several papers where multiple regression models are compared such as Al-Nasser (2014) and Torabi et al. (2009). This criterion is also used in Heij et al. (2007), where PCR is compared with another model, which motivates the use of the MSE for this research.
There are several evaluation criteria that closely resemble the MSE. Some instances are discussed in Klimberg et al. (2010). Such as the root mean squared error (RMSE), which is the square root

of the MSE and the mean absolute error (MAE). Which is a more robust evaluation measure. They also mention the mean absolute percentage error (MAPE) developed in Lewis (1982). The MAPE can be used to evaluate the forecasting performance of a model by comparing estimated values with actual values.

To summarise, there are many evaluation criteria to be found in the literature. A certain group of these closely resemble the MSE. However, there are some small differences within this group, with some criteria being more robust for instance. The choice of which evaluation measure to use also comes down to preference.

2.3 Research Setup

In this section, the setup of our research will be explained.

In the previous sections, many different PCA methods and evaluation criteria were presented. However, as this paper is an extension of the paper Hubert et al. (2019), it is logical to follow their suggestions. Therefore, the following methods will be combined with a linear regression to create new PCR methods.
The first method is ICPCA from Nelson et al. (1996) and Kiers (1997). This method accounts for missing values but is not robust to outliers of any form. The second method is Serneels and Verdonck (2008) where the robust PCA method used is ROBPCA from Hubert et al. (2005). This method can deal with missing values and is robust to rowwise outliers. The final method is the MacroPCA method from Hubert et al. (2019), which is robust to rowwise and cellwise outliers and can handle missing values.
In Hubert et al. (2019), these methods are evaluated using a derivation of the MSE. Therefore, the MSE will also be used to compare the PCR models in this paper.

To perform our research the following is done. First, each PCA method is combined with the linear regression estimation such that we obtain three different PCR models. Then, a training dataset of clean data is simulated as a benchmark. Consequently, a test dataset is generated for model evaluation. Next, the data is contaminated in several ways. After that, the three models will be separately trained for each scenario and trained using the training dataset. Finally, forecasts are computed for the test dataset and the MSE is computed to compare the models.

This paper will provide valuable contributions to the existing literature for the following reasons. First, to our knowledge there does not yet exist a PCR method that accounts for missing

values as well as rowwise and cellwise outliers. Therfore, the MacroPCLR method introduced in this paper is brand new and a valuable addition to the literature. Secondly, this method will be compared to other PCR methods and will give insight into which PCR model to use for different contamination structures of data.

3 Methodology

In this section the methods used in this research are explained. The main method that of focus is called Principal Component Regression (PCR). PCR entails combining Principal Component Analysis (PCA) with a regression. Our implementation of PCR entails first performing a PCA method on the data and then using the acquired principal components to perform a linear regression (LR). To that end, the different PCA methods used are explained first after which the details of the LR are discussed.

3.1 Notation

Let X be an $n \times k$ data matrix where the rows represent individual observations and the columns represent various explanatory variables. An assumption is made that k is large. The element on the i-th row and the j-th column of X is denoted by x_{ij}. Let y be a vector of n observations of some dependent variable.

3.2 Principal Component Analysis

The goal of PCA when applied to a multivariate dataset is to drastically decrease the number of dimensions while retaining as much information or variability as possible according to Jolliffe and Cadima (2016). PCA accomplishes this by creating new variables called principal components (PCs). PCs are linear combinations of the original variables and are constructed such that they are uncorrelated. There are numerous PCA methods of which Classical PCA (CPCA) is the most simple, yet most impressionable method. This method is neatly described in Abdi and Williams (2010) and is shortly summarised here. After that, more advanced PCA methods are introduced.

3.2.1 Classical Principal Component Analysis

CPCA constructs the PCs by performing a singular value decomposition. This is a factorization on $\tilde{X}$, which is the centered version of X, thus $\tilde{X} = X - 1_n\mu'$. To be precise; $\tilde{X} = ADP^{\mathrm{T}}$. The PCs can now be extracted from the loading matrix P, based on a specified condition that determines the appropriate number of PCs, this number is denoted by τ. The scores T, which are the observed values of each PC, are then calculated by multiplying $\tilde{X}$ with the columns of the selected PCs in P. Hubert et al. (2019) summarises CPCA (or any PCA method) as estimating the model:

$$X = 1_n\mu' + \mathcal{T}(\mathcal{P})' + \mathcal{E}. \tag{3}$$

Where μ is the center, $\mathcal{T}$ is the score matrix, $\mathcal{P}$ is the loading matrix and $\mathcal{E}$ represents the residuals. Estimates for these vectors and matrices are respectively denoted by m, T and P. However, CPCA is very sensitive to outliers and cannot deal with missing values. Therefore this method is undesirable.

3.2.2 MacroPCA

The first PCA method that will be combined with a linear regression is MacroPCA. This method was introduced in Hubert et al. (2019) and is the first PCA method that accounts for missing values as well as rowwise and cellwise outliers. MacroPCA is performed in two different stages, the first stage is the Detecting Deviating Cells (DDC) stage and the second stage is the PCA stage. Therefore, the DDC stage will be explained first and after that the PCA stage will be discussed.

Detecting Deviating Cells: Computing Initial Imputations for Missing Values and Cellwise Outliers

The DDC stage consists of performing the DetectDeviatingCells algorithm. The goal of this algorithm is to make initial imputations for cellwise outliers and missing values. This algorithm was developed in Rousseeuw and Bossche (2018) and is the first algorithm of its nature to take the correlation between variables into account. The DDC algorithm can be broken down into eight steps. The entire DDC algorithm can be found in the appendix A.1. DDC outputs the sets $I_{c,\text{DDC}}$ and $I_{r,\text{DDC}}$ which respectively hold the indices of the cellwise outliers and the initial rowwise outlier. DDC also outputs the matrices $\mathring{X}$ and $\dot{X}$ which respectively represent the NA-imputed matrix and the NA-imputed and cell-imputed matrix.

The PCA Stage

The PCA stage of MacroPCA consists of six steps and uses $\mathring{X}$, $\dot{X}$, $I_{r,\text{DDC}}$ and $I_{c,\text{DDC}}$ from the DDC stage.

Step 1: Using Projection Pursuit To Construct a Robust Basis

The aim of this step is to get an initial assessment on which rows are the least outlying. More precisely, the algorithm will select $h < n$ rows to be the least outlying. h is conditioned by the constant α which is defined as follows: $0.5 \leq \alpha = h/n < 1$. In this paper $\alpha = 0.5$ is used.

Since the goal of this step is to get an initial assessment on outlying *rows*, outlying cells must be accounted for first. However, using the fully imputed matrix $\dot{X}$ might mask outlying rows as outlying cells in these rows have been imputed. In other words, using the cell imputed matrix makes rows that hold outlying cells seem less outlying than they really are. For these reasons,

a new matrix $\dot{X}^{(0)}$ is constructed from the original data matrix X in the following way. First, all missing values in X are imputed by the values of the corresponding cells in $\mathring{X}$. Then, the h rows of X that are not in $I_{r,\text{DDC}}$ and have the smallest number of cells in $I_{c,\text{DDC}}$ are selected. These selected rows are replaced by the corresponding rows of the fully imputed matrix $\dot{X}$. Thus $\dot{x}_i^{(0)} = \dot{x}_i$.

After $\dot{X}^{(0)}$ has been obtained, the outlyingness outl(.) of each row is computed. Where the outlyingness is equal to:

$$\text{outl}(\dot{x}_i^{(0)}) = \max_{v \in B} \frac{|v'\dot{x}_i^{(0)} - m_{\text{MCD}}(v'\dot{x}_j^{(0)})|}{s_{\text{MCD}}(v'\dot{x}_j^{(0)})}. \tag{4}$$

Here m_{MCD} and s_{MCD} are minimum covariance determinant (MCD) location and scale estimates. For more information on MCD estimators see Rousseeuw and Leroy (2005). The set B is a set of directions. Directions are the difference between two rows. The number of directions in B is equal to $\min(T, \min(250, T))$ where $T = (n(n-1))/2$ is the maximum number of directions possible.

Based on the outlyingness of each row, the h rows that are not in $I_{r,\text{DDC}}$ with the lowest outlyingness are put in a new set called H_0. This set represents the robust basis.

Step 2: Determining the Number of Principal Components

From this point onward, dimensions are provided below or after in parentheses of matrices and vectors when deemed appropriate. The goal of this step is to determine the number of principal components τ. To accomplish this, a new cell-imputed matrix $\dot{X}^{(1)}$ is constructed. It is constructed in the following way for rows $i = 1, ..., n$:

$$\dot{x}_i^{(1)} = \begin{cases} \dot{x}_i & \text{if } i \in H_0 \\ \mathring{x}_i & \text{if } i \notin H_0. \end{cases} \tag{5}$$

After that, the rows in H_0 are selected from the newly constructed matrix $\dot{X}^{(1)}$ and CPCA is applied on these rows exclusively. This results in the mean $m^{(1)}$ $(k \times 1)$, the loading matrix $P^{(1)}$ $(k \times k)$ and a diagonal matrix of the sorted eigenvalues $L^{(1)}$ $(k \times k)$. From these eigenvalues an appropriate τ can be chosen in multiple ways. In this paper, the default method is used unless stated differently, which chooses a τ such that 80% of the explained variance is retained. There is also an upper bound to τ called τ_{max}, which is set to 10 in this paper.

Step 3: Iteratively Applying CPCA on Non-Outlying Rows

The goal of this step is to estimate the τ PCs that fit our original data. To accomplish this, an iterative process takes place which is described below. Our starting input is $\dot{X}^{(1)}$ and the output of the CPCA performed in step 2.

convergence $\leftarrow$ FALSE.

$s \leftarrow 1$.

while $s < 20$ **and** convergence = FALSE **do**

$\underset{n\times\tau}{\dot{T}^{(s-1)}} = (\underset{n\times k}{\dot{X}^{(s-1)}} - 1_n(\underset{k\times 1}{m^{(s-1)}})')\underset{k\times\tau}{P^{(s-1)}}$.

$\hat{X}^{(s)} = 1_n(m^{(s-1)})' + \dot{T}^{(s-1)}(P^{(s-1)})'$.

for $i = 1$ to n **do**

for $j = 1$ to k **do**

$$\mathring{x}_{ij} = \begin{cases} \hat{x}_{ij}^{(s)} & \text{if } x_{ij} \text{ is NA} \\ \mathring{x}_{ij} & \text{otherwise.} \end{cases} \qquad \dot{x}_{ij}^{(s)} = \begin{cases} \hat{x}_{ij}^{(s)} & \text{if } ((i,j) \in I_{c,\text{DDC}} \text{ and } i \in H_0) \text{ or } x_{ij} \text{ is NA} \\ \dot{x}_{ij}^{(s-1)} & \text{otherwise.} \end{cases}$$

end for

end for

Apply CPCA to the rows of $\dot{X}^{(s)}$ for which holds that $i \in H_0$.

This results in the updated $m^{(s)}$ $(k \times 1)$ and $P^{(s)}$ $(k \times \tau)$.

Retrieve the smallest eigenvalue λ from the matrix $(P^{(s)})'P^{(s-1)}(P^{(s-1)})'P^{(s)}$.

if $\arccos(\sqrt{\lambda}) < tol$ **then**

convergence $\leftarrow$ TRUE.

end if

$s = s + 1$.

end while

Where $tol = 0.005$ is used in this paper. After this procedure the matrices $\mathring{X}$, $\dot{X}^{(s)}$, $m^{(s)}$ and $P^{(s)}$ are retrieved. Where $\mathring{X}$ is the NA-imputed matrix, $\dot{X}^{(s)}$ is the cell-imputed matrix, $m^{(s)}$ is the new estimated center and $P^{(s)}$ is the new loading matrix.

Step 4: Improving the Statistical Efficiency

The goal of this step is to improve the statistical efficiency for a low computational cost. See Rousseeuw and Leroy (2005) and Engelen et al. (2005) for more information about this topic. To accomplish this, the orthogonal distance to the current PCA subspace is calculated for each row $\dot{x}_i^{(s)}$ in $\dot{X}^{(s)}$. The orthogonal distance *od* is calculated in the following way:

$$od_i = ||\dot{x}_i^{(s)} - \hat{\dot{x}}_i^{(s)}|| = ||\dot{x}_i^{(s)} - (m^{(s)} + (\dot{x}_i^{(s)} - m^{(s)})P^{(s)}(P^{(s)})')||. \tag{6}$$

Where $||.||$ is defined as:

$$||p - q|| = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \tag{7}$$

for vectors p $(n \times 1)$ and q $(n \times 1)$. According to Hubert et al. (2005) the distribution of the orthogonal distances without the outliers to the power of 2/3 are roughly normal. Due to this

property, the cutoff value can be computed and new non-outlying rows can be selected. The cutoff value is computed as follows:

$$c_{od} := (m_{\text{MCD}}(od^*) + s_{\text{MCD}}(od^*)\Phi^{-1}(0.99))^{3/2}. \tag{8}$$

Where od^* is the vector of all $(od_i)^{2/3}$.

Each row for which holds that $od_i \leq c_{od}$ is marked as non-outlying with respect to the PCA subspace. The index of each non-outlying row for which also holds that this index is not in $I_{r,\text{DDC}}$ is stored in a new set called H^*. The number of rows in H^* is equal to n^*. The reweighted NA-imputed and cell-imputed matrix $\dot{X}$ is then computed in the following way:

$$\dot{x}_{ij} = \begin{cases} \hat{\dot{x}}_{ij}^{(s)} & \text{if } (i,j) \in I_{c,\text{DDC}} \text{ and } i \in H^* \\ \mathring{x}_{ij} & \text{otherwise.} \end{cases} \tag{9}$$

Finally, CPCA is applied to the n^* rows $\dot{x}_i$ for which holds that $i \in H^*$. This results in a new center m^* $(k \times 1)$ and a new loading matrix P^* $(k \times \tau)$.

Step 5: Computing a Robust Basis for the Final PCA Subspace

The goal of this step is to develop a robust basis of the final estimated subspace. To accomplish this a new scores matrix has to be computed first:

$$\underset{n^* \times \tau}{\dot{T}} = (\underset{n^* \times k}{\dot{X}} - 1_{n^*} \underset{k \times 1}{m^{*\prime}}) \underset{k \times \tau}{P^*}. \tag{10}$$

After this, the DetMCD method developed in Hubert et al. (2012) is applied to $\dot{T}$. The goal of applying this method is to obtain robust location and scatter estimators in a multivariate setting. In this setting this will result in the robust location estimate m_{MCD} $(\tau \times 1)$ and the robust scatter estimate S_{MCD} $(\tau \times \tau)$. Next, the spectral decomposition of S_{MCD} is performed to retrieve a new loading matrix P_{MCD} $(\tau \times \tau)$. Finally, the final center $m = m^* + P^* m_{\text{MCD}}$ and the final loading matrix $P = P^* P_{\text{MCD}}$ are computed.

Step 6: Calculating the Final Scores and Predicted Values

The goal of this step is to calculate the final output. To do this, new score matrices are computed first:

$$\mathring{T} = (\mathring{X} - 1_n m')P \text{ and } \dot{T} = (\dot{X} - 1_n m')P. \tag{11}$$

These score matrices are then used to compute the final predictions for X:

$$\hat{\mathring{X}} = 1_n m + \mathring{T}(P)' \text{ and } \hat{\dot{X}} = 1_n m + \dot{T}(P)'. \tag{12}$$

This concludes the MacroPCA method. In the Linear Regression section $\dot{T}$ will be used as the score matrix T_{M} that the MacroPCA method outputs.

3.2.3 ICPCA and MROBPCA

The ICPCA and MROBPCA methods will be shortly described in this section.

ICPCA

Iterative CPCA (ICPCA) is a PCA method developed by Nelson et al. (1996) and Kiers (1997) that can deal with missing values. The method initially imputes missing values by the column means of non-missing values. After that, an iterative procedure is initiated until a certain convergence condition is met or after a certain amount of iterations. The procedure starts by applying CPCA. Then using the obtained estimates for the center, scores and loadings, an estimate for X is calculated called $\hat{X}$. The cells of X that originally contained missing values are replaced by the corresponding values in $\hat{X}$ and the other cells remain unaffected. Finally the termination conditions are checked and the procedure is repeated until these are satisfied.

MROBPCA

MROBPCA combines the methods of Serneels and Verdonck (2008) and Hubert et al. (2005) and is a PCA method that can also deal with missing values. Additionally, this method is robust to rowwise outliers. It works similar to ICPCA with two key differences. The first difference is that the initial imputations of missing values are not equal to the column mean. Instead, they are imputed using robust location and scale estimates. Secondly, MROBPCA first calculates a certain number of least outlying rows and then applies CPCA on only these rows.

Each of the aforementioned PCA methods result in separate scores, loadings and centers. For the MacroPCA method these matrices will be denoted by T_{M}, P_{M} and m_{M}. For the ICPCA method these matrices will be denoted by T_{I}, P_{I} and m_{I}. For the MROBPCA method these matrices will be denoted by T_{R}, P_{R} and m_{R}.

3.3 Linear Regression

In this section the LR stage of PCR will be discussed. We first consider an arbitrary PCA method where T $(n \times \tau)$ denotes the score matrix, P $(k \times \tau)$ the loading matrix and m $(k \times 1)$ the center. Later, the model will be specified for each method.

The goal of the LR stage is to construct and estimate a LR model using the output of the PCA stage. To this end, the retrieved score matrix will represent the matrix of explanatory variables. This results in the following model:

$$y = T\beta + \epsilon. \tag{13}$$

The estimator b of β, is then computed by performing the LR. The LR method used is the QR method. This method is further explained in Ansley (1985), but is not relevant for this research. Also note that an intercept is not included in the regression. This choice was made after a review of existing PCR methods, of which the majority did not include an intercept. See for instance, Agarwal et al. (2021), Faber and Kowalski (1997) and Xie and Kalivas (1997). This means that the PCR models that will be compared in the results section are the following. ICPCLR, which combines ICPCA with the LR model. MROBPCLR, which combines MROBPCA with the LR model. MacroPCLR, which combines MacroPCA with the LR model.

3.4 Simulation

To compare these PCR methods, two simulated datasets are created by combining the simulation methods of Artigue and Smith (2019) and Hubert et al. (2019). These datasets represent the training dataset and the testing dataset. The training dataset is used to train the PCR models and the testing dataset is used to evaluate them. This is a standard practice in econometrics that is applied to properly evaluate the performance of models without bias. To this end, we first describe how the clean datasets are generated. Next, we explain how these clean datasets are contaminated.

3.4.1 Clean Data

To simulate the entire training dataset, the clean data matrix X^0_{Train} has to be created first. X^0_{Train} will then be used to construct y_{Train}. X^0_{Train} is an $(n \times k)$ matrix with $n = 100$ and $k = 200$ that is generated by a multivariate normal distribution (N) with mean $\mu = 0$ and covariance matrix Σ. Σ is created from a correlation matrix called A09. Each element a_{ij} in A09 is computed by $a_{ij} = (-0.9)^{|i-j|}$. The spectral decomposition of A09 is then performed yielding the matrices E $(k \times k)$ which contains the ordered eigenvalues of A09 and P $(k \times k)$ which contains the eigenvectors corresponding to the eigenvalues in E. The diagonal matrix $L = diag(30, 25, 20, 15, 10, 5, v_6, ..., v_k)$ is then constructed where $v_i = 0.098 - 0.0005(i - 6)$ for $i = 6, ..., k$. The number of PCs in each method is manually specified to be equal to $\tau = 6$. Finally, the covariance matrix is computed by $\Sigma = PLP'$.

The generated matrix X^0_{Train} can now be used to construct y_{Train}. To this end, five random columns of X^0_{Train} are selected denoted by x_j, five coefficients $c_j \sim \text{UNIFORM}(2, 4)$ are generated and a noise vector $\underset{n\times 1}{\varepsilon} \sim N(0, 1)$ is generated. If for instance, columns $5, 17, 69, 134$ and 189 are selected the generated coefficients will be denoted by c_5, c_{17}, c_{69}, c_{134} and c_{189}. y_{Train} is then computed in the following way:

$$y_i = \sum_{j=1}^{5} c_j x_{ij} + \varepsilon_i. \tag{14}$$

X^0_{Test} and y_{Test} are generated in the same way, except that for X^0_{Test} 50 rows denoted by n' are generated instead of 100.

3.4.2 Contamination

To asses the performance of ICPCLR, MROBPCLR and MacroPCLR with the presence of missing values as well as rowwise and cellwise outliers, the clean data, so both the training and the test data, are contaminated in four different ways, resulting in four different scenarios. These scenarios are directly taken from Hubert et al. (2019). Note that the training data and test data are contaminated separately.

Scenario 1: $n * k * p_t$ cells are randomly selected and imputed by NAs.
With $p = (0.05, 0.1, 0.15, 0.2, 0.25, 0.3)$. The resulting data is denoted by X^1_{Train}.
Scenario 2: First $n * k * 0.2$ cells are randomly selected and imputed by missing values. Of the remaining $n * k * 0.8$ cells, $n * k * 0.2$ cells are randomly selected and imputed with cellwise outliers. If x_{ij} is selected to be an outlier, it is imputed by $\gamma_t \sigma_j$ where σ_j^2 is the j-th diagonal element of Σ and $\gamma = (0, 1, ..., 20)$. The resulting data is denoted by X^2_{Train}.
Scenario 3: First $n * 0.2$ rows are replaced by rows generated from $N(\gamma_t e_{\tau+1}, \Sigma)$ with $\gamma = (0, 1, ..., 50)$ and where $e_{\tau+1}$ represents the $(\tau + 1)$-th column of P. After that, $n * d * 0.2$ random cells are imputed with NAs. The resulting data is denoted by X^3_{Train}.
Scenario 4: First $n * k * 0.2$ cells are randomly selected and stored in the set C_{NA}. Then of the remaining $n * k * 0.8$ cells, $n * k * 0.1$ cells are randomly selected and stored in the set C_{out}. Next, $n * 0.1$ rows are randomly selected and stored in the set R_{out}. Then in the following order the data is contaminated. First the cells in C_{out} are imputed in the same fashion the cellwise outliers in scenario 2 are imputed. Next, the rows in R_{out} are imputed in the same way as in scenario 3. Finally, the cells in C_{NA} are imputed by missing values. The resulting data is denoted by X^4_{Train}.
The above scenarios specify how the training data is contaminated. To contaminate the test data, simply replace n by n' and apply the procedures.

It is important to note that in this research the choice was made to compute both y_{Train} and y_{Test} before any contamination takes place. The reason for this choice is the way that each PCA method handles outliers. Namely, they are replaced for non-outlying values. Introducing contamination to our dependent variable would thus lead to contradictory results. However, this also means that the proposed methods cannot appropriately deal with outlying dependent variables.

3.5 Evaluation

The generated explanatory data, both contaminated and uncontamined, as well as the generated dependent variables are then used to evaluate ICPCLR, MROBPCLR and MacroPCLR in several ways.

First, the accuracy of the different PCA methods is measured as is done in Hubert et al. (2019). To accomplish this, a new matrix is created for each scenario called X^R_{Train} $(r \times k)$ which is equal to the matrix X_{Train}, but with rows selected to be contaminated removed, thus rows in R_{out}. Next, CPCA is applied on X^R_{Train} and the resulting output is used to compute predictions for X^R_{Train} denoted by $\hat{x}^R_{ij}$. After that, the PCA methods ICPCA, MROBPCA and MacroPCA are applied to X^s_{Train} with $s = 1, 2, 3, 4$ and the resulting predictions are denoted by $\hat{x}_{ij}$. Finally, the mean squared error (MSE) is calculated to compare the methods:

$$\text{MSE}_{\text{PCA}} = \frac{1}{rk} \sum_{i \in R_{\text{out}}} \sum_{j=1}^{k} (\hat{x}_{ij} - \hat{x}^R_{ij})^2. \tag{15}$$

This procedure was repeated 100 times and the MSE was averaged over these repetitions. For this part, the data was also simulated and evaluated using the ALYZ correlation matrix developed in Agostinelli et al. (2015).

Next, the performance of the full PCR methods was assessed. To accomplish this for each scenario, the following LR models were estimated first:

$$y_{\text{Train}} = T^s_{\text{I}}\beta + \epsilon, \quad y_{\text{Train}} = T^s_{\text{R}}\beta + \epsilon \quad \text{and } y_{\text{Train}} = T^s_{\text{M}}\beta + \epsilon. \tag{16}$$

Where T^s_{I} represents the score matrix retrieved from applying the PCA method specified in the subscript (See the end of section 3.4) on X^s_{Train} and where $s = 1, 2, 3, 4$ indicates the scenario number. These regressions will result in the estimators b^s_{I} for the first model, b^s_{R} for the second model and b^s_{M} for the third model. These estimators will then be used in cooperation with X^s_{Test} to make predictions $\hat{y}$ for y_{Test}.

To compute $\hat{y}$, the scores matrix is needed first. If there were no missing values in each X^s_{Test}, one could simply project this matrix onto the existing PCA subspaces to retrieve the predictions. However, missing values are present in each X^s_{Test} and therefore, these have to be imputed first. For ICPCLR missing values in X^s_{Test} are imputed by the column mean. For MROBPCLR missing values are imputed by equation (23). For MacroPCLR the MacroPCApredict function from Hubert et al. (2019) is used to immediately project each X^s_{Test} onto the existing PCA subspace computed for each X^s_{Train}, this results in the scores matrices $\hat{T}^s_{\text{M}}$.

For ICPCLR and MROBPCLR the scores matrices are computed in the following way after

imputation:

$$\hat{T}_{\mathrm{I}}^{s} = (X_{\mathrm{Test}}^{s} - 1_{n'} m_{\mathrm{I}}')P_{\mathrm{I}} \quad \text{and} \quad \hat{T}_{\mathrm{R}}^{s} = (X_{\mathrm{Test}}^{s} - 1_{n'} m_{\mathrm{R}}')P_{\mathrm{R}}. \tag{17}$$

The obtained score matrices are then used to compute $\hat{y}$ for each scenario and method:

$$\hat{y}_{\mathrm{F}}^{s} = \hat{T}_{\mathrm{F}}^{s} b_{\mathrm{F}}^{s} \text{ for } F = \mathrm{I},\ \mathrm{R},\ \mathrm{M} \text{ and } s = 1, 2, 3, 4. \tag{18}$$

The predictions are then evaluated by the MSE:

$$\mathrm{MSE}_{\mathrm{PCR}} = \frac{1}{n'} \sum_{i=1}^{n'} (y_i^{\mathrm{Test}} - \hat{y}_i)^2. \tag{19}$$

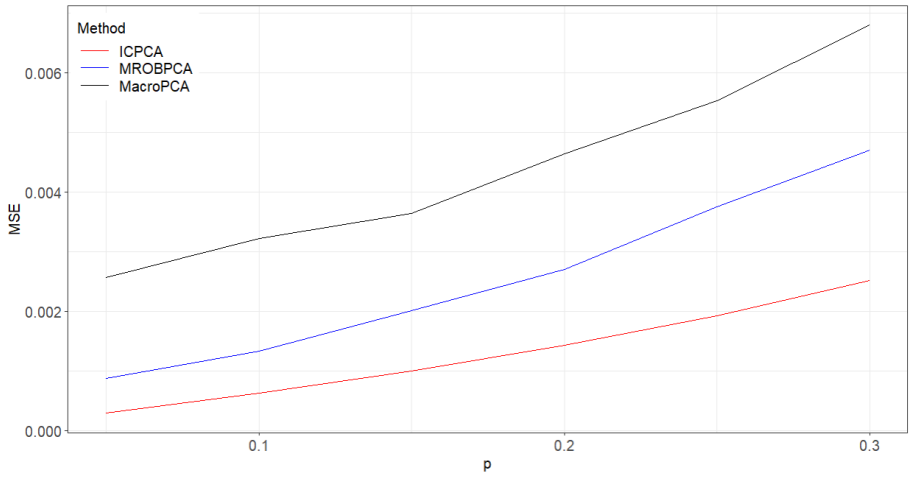
This procedure was repeated 50 times and the MSE was averaged over these repetitions.
In the next section the results of our research will be presented.

4 Results

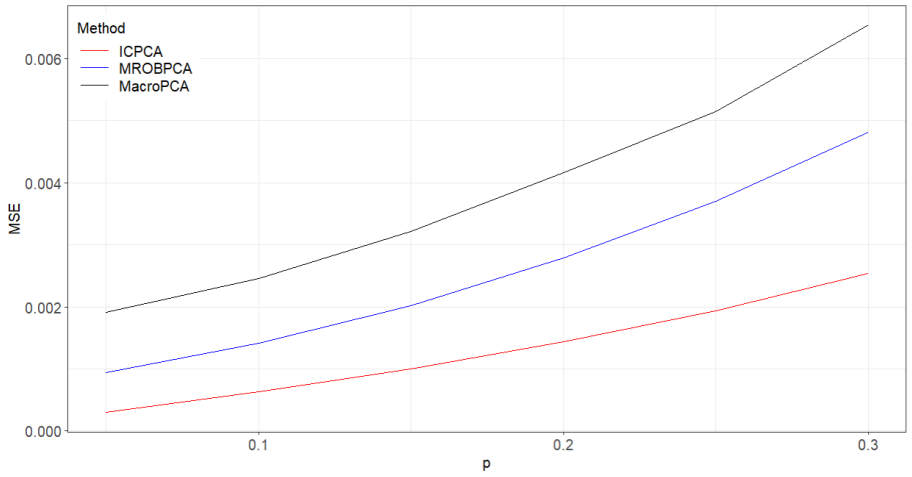
In the previous section, the methods that were implemented to conduct our research were explained. Additionally, we explained how the data was generated and which evaluation criteria were used. In this section, the results of our research will be presented. First, the comparison of the PCA methods will be presented. After that, the comparison of the PCR methods will be presented. To acquire these results the programming language R (Version 4.1.2) and the compiler Rstudio were used. A description of the code that was developed can be found in the appendix A.3. Note that the code implements functions from the R packages *cellWise* and *rrcov*.

4.1 PCA Comparison

The comparison of the three PCA methods tries to replicate the simulation study conducted in Hubert et al. (2019). While most functions and methods used in Hubert et al. (2019) were convenient to implement, such as MacroPCA, other methods were unclear and made their results harder to reproduce. For instance, the exact details of their implementation of the MROBPCA method were missing which forces the reader to use their own MROBPCA method, as is done in this research. Therefore, the results for the MROBPCA method will differ throughout all contamination scenarios. Which leads to another point of critique. The setup of the simulation in Hubert et al. (2019) is not completely evident. The order of contamination for some of the scenarios was not completely clear either, for instance it was unclear if imputed rowwise outliers could contain missing values. This also lead to some deviating results. Another specification left unspecified were the exact values of the parameters of ICPCA, MROBPCA, DDC and MacroPCA. The results can be found below and will now be discussed for each scenario.
The first scenario, see Figure 1, only contaminated the data with missing values.



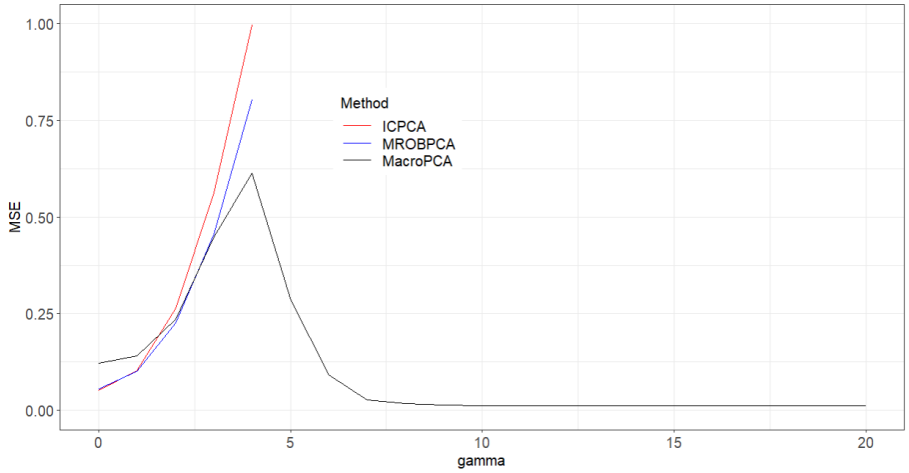
(a) A09



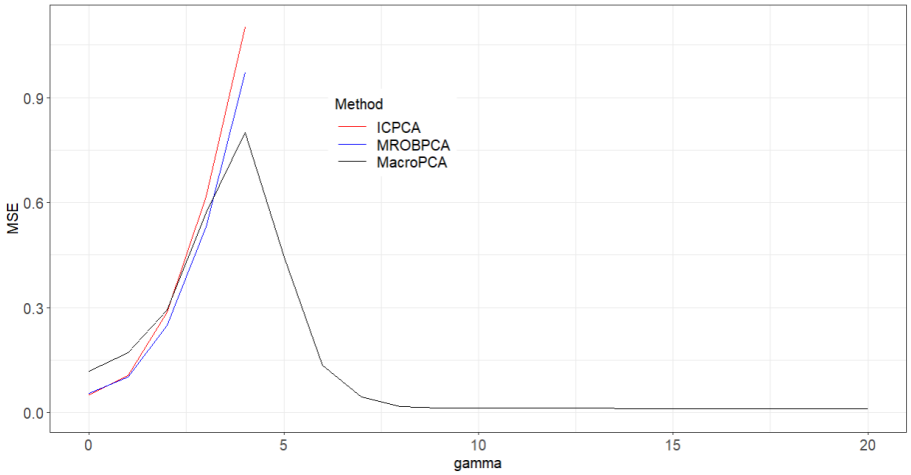
(b) ALYZ

Figure 1: PCA Comparison: Scenario 1

This is congruent with the fact that ICPCA has the lowest MSE for this scenario. There are no outliers that cause this non-robust method to break down. The second scenario, see Figure 2 introduced cellwise outliers in addition to the missing values.



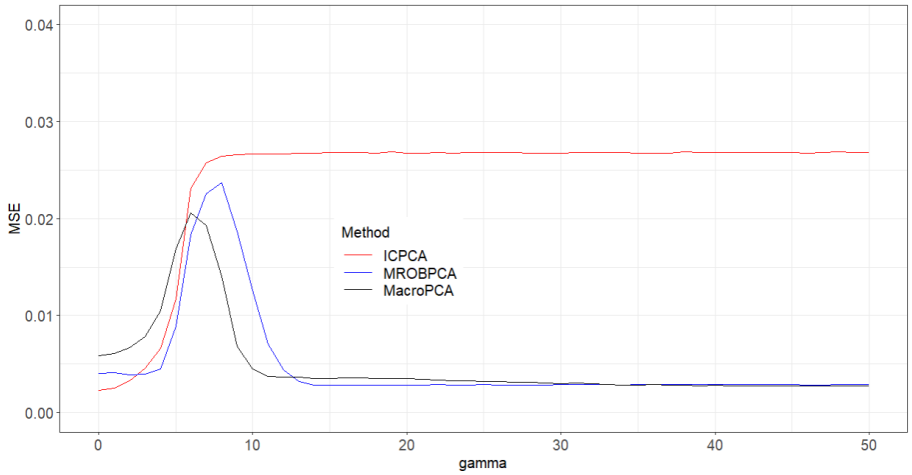
(a) A09



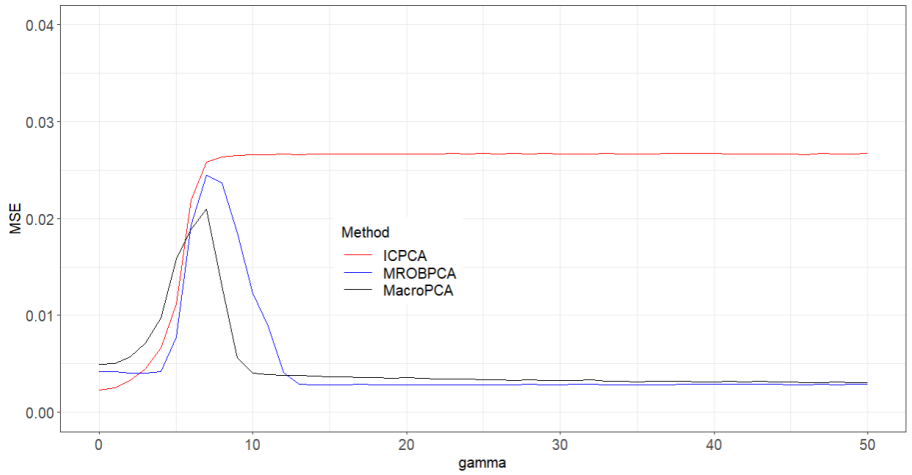
(b) ALYZ

Figure 2: PCA Comparison: Scenario 2

It is evident that both the ICPCA and MROBPCA methods diverge quickly, due to their inability to handle cellwise outliers. The third scenario, see Figure 3, combined missing values with rowwise outliers.



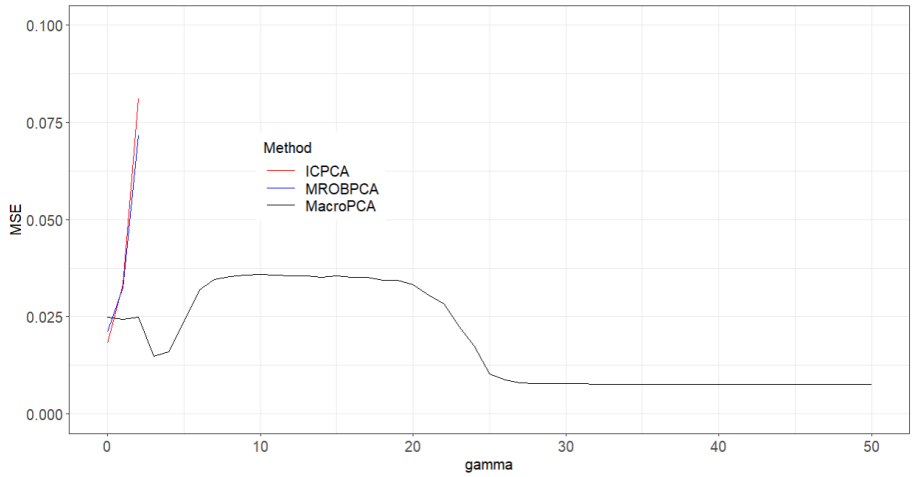
(a) A09



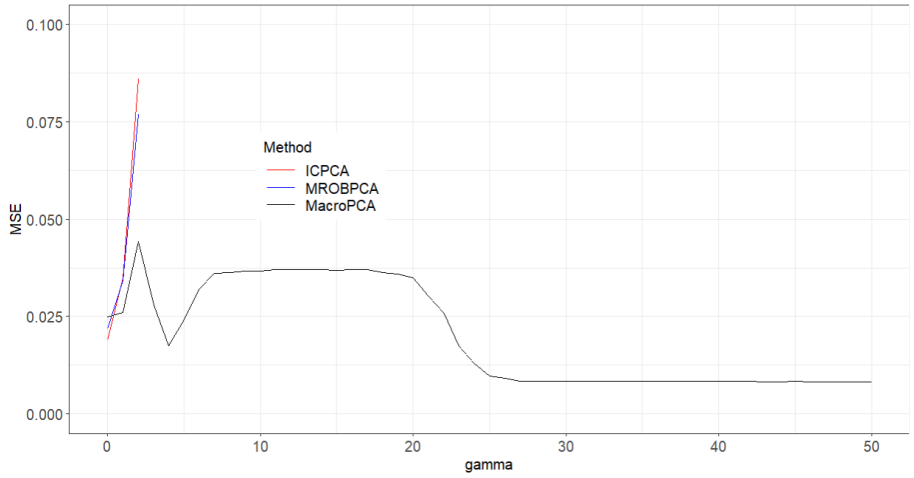
(b) ALYZ

Figure 3: PCA Comparison: Scenario 3

An interesting observation is the fact that the MSE of the ICPCA method seems to converge to a certain upperbound in our implementation. This was not the case in Hubert et al. (2019). There could be several reasons for this including a different contamination order and a wrong implementation of the ICPCA method. Determining the exact reason lies beyond the scope of this research. Despite that, the acquired results are logical. ICPCA has a significantly larger MSE than the other methods as it is the only method that is not robust to rowwise outliers. The results of the last scenario, see Figure 4 are again quite similar to the results of Hubert et al. (2019), despite some minor differences.



(a) A09



(b) ALYZ

Figure 4: PCA Comparison: Scenario 4

The MSE of both ICPCA and MROBPCA grow very fast and the MSE of MacroPCA shrinks to zero as the outliers become large enough to be picked up by the algorithm.

To conclude, if there are no outliers in the data one should use ICPCA. A combination of rowwise outliers and missing values can be treated with MROBPCA or MacroPCA. Finally, in the presence of cellwise outliers one should opt for MacroPCA.

4.2 PCR Comparison

In this section the PCR models ICPCLR, MROBPCLR and MacroPCLR will be compared. The results of each scenario can be found below.

The first thing to note is that in each scenario, MacroPCLR has the lowest MSE. This is in contrast to results of Hubert et al. (2019), in which ICPCA had the lowest MSE for the first scenario, though by a small margin. This systematic superiority of the MacroPCLR method is

most likely a result of the usage of MacroPCApredict. This function imputes missing values and outliers in the test data over iterations, instead of once which is the case for ICPCLR and MROBPCLR. From this can be concluded that a PCR method where missing values and outliers in the test data are imputed in an iterative manner is preferable to a PCR method which imputes these values just once. However, this also means that the results of the PCR methods compared to each other might be biased.

The results of the first scenario can be seen in Figure 5.

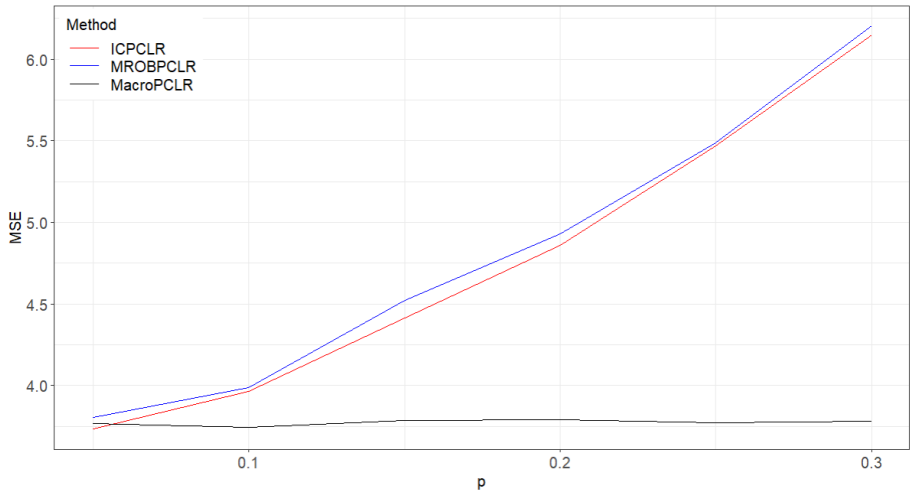


Figure 5: PCR Comparison: Scenario 1

Interesting is the large growth of the MSE of ICPCLR and MROBPCLR, while MacroPCLR grows much slower. However, this difference seems logical since the NAs in ICPCLR and MROBPCLR are imputed once by a shrinking sample. This leads to increasingly larger biases of the imputed values. The slow increasing MSE of MacroPCLR is in line with the results of Hubert et al. (2019) for the first scenario. This slow growth can be seen in Figure 9 in the appendix.

The results of the second scenario can be seen in Figure 6 and are quite logical as MacroPCLR is the only method equipped to deal with cellwise outliers.

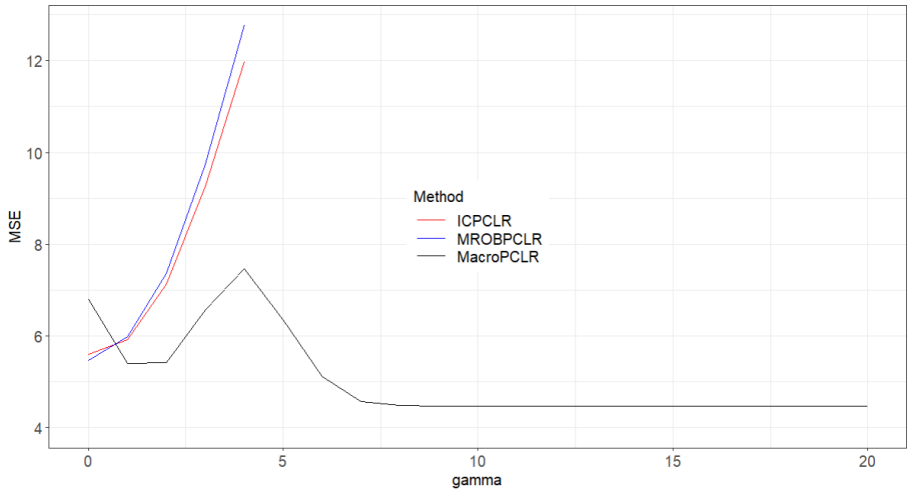


Figure 6: PCR Comparison: Scenario 2

The shape of the graph also resembles the results of the PCA comparison quite well. Interesting is the case with $\gamma = 0$, for this case MacroPCLR has a higher MSE. This is logical as the clean data has mean 0 and thus the imputed outliers will not represent actual outliers.
The results of the third scenario can be seen in Figure 7 and display some interesting behaviour.

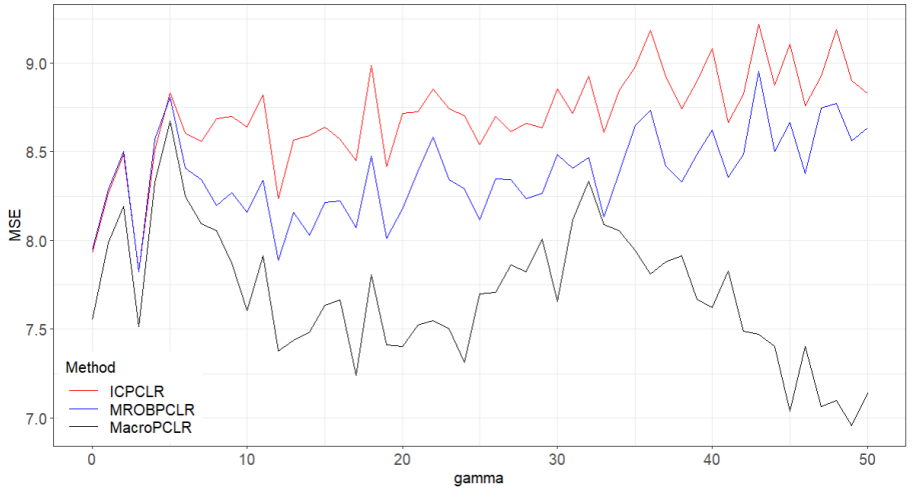


Figure 7: PCR Comparison: Scenario 3

In contrast to the other scenarios, the MSE of ICPCLR and MROBPCLR do not rapidly grow as γ increases. Additionally, the graphs are much less smooth than those of the other scenarios. The reason for this is probably correlated with the fact that the estimated model is also contaminated with rowwise outliers. However, to get a grasp of the exact reason an extensive analysis is needed, which is beyond the scope of this research.
The results of the fourth scenario can be seen in Figure 8 and are logical again.

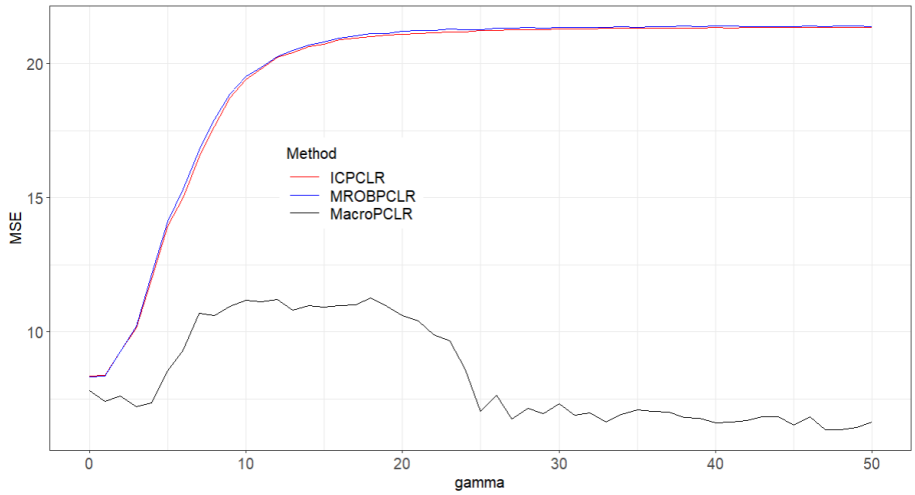


Figure 8: PCR Comparison: Scenario 4

They closely resemble the graph of the PCA comparison of the fourth scenario except for the seemingly upperbounded ICPCLR and MROBPCLR.

In the next section, the findings of this research will be summarised and discussed.

5 Conclusion

The main research question of this research is: *How does the forecasting accuracy of MacroPCLR perform compared to other PCR methods in the presence of outliers and missing values?*. To answer this question, three new PCR methods were developed, ICPCLR, MROBPCLR and MacroPCLR. These methods combine a LR with ICPCA, MROBPCA and MacroPCA respectively. Of these PCA methods, MacroPCA is the most refined as it can deal with missing values and is robust to rowwise as well as cellwise outliers. These PCA methods and the corresponding PCR methods were compared in four different settings where missing values and outliers were present. They were compared by first generating a clean training dataset of 100 observations with 200 explanatory variables and one dependent variable y, also with 100 observations. The PCA methods were compared by calculating a type of MSE for predictions of X. The results of this comparison closely resembled the simulation study conducted in Hubert et al. (2019). They showed that in a dataset with no outliers one should use ICPCA and MacroPCA otherwise. Predictions for y were then made using the three PCR models, the trainig dataset and the similarly generated test dataset. These predictions were evaluated by calculating the MSE. This showed that using the MacroPCLR method should be used no matter the contamination structure. However, this result is likely biased due to the fact that the MacroPCLR method iteratively imputes missing values and cellwise outliers in the test dataset where as ICPCLR and MROBPCLR impute missing values just once. Nonetheless, the forecasting accuracy of PCR in combination with MacroPCLR appears to perform better than other PCA methods combined

with PCR.

This implies that anyone who has a high-dimensional dataset and wants to use a LR model for forecasting; using the MacroPCLR method is an option worth considering. In further research one could consider implementing ICPCA and MROBPCA versions of MacroPCApredict and then comparing the PCR models. Additionally, one could compare these methods for a real dataset. Finally, one could consider dropping insignificant PCs in the LR stage of the PCR methods.

During the course of our research, some underlying assumptions were made. It is important to consider these assumptions when interpreting the results. The first assumption is that cells are missing completely at random (MCAR). The next assumptions are the assumptions needed for PCA, linearity, large variances have important structures and PCs are orthogonal. For more details see Shlens (2014). Finally, to perform a linear regression Heij et al. (2004) lists seven assumptions that must hold. Consequent of the design of the simulation, all seven assumptions hold. However, this is not inherent for most datasets. In further research the violation of these assumptions in light of the forecasting performance of MacroPCLR could be investigated.

References

H. Abdi. Partial least squares regression and projection on latent structure regression (pls regression). *Wiley interdisciplinary reviews: computational statistics*, 2(1):97–106, 2010.

H. Abdi and L. J. Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

A. C. Acock. Working with missing values. *Journal of Marriage and family*, 67(4):1012–1028, 2005.

A. Agarwal, D. Shah, D. Shen, and D. Song. On robustness of principal component regression. *Journal of the American Statistical Association*, 116(536):1731 – 1745, 2021. ISSN 01621459.

C. Agostinelli, A. Leung, V. J. Yohai, and R. H. Zamar. Robust estimation of multivariate location and scatter in the presence of cellwise and casewise contamination. *Test*, 24(3):441–461, 2015.

A. D. Al-Nasser. Two steps generalized maximum entropy estimation procedure for fitting linear regression when both covariates are subject to error. *Journal of Applied Statistics*, 41(8): 1708–1720, 2014.

F. Alqallaf, S. Van Aelst, V. J. Yohai, and R. H. Zamar. Propagation of outliers in multivariate data. *The Annals of Statistics*, pages 311–331, 2009.

C. F. Ansley. Quick proofs of some regression theorems via the qr algorithm. *The American Statistician*, 39(1):55–59, 1985.

H. Artigue and G. Smith. The principal problem with principal components regression. *Cogent Mathematics & Statistics*, 6(1):1622190, 2019.

C. Croux and G. Haesbroeck. Principal component analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies. *Biometrika*, 87(3):603–618, 2000.

S. Engelen, M. Hubert, and K. V. Branden. A comparison of three procedures for robust pca in high dimensions. *Austrian Journal of Statistics*, 34(2):117–126, 2005.

K. Faber and B. R. Kowalski. Propagation of measurement errors for the validation of predictions obtained by principal component regression and partial least squares. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(3):181–238, 1997.

F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1): 1–21, 1969.

B. Grung and R. Manne. Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 42(1-2):125–139, 1998.

C. Heij, P. de Boer, P. H. Franses, T. Koek, and H. K. van Dijk. *Econometric Methods with Applications in Business and Economics*. Oxford University Press, 2004.

C. Heij, P. J. Groenen, and D. van Dijk. Forecast comparison of principal component regression and principal covariate regression. *Computational statistics & data analysis*, 51(7):3612–3625, 2007.

M. Hubert and S. Verboven. A robust pcr method for high-dimensional regressors. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(8-9):438–452, 2003.

M. Hubert, P. J. Rousseeuw, and K. Vanden Branden. Robpca: a new approach to robust principal component analysis. *Technometrics*, 47(1):64–79, 2005.

M. Hubert, P. J. Rousseeuw, and T. Verdonck. A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics*, 21(3):618–637, 2012.

M. Hubert, P. J. Rousseeuw, and W. Van den Bossche. Macropca: An all-in-one pca method allowing for missing values as well as cellwise and rowwise outliers. *Technometrics*, 61(4): 459–473, 2019.

I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

H. A. Kiers. Weighted least squares fitting using ordinary least squares algorithms. *Psychometrika*, 62(2):251–266, 1997.

R. K. Klimberg, G. P. Sillup, K. J. Boyle, and V. Tavva. Forecasting performance measures–what are their practical meaning? In *Advances in business and management forecasting*. Emerald Group Publishing Limited, 2010.

C. Lewis. A radical guide to exponential smoothing and curve fitting. *Industrial and Business Forecasting Methods*, 1982.

G. Li and Z. Chen. Projection-pursuit approach to robust dispersion matrices and principal components: primary theory and monte carlo. *Journal of the American Statistical Association*, 80(391):759–766, 1985.

R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibián-Barrera. *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019a.

R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibián-Barrera. *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019b.

P. R. Nelson, P. A. Taylor, and J. F. MacGregor. Missing data methods in pca and pls: Score calculations with incomplete observations. *Chemometrics and intelligent laboratory systems*, 35(1):45–65, 1996.

S. Oba, M.-a. Sato, I. Takemasa, M. Monden, K.-i. Matsubara, and S. Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

L. V. Perez. Principal component analysis to address multicollinearity. *Whitman College: Walla Walla, WA, USA*, 2017.

U. Ramanathan. Supply chain collaboration for improved forecast accuracy of promotional sales. *International Journal of Operations & Production Management*, 2012.

P. J. Rousseeuw and W. V. D. Bossche. Detecting deviating data cells. *Technometrics*, 60(2): 135–145, 2018.

P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005.

S. Serneels and T. Verdonck. Principal component analysis for data containing outliers and missing elements. *Computational Statistics & Data Analysis*, 52(3):1712–1727, 2008.

K. A. Severson, M. C. Molaro, and R. D. Braatz. Principal component analysis of process datasets with missing values. *Processes*, 5(3):38, 2017.

J. Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

M. Torabi, G. S. Datta, and J. Rao. Empirical bayes estimation of small area means under a nested error linear regression model with measurement errors in the covariates. *Scandinavian Journal of Statistics*, 36(2):355–369, 2009.

Y.-L. Xie and J. H. Kalivas. Evaluation of principal component selection methods to form a global prediction model by principal component regression. *Analytica chimica acta*, 348(1-3): 19–27, 1997.

A Appendix

A.1 DDC

The DDC stage consists of performing the DetectDeviatingCells algorithm. The goal of this algorithm is to make initial imputations for cellwise outliers and missing values. This algorithm was developed in Rousseeuw and Bossche (2018) and is the first algorithm of its nature to take the correlation between variables into account. The DDC algorithm can be broken down into eight steps.

Step 1: Standardization

The goal of this step is to standardize each column j of X, denoted by x_j, by using robust estimators of location and scale.

To calculate these estimators, Tukey's biweight function from Maronna et al. (2019b) is needed. The formula of this function can be seen below:

$$W(t) = \left(1 - \left(\frac{t}{c}\right)^2\right)^2 I(|t| \leq c). \tag{20}$$

Here $c > 0$ is a constant. Its standard value is $c = 3$, which is the value used in this research. To now calculate the estimators set:

$$m_{j0} = med_{i=1}^{n}(x_{ij}), \quad s_{j0} = med_{i=1}^{n}|x_{ij} - m_{j0}| \quad \text{and} \quad s_{j1} = med_{i=1}^{n}(|x_{ij}|). \tag{21}$$

Where *med* stands for the median. Then set:

$$w_{ij} = W\left(\frac{(x_{ij} - m_{j0})}{s_{j0}}\right). \tag{22}$$

The estimate of the robust location of column x_j is then computed as follows:

$$m_j = robLoc(x_j) = \left(\sum_{i=1}^{n} w_{ij}x_{ij}\right) \Big/ \left(\sum_{i=1}^{n} w_{ij}\right). \tag{23}$$

The estimate of the robust scale of x_j is equal to:

$$s_j = robScale(x_j) = s_{j1}\sqrt{\frac{1}{\delta n}\sum_{i=1}^{n}\gamma\left(\frac{x_{ij}}{s_{j1}}\right)}. \tag{24}$$

Where $\delta = 0.845$ and $\gamma(t) = min(t^2, a^2)$ with $a = 2.5$.

These formulas are used to standardize the columns of X. The standardized version of X will

be denoted by Z and is calculated as $z_{ij} = (x_{ij} - m_j)/s_j$.

Step 2: Univariate Outlier Detection

The goal of this step is to make an initial assessment on which cells are outlying.

To accomplish this, a new matrix U is created. This matrix is identical to Z, but in this matrix outlying cells will be replaced by missing values. It is constructed as follows:

$$u_{ij} = \begin{cases} z_{ij} & \text{if } |z_{ij}| \leq c_u \\ \text{NA} & \text{if } |z_{ij}| > c_u. \end{cases} \tag{25}$$

Where $c_u = \sqrt{\chi^2_{1,p}}$ with $p = 0.99$. c_u is called the cutoff value as it is the cutoff of what is considered as an outlier.

Step 3: Bivariate Relations

The goal of this step is to find correlated variables to increase the accuracy of the imputations. To find these variables, a robust correlation measure is calculated for each combination of two different columns in U. As a demonstrative example, consider the columns l and h with $l \neq h$. For columns u_l and u_h, all rows where at least one NA value is present, are dropped. This results in columns u_l^* and u_h^* which have no missing values and the same number of elements as each other. To now estimate the robust correlation measure set:

$$\hat{\rho}_{lh} = ((robScale(u_l^* + u_h^*))^2 - (robScale(u_l^* - u_h^*)^2)/4. \tag{26}$$

$\hat{\rho}_{lh}$ is then used to construct a tolerance ellipse around $(0, 0)$ for u_l^* and u_h^*. The coverage probability of this ellipse is the same as the p used to calculate the cutoff value in (8). $robCorr$ is then defined as the standard correlation measure of all data points of the columns u_l^* and u_h^* that are inside this ellipse. Thus, $cor_{lh} = robCorr(u_l^*, u_h^*)$.

Columns j for which holds that $|cor_{jl}| \geq 0.5$ for some column $l \neq j$ are labelled as connected variables. Pairs of columns that satisfy this condition are called connected pairs. Variables that are not connected are labelled as standalone variables.

As connected pairs are sufficiently correlated, they can be used to predict each other. This is why for these pairs another measure called $robSlope$ is calculated. To calculate this measure assume that columns l and h are a connected pair and set:

$$b_{lh} = med_{i=1}^{m} \left(\frac{u_{il}^*}{u_{ih}^*} \right). \tag{27}$$

If there is an i for which $u_{ih}^* = 0$, then this i is taken out of the equation. This means that m is equal to the number of elements in u_h^* that are not equal to 0. Next, compute $r_{ilh} = u_{il}^* - b_{lh} u_{ih}^*$ for $i = 1, ..., m$. Then, select the observations for which holds that: $|r_{ilh}| \leq c_u robScale(r_{lh})$ and put these i in the set G. Where c_u is the cutoff value from Step 2 and r_{lh} is the vector of all

calculated r_{ilh}. Finally, $\beta_{lh} = robSlope(u_l^*|u_h^*)$ is defined as the slope of the regression line of the no-intercept least-squares regression on all observations i in the set G.

Step 4: Predicted Values

The goal of this step is to make a new matrix $\hat{Z}$ which contains predictions for the outliers and missing values in Z. The calculation of these predictions differ for standalone variables and connected variables. For standalone variables, predictions for outlying z_{ij} are set to 0 and are thus predicted by m_j when destandardizing Z back to X. Predictions for non-outlying z_{ij} are equal to z_{ij}.

Calculating the predictions for connected variables is more complicated. Consider the connected variable l. Then predictions for the variable l are calculated as follows.

The set C_l consists of all variables h that make a connected pair with l and l itself. Then the predictions for the connected variables are calculated as follows:

$$\hat{z}_{ij} = \frac{\sum_{h \in C_j} w_{jh} \beta_{jh} u_{ih}}{\sum_{h \in C_j} w_{jh}} \text{ for } i = 1, ..., n \text{ and } j = 1, ..., k. \tag{28}$$

Where $w_{jl} = cor_j l$. Standalone variables are skipped as well as combinations of i and j for which $u_{ij} = \text{NA}$. If there remain missing values in $\hat{Z}$ these are replaced by 0.

Step 5: Deshrinkage

According to Rousseeuw and Bossche (2018), the predictions made in (11) tend to be larger than the actual values z_{ij}. To account for this, the following computation are made. First set $a_j = robSlope(z_j^*|\hat{z}_j^*)$ then replace $\hat{z}_{ij}$ with $a_j \hat{z}_{ij}$. Where z_j^* and $\hat{z}_j^*$ represent the columns of the matrix $(z_j, \hat{z}_j)$ where all rows with at least one NA value are removed.

Step 6: Flagging Cellwise Outliers

The goal of this step is to use the predictions in $\hat{Z}$ to flag cellwise outliers. To accomplish this, the standardized cell residuals are calculated as follows:

$$\eta_{ij} = \frac{z_{ij} - \hat{z}_{ij}}{robScale(z_j^* - \hat{z}_j^*)}. \tag{29}$$

Then, all cells which satisfy the condition $|\eta_{ij}| > c_u$ are flagged as a cellwise outlier. The positions of these cells are stored in the set $I_{c,\text{DDC}}$. If for instance, the cell that intersects row a and column b is flagged as an outlier, the pair (a, b) is stored in the set $I_{c,\text{DDC}}$.

Step 7: Flagging Rowwise Outliers

The goal of this step is to use the standardized residuals in (13) to flag rowwise outliers. To accomplish this, first calculate T_i for each row as follows:

$$T_i = \frac{1}{d_i} \sum_{j=1}^{d_i} F\left((\eta_{ij})^2\right). \tag{30}$$

Where d_i is the number of η_{ij} that were able to be calculated for row i. F is the cdf of χ^2_1. Next, the T_i are robustly standardized in the same fashion as the columns of X in step 1. The rows i for which the standardized T_i is bigger than c_u^2 are flagged as rowwise outliers. The indices of these rows are stored in the set $I_{r,\text{DDC}}$.

Step 8: Impute and Destandardize

This is the final step of the DDC algorithm and its goal is to use everything calculated so far to produce new versions of the data matrix X. These versions consist of X with only the missing values imputed and of X with the missing values and cellwise outliers imputed. These new matrices will be denoted by $\mathring{X}$ and $\dot{X}$ respectively.

To retrieve these matrices, the predictions from $\hat{Z}$ have to be used first. To this end, $\mathring{Z}$ is retrieved by imputing the NA-values in Z with the corresponding cells in $\hat{Z}$. $\dot{Z}$ is retrieved by both imputing missing values and cells that were flagged as outlying. To be precise:

$$\dot{z}_{ij} = \begin{cases} \hat{z}_{ij} & \text{if } (i,j) \in I_{c,DDC} \text{ or } z_{ij} \text{ is NA} \\ z_{ij} & \text{otherwise.} \end{cases} \tag{31}$$

Then, $\mathring{X}$ and $\dot{X}$ are retrieved from $\mathring{Z}$ and $\dot{Z}$ by destandardizing. Thus, $\mathring{x}_{ij} = \mathring{z}_{ij} s_j + m_j$ and $\dot{x}_{ij} = \dot{z}_{ij} s_j + m_j$.

A.2 PCR Comparison Graphs Zoomed In

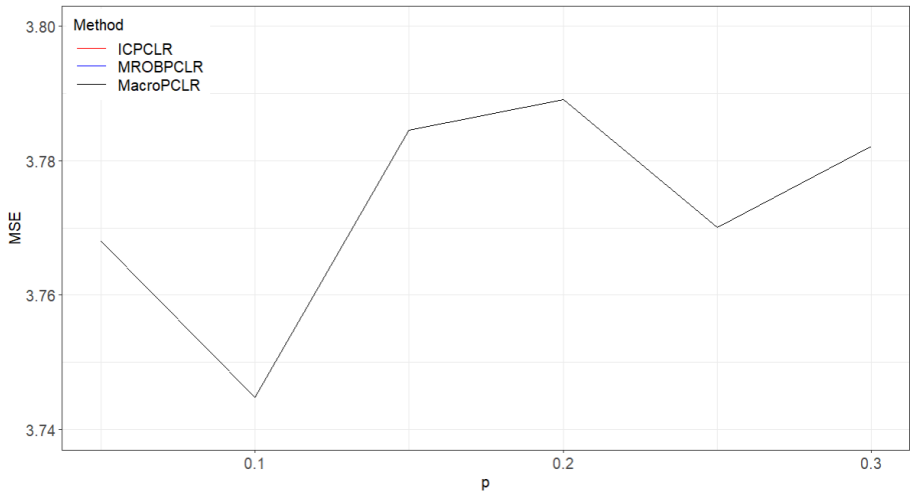


Figure 9: PCR Comparison: Scenario 1 Zoomed in

A.3 Code Description

In this section follows a short description of the created code used to conduct this research.

aux_functions.R: A file that stores auxiliary functions from the cellWise and rrcov packages. This file is called upon to create the MROBPCA2 function.

Extension.R: This file performs the PCR comparison and stores all calculated MSE values in four different dataframes, one for each scenario. These dataframes are saved as RDS files, which allows data to be stored more compactly.

extension_graphs.R: This file loads the dataframes computed in Extension.R and uses the ggplot2 package to graph the calculated MSEs.

MROBPCA.R: This file stores our implementation of the MROBPCA algorithm. Note that this function is called MROBPCA2. The MROBPCA function is present for reference.

MSE.R: This file holds a function that computes the MSE for the PCA comparison.

Replication.R: This file performs the PCA comparison and stores all calculated MSE values in four different dataframes, one for each scenario. These dataframes are saved as RDS files, which allows data to be stored more compactly.

replication_graphs.R: This file loads the dataframes computed in Replication.R and uses the ggplot2 package to graph the calculated MSEs.

robust_estimator.R: This file holds the function that calculates the One-Step M-estimators.

A.4 Abbreviations and Notations

Table 1: Table of Abbreviations

CPCA	=	Classical Principal Component Analysis.
CPCR	=	Classical Principal Component Regression.
DDC	=	Detecting Deviating Cells.
ICPCA	=	Iterative Classical Principal Component Analysis.
ICPCLR	=	Principal Component Regression which inherits ICPCA and the LR model.
LR	=	Linear Regression.
MacroPCA	=	Principal Component Analysis Accounting for Missing values And Rowwise as well as Cellwise Outliers.
MacroPCLR	=	Principal Component Regression which inherits MacroPCA and the LR model.
MCAR	=	Missingness Completely At Random.
MROBPCA	=	A ROBPCA method that can deal with missing values.
MROBPCLR	=	Principal Component Regression which inherits MacroPCA and the LR model.
MSE	=	Mean Squared Error.
PCA	=	Principal Component Analysis.
PCR	=	Principal Component Regression.
PC	=	Principal Component.
ROBPCA	=	Robust Principal Component Analysis.

Table 2: Table of Notations

n	:=	The number of rows.
k	:=	The number of columns.
τ	:=	The number of PCs selected.
$I_{c,\mathrm{DDC}}$	:=	The set of outlying cells flagged by DDC.
$I_{r,\mathrm{DDC}}$	:=	The set of outlying rows flagged by DDC.
m	:=	$(k \times 1)$ The estimated center of X.
P	:=	$(k \times \tau)$ The estimated loading matrix of X.
T	:=	$(n \times \tau)$ The estimated score matrix of X.
X	:=	$(n \times k)$ The original data matrix of explanatory variables.
$\mathring{X}$	:=	$(n \times k)$ The NA-imputed version of X.
$\dot{X}$	:=	$(n \times k)$ The NA-imputed and cell-imputed version of X.
y	:=	$(n \times 1)$ The dependent variable.
Z	:=	$(n \times k)$ The standardized version of X.
$\mathring{Z}$	:=	$(n \times k)$ The NA-imputed version of Z.
$\dot{Z}$	:=	$(n \times k)$ The NA-imputed and cell-imputed version of Z.
$\hat{Z}$	:=	$(n \times k)$ A matrix with predicted values for Z.