Identifying Significant Factors in CGM Data

Jaiden Goerlitz, Corina Ramont, Nathan Robinson, Megan Tung

February 6, 2023

1 Introduction

In the era of big data and machine learning, Principal Component Analysis (PCA), is a useful unsupervised machine learning technique that reduces the dimensionality of a data set while capturing most of the variance. In image compression, PCA is used to reduce the size, or dimension, of an image. The resolution is also reduced, but most of the significant characteristics are still identifiable and the image takes up less storage in a machine. In health insurance data, there are multiple factors, or features, taken into account when determining a client's policy. PCA is used in this case to

For this project, are dealing with a data set containing summary statistics, or metrics, drawn from Continuous Glucose Monitor (CGM) data. The monitor is a small device that measures the user's blood glucose level every five or fifteen minutes. The CGM data is relatively large, containing 48 different metrics on 210 individuals' CGM devices such as glucose variability percentage, high blood glucose index, hyperglycemia index parameters, and average daily risk range. Some of the metrics in the CGM data are highly related, or correlated (e.g. range and interquartile range, percent of data above 140 mg/dL, above 180 mg/dL, and above 250 mg/dL), which can make interpreting the original data difficult and time consuming. Applying PCA would reduce the number of features while retaining most of the variability from the original data. In particular, applying sparse PCA methods would take only the most correlated subset of original features to build the principal components, or "super-features", to make the data even more interpretable. To get started, we compared the performance of existing R PCA methods.

We have selected four Principal Component Analysis methods to evaluate, and one method for choosing the number of super-features to compute, k. The Bartlett procedure gives an integer number of super-features to retain, and acts as input for the rank of the matrix, k, for the other methods that perform sparse or nonlinear principal component analysis. These include Sparse Principal Component Analysis (SPCA), Randomized Sparse Principal Component Analysis (RSPCA), Sparse Principal Component Analysis via Regularized Singular Value Decomposition (SPCA-RSVD), and Non-Linear Principal Component Analysis (NLPCA). The aim of the simulation is to test the PCA methods performance when given a matrix of high and low rank, high and low signal to error ratio, and sparsity level, or

how many non-zero rows there are in the data matrix. After evaluating each of the PCA methods, we will apply the best methods to the CGM data.

In the next section, Methods for Comparison, we describe the different methods of principal component analysis used. In Section 3, Simulation Settings for Comparison, we describe the six different settings (set values for matrix rank, sparsity, and error) used to generate simulated data used to test the methods from Section 2. In Section 4, Evaluation Metrics, we specify the reasoning and calculations behind our accuracy measurements for the loadings, scores, rank, and values of the generated "true" matrix, and the matrix produced from the principal component analysis methods.

2 Methods for comparison

Principal Component Analysis is a dimension reduction technique that creates "super features" by creating linear combinations from the original features of the data. If $x_1, ..., x_p$ are the original features, then PCA leads to $u_1, ..., u_m$ such that

$$u_m = \sum_{j=1}^{p} w_{mj} x_j, \tag{1}$$

where w_{mj} are the weights to for the new m-th super-feature.

PCA outputs include the reduced dimension number m, which must have $m < min(n, p)$, the weight matrix $W \epsilon \mathbb{R}^{pxn}$ (also called the loadings matrix), and U, the new features matrix $U \epsilon \mathbb{R}^{pxn}$ (also called the scores matrix).

The scores and loadings are created by

$$X \approx UW^T \text{ or } u_{ik} = \sum_{j=1}^{p} w_{jk} x_{ij} \tag{2}$$

for the i-th sample, the new k-th feature is formed as a linear combination of the original j features taken with weights. There are m total such features. The summation of the weights squared (the columns) are orthogonal. Because $X \approx UW^T$, the PCA methods assume

$$X = UW^T + E \tag{3}$$

where E is a matrix of random errors. We first look to find k and then apply the k to our chosen methods that are detailed below.

2.1 Selection Method: nBartlett

The nBartlett method computes k, which is a required input for all of the PCA methods. This function computes the Bartlett, Anderson, and Lawley indices for determining the number of principal components to retain. (Bartlett, 1950). This method is used to find the targeted rank k, which will be used in the below methods' inputs. The hypothesis tested is:

$$H_k : \lambda_{k+1} = \ldots = \lambda_p \tag{4}$$

The hypothesis test is testing the significance of the eigenvalues obtained from a correlation matrix. The hypothesis test is confirmed by applying a χ^2 test. If the test is accepted then the data matrix can be sorted into linear combinations of variables. Though the function returns three different possible values for k, we are only using the Bartlett output for k estimation. To implement, we use nBartlett within nFactors R package (Raiche and Magis, 2020).

2.2 Sparse PCA (SPCA)

Sparse PCA produces modified principal components with sparse loadings. SPCA attempts to find the loadings vector with only a few "active" non-zero values. In addition, SPCA tries to reduce dimensionality and the number of explicit variables. A drawback of PCA is that each principal component produced is typically non-zero, which makes them hard to interpret (Erichson et al., 2020). Furthermore, SPCA avoids over-fitting a high-dimensional data set where the number of variables is greater than the number of observations. Given an (n, p) data matrix, SPCA minimizes the following function:

$$f(A, B) = ||X - XBA^T||_f^2 + \psi B \tag{5}$$

where B is the sparse loading matrix and A is the orthonormal matrix. *psi* denotes sparsity. The principal component, Z is denoted by

$$Z = XB \tag{6}$$

To implement, we use function spca within sparsepca R package (Erichson et al., 2018).

2.3 Randomized Sparse PCA (RSPCA)

Randomized Sparse PCA is a randomized accelerated version of Sparse PCA. RSPCA is similar to SPCA, which both use variable projection as an optimization strategy.

RSPCA differs by using randomized methods of linear algebra to speed up the computations. An argument for RSPCA is o, which is the oversampling parameter to improve the approximation (Erichson et al., 2020). The default is $o = 20$. If $k > (min(n, p)/4)$, SPCA might be faster. Also, RSPCA differs by using the argument q which specifies the number of subspace iterations to reduce the approximation error. To implement, we use function rspca within sparsepca R package (Erichson et al., 2018).

2.4 Sparse PCA via Regularized Singular Value Decomposition (SPCA-RSVD)

Sparse PCA via Regularized Singular Value Decomposition is a method of sparse PCA similar to previous methods, and performs PCA on a given matrix based on singular value decomposition. However, it assigns sparsity based on a few different threshold methods: hard, soft, and scad. The hard method assigns elements in the output loadings matrix to

0 if their absolute value is below a certain threshold. The soft method does the same, but penalizes the remaining non-zero values by subtracting the threshold from its original value. The SCAD, or smoothly clipped absolute derivations, method applies penalty similar to the soft method, but decreases the penalty as the absolute value of the coefficient increases using different quadratic equations. (Shen and Huang, 2008).

To implement this method, we use the function sPCA_rSVD in the ltsspca R package (Wang et al., 2019). The default threshold setting is "hard", and is currently the only functioning threshold method for this package, so it is the only threshold method we will be performing.

2.5 Nonlinear PCA (NLPCA)

Nonlinear PCA trains a neural network, created by the loadings, that generalizes the principal components from straight lines to curves. The goal is to find a curve that goes through the multidimensional space of X that incorporates as much variance as possible. Consequently, NLPCA does not return a loadings matrix unlike other PCA methods; meaning classical ways of interpreting PCA does not apply to NLPCA. To implement, we use function nlpca within pcaMethods R package (Stacklies et al., 2007).

3 Simulation settings for comparison

Four total matrices are generated: U, D, V^T, and E to create our X matrix.

For a matrix $\tilde{U}$, its elements are drawn from a normal(0, 1). SVD is then performed on $\tilde{U}$ and retrieving its first k singular column vectors from one of its resulting matrices. The result is a n by k matrix U containing the scores.

For a matrix $\tilde{V}$, its elements are drawn from a normal(0, 1). SVD is then performed on $\tilde{V}$ and retrieving its first k singular column vectors from one of its resulting matrices. The result is a p by k matrix V containing the loadings or weights.

The matrix D, with k rows and k columns, is a diagonal matrix that determines the signal strength of the overall X matrix and the strength of each "superfeature".

Lastly, by randomly drawing from a normal(0) with varying standard error, this forms n by p error matrix, E.

The n by p matrix X then results from:

$$X = UDV^T + E, \tag{7}$$

All of the settings have a size of $n = 210$ and total of $p = 48$ features. The following settings are used:

1. $k = 3, s = 3, sd = 0.1, diagonal = [50, 10, 4]$
2. $k = 3, s = 3, sd = 0.5, diagonal = [50, 10, 4]$
3. $k = 8, s = 8, sd = 0.1, diagonal = [36, 25, 20, 15, 10, 5, 3, 2]$

4. $k = 8, s = 8, sd = 0.5, diagonal = [36, 25, 20, 15, 10, 5, 3, 2]$

5. $k = 3, s = 5, sd = 0.1, diagonal = [50, 10, 4]$

6. $k = 3, s = 10, sd = 0.1, diagonal = [50, 10, 4]$

The values are chosen based on three characteristics we wanted to compare:

1. A data matrix with a high vs. low number of superfeatures

2. A data matrix with that is noisy vs. less noisy

3. And lastly, a data matrix has more sparse loadings vs. less sparse loadings.

4 Evaluation metrics

Four metrics were considered.

First, we measure the accuracy of our target rank estimator $\hat{k}$ by taking the proportion:

$$(k - \hat{k})/k. \tag{8}$$

The lower the proportion, the closer the value of $\hat{k}$ is to the true k.

Second, we compare our generated $U\hat{D}V$ matrix to the true UDV matrix in a similar way by the equation

$$\frac{1}{n * p}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p}((UDV^T)_{ij} - (U\hat{D}V^T)_{ij})^2} \tag{9}$$

We divide by the matrix dimensions $n * p$ to standardize the metric. Again, the lower the proportion, the more accurate $U\hat{D}V$ is.

Additionally, the true scores, UD, are compared to estimated scores $U\hat{D}$, and is measured by the following:

$$\frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p}((UD)_{ij} - (U\hat{D}^T)_{ij})^2}. \tag{10}$$

Lastly, we compare our estimated loadings $\hat{V}$ relative to the true V. We use the following equation:

$$\frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p}((VV^T)_{ij} - (V\hat{V}^T)_{ij})^2}, \tag{11}$$

where VV^T and $V\hat{V}^T$ are the projection matrices of for V and $\hat{V}$. This chordal distance is the final measure of accuracy, with a lower value indicating higher accuracy.

Table 1: Estimation of K metrics

setting	value
Sim1	-11.04 (0.526)
Sim2	-11.04 (0.526)
Sim3	-4.86 (0.0036)
Sim4	-4.77 (0.0097)
Sim5	0 (0)
Sim6	-14.33 (0.0335)

Table 2: Estimation of Scores Metrics

setting	nlpca	rspca	spca	spca_rsvd
Sim1	1.56 (0.0035)	1.46 (0.0046)	1.46 (0.0046)	1.42 (0.0058)
Sim2	1.56 (0.0035)	1.46 (0.0046)	1.46 (0.0046)	1.42 (0.0058)
Sim3	2.45 (0.0039)	2.3 (0.0054)	2.3 (0.0055)	2.32 (0.0069)
Sim4	2.5 (0.0033)	2.36 (0.0043)	2.36 (0.0043)	2.38 (0.0053)
Sim5	1.4 (0.0063)	1.18 (0.0098)	1.18 (0.0097)	1.13 (0.0127)
Sim6	1.27 (0.0116)	1.01 (0.0015)	1.01 (0.0016)	1.01 (0.001)

5 Simulation Results

Referring to Table 1 and Figure 1 (left), all the mean values for the Bartlett method are negative or zero, meaning this method is is highly likely to overestimate the rank. Simulations 1 and 2 have the same error, and simulations 3 and 4 have similar error, where the only parameter changing is the error added to the true data. This indicates the consistent nature of the Bartlett method, however, the error has a high variance in comparison to the rest of the simulations. The error in simulations 1 and 2 are higher than that of simulations 3 and 4, where the rank and sparsity level (equal to the rank) is higher in simulations 3 and 4. Simulation 5 has a lower sparsity level than simulation 6, all other factors held the same. Surprisingly, the error for estimation of rank in simulation 5 is zero, with no variation. This is contrasted in simulation 6 that has the highest error, as well as the highest sparsity level of all simulations.

In Table 2 and Figure 1 (right), simulations 3 and 4 had the largest mean value for scores error across all methods, being the ones with the highest rank ($k = 8$). For each simulation setting, the non-linear PCA method had the highest mean value, but the lowest standard error. This indicates that NLPCA had consistent incorrect estimates. It is likely due to how NLPCA generalizes PCs as curves rather than straight lines. The sparse methods performed similarly, the RSPCA and SPCA methods had a mean value and standard deviation of error being nearly the same. In comparison, SPCA-RSVD had the highest standard error but relatively lower mean values than the other methods. SPCA-RSVD did the best out of all the methods.

Table 3 and Figure 2 (left) shows the errors of the sparse PCA methods recreated X matrix. Overall, the errors are much lower than the scores and loadings individually. Sim-

Table 3: Estimation of UDV metrics

setting	rspca	spca	spca_rsvd
Sim1	0.0063 (2.3e-05)	0.0063 (2.3e-05)	0.006 (2.8e-05)
Sim2	0.0063 (2.3e-05)	0.0063 (2.3e-05)	0.006 (2.8e-05)
Sim3	0.0055 (1.3e-05)	0.0055 (1.3e-05)	0.0053 (1.9e-05)
Sim4	0.0057 (1.2e-05)	0.0057 (1.2e-05)	0.0055 (1.6e-05)
Sim5	0.0051 (3.9e-05)	0.0051 (3.8e-05)	0.0047 (5.1e-05)
Sim6	0.0039 (2.4e-05)	0.0039 (2.4e-05)	0.0036 (2.6e-05)

Table 4: Estimation of V metrics

	setting	rspca	spca	spca_rsvd
1	Sim1	1.53 (0.0039)	1.53 (0.0038)	1.46 (0.006)
2	Sim2	1.53 (0.0039)	1.53 (0.0038)	1.46 (0.006)
3	Sim3	2.41 (0.0046)	2.41 (0.0046)	2.41 (0.0046)
4	Sim4	2.43 (0.004)	2.43 (0.004)	2.43 (0.004)
5	Sim5	1.31 (0.0082)	1.31 (0.0081)	1.21 (0.0123)
6	Sim6	1.16 (0.004)	1.16 (0.004)	1.12 (0.0044)

ulations 3 and 4 had a high rank ($k = 8$) that produced marginally lower mean errors and variance of errors than the lower rank simulations 1 and 2. Between simulations 5 and 6, the error is most significantly the lowest with a lower sparsity level (simulation 6) across all methods. SPCA-RSVD method consistently had the lowest error for each simulation, but the differences between methods are very small. Overall, the estimation of $U\hat{D}V$ was consistent across sparse methods.

In Table 4 and Figure 2 (right), error of the loadings is measured by the chordal distance. Only the sparse methods are displayed because the NLPCA method does not return the calculated loadings. Similar to the scores metrics, the highest error occurred in the simulations with the high rank value ($k = 8$) which were settings 3 and 4. Interestingly, RSPCA and SPCA methods had the same distribution errors of loadings metrics for each simulation settings. Furthermore, all sparse PCA methods returned the same error distribution for V metrics only for simulations 3 and 4. This suggests these methods function similarly such as assigning weight. For lower rank simulation settings, SPCA-RSVD method performed marginally better in terms of errors.

6 Data Application

The CGM data contains 210 observations, each with 48 features and values such as the mean, the IQR, and percentage of time where the raw CGM data was above 180 d/mL.The data had one missing value, so before applying our PCA methods to the data, we replaced the missing value with the mean of the observations for that feature (MAGE). Then, we centered and scaled the data.

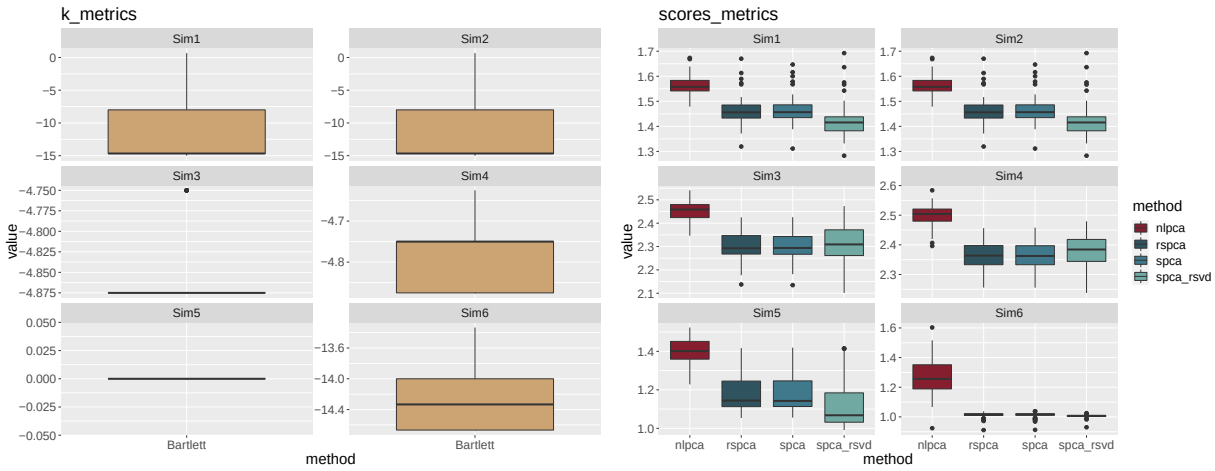

Figure 1: k (left) and scores (right) metrics for each simulation setting

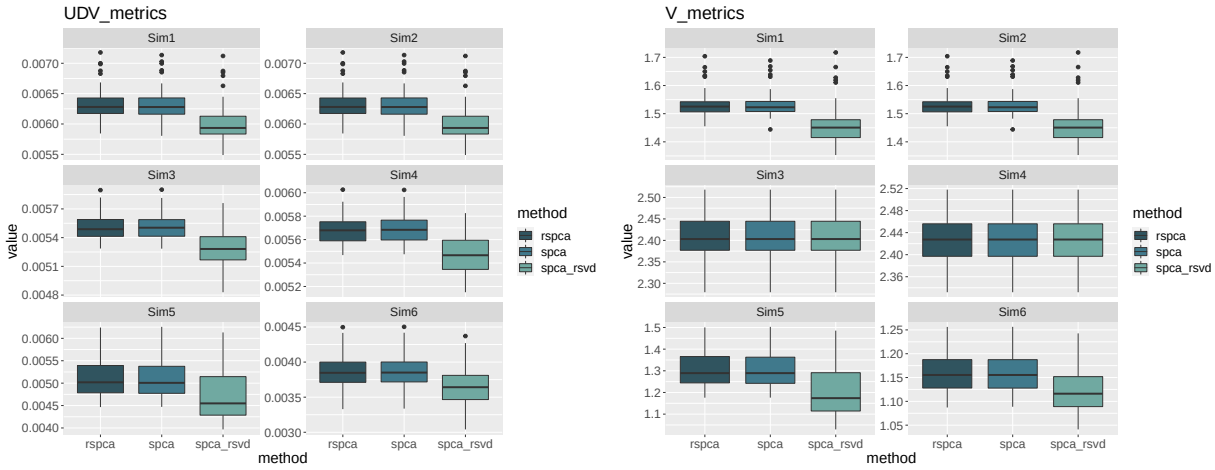

Figure 2: UDV (left) and V (right) metrics for each simulation setting

The Bartlett method of rank selection chose 30 super-features to retain, which is extremely high considering the maximum amount of super-features is 48. However, from simulations, we know that this method tends to overestimate the true rank of the matrix, so we proceed with caution. We plot the first two principal components which captured the highest proportion of variance. We see that the spread of the data is similar for all sparse methods where the second principal component's variance seems to be roughly half of the variance from the first principal component. The NLPCA plot follows this trend as well, but is an exception to the rest because of the drastically lower variance the first two principal components display. This may be because the goal of NLPCA is to find a nonlinear curve that explains most of the variance, not multiple linear combinations as the rest of the methods do.

Examining which original features were included in the first few super-features from the

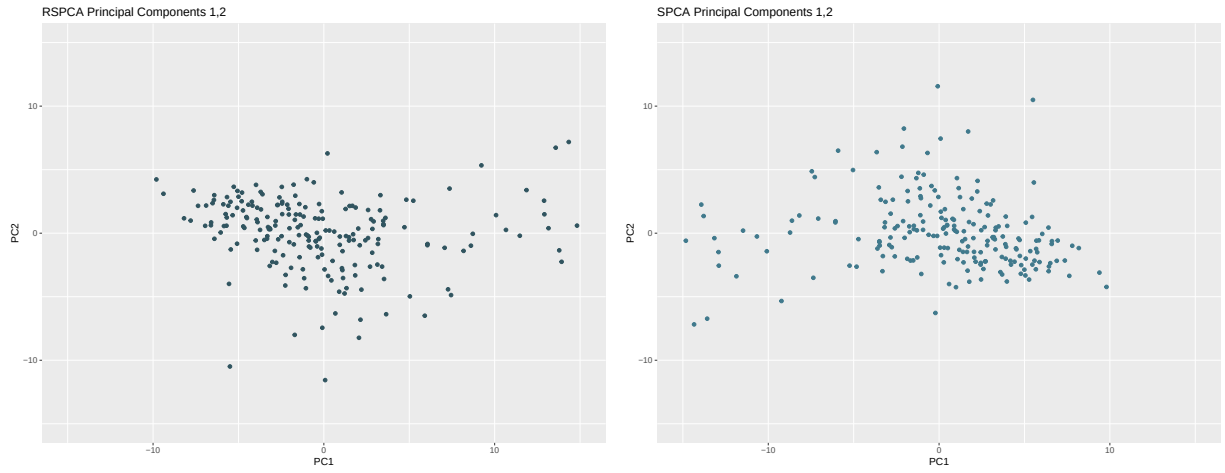

Figure 3: PC plots of RSPCA (left) and SPCA (right)

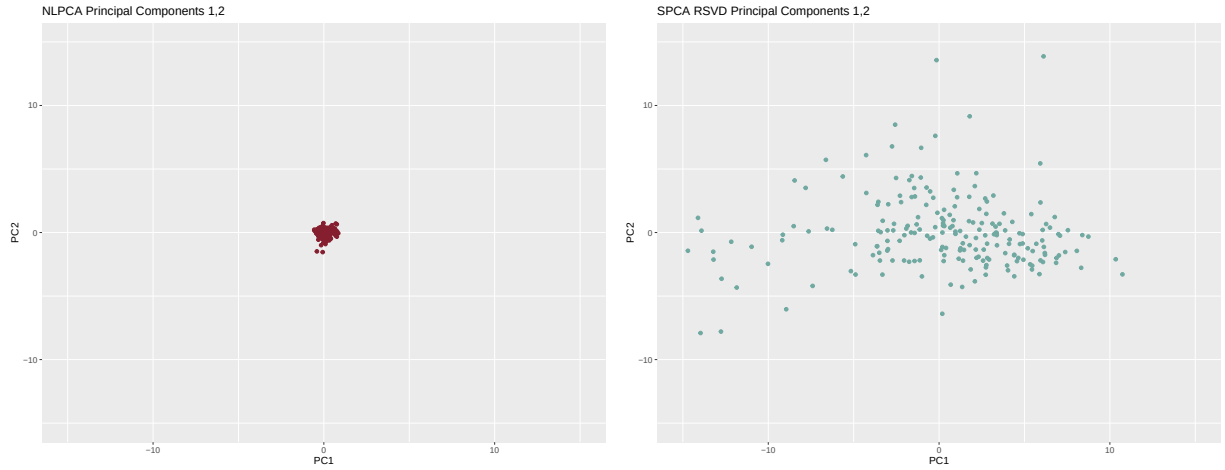

Figure 4: PC plots of NLPCA (left) and SPCA-RSVD (right)

sparse PCA methods, we find the results are not actually sparse. In the first principal component, RSPCA and SPCA chose to include the same 43 of the 48 original features. The excluded features were the Coefficient of Variation of glucose levels (CV), percentage of GRADE score attributable to hypoglycemia (grade_hypo), the percentage of values below 54 (below_54), the percentage of values below 70 (below_70), and the calculated Hypoglycemia Index (hypo_index). SPCA-RSVD chose more original features to include, 45 of the original 48. They were the same as RSPCA and SPCA, but excluded CV and grade_hypo. Interestingly, all these features are associated with lower blood glucose levels. The trend follows for the following few super-features. In the second principal component, RSPCA and SPCA chose the same 44 features to retain, and SPCA-RSVD 46. RSPCA and SPCA excluded the percentage of values between 70 and 180 (in_range_70_180), and the percentage of GRADE score attributable to hyperglycemia (GRADE_hyper), and all three sparse methods excluded

the percentage of GRADE score attributable to target range (GRADE_eugly) and the calculated Index of Glycemic Control (igc). These features also seemed to be associated with the upper range of possible values. In the third principal component, RSPCA and SPCA choose the same 38 features to retain, and SPCA-RSVD chooses 43. At the fourth principal component, the results become more sparse, with SPCA and RSPCA choosing 25 original features and SPCA-RSVD choosing 28, but it is not enough to say this principal component is strictly sparse.

Overall, the sparse methods performed better than NLPCA in terms of our goals. However, the three sparse methods' performance was consistent with each other, all producing non-sparse results. When aiming for more sparsity, RSPCA and SPCA seem to produce exactly the same results, choosing slightly less features than SPCA-RSVD. However, from simulations, SPCA-RSVD had the lowest error for scores, loadings, and recreated matrix, so it may construct the underlying matrix better than RSPCA and SPCA.

7 Discussion

From completing this project, we encountered several difficulties and made adjustments to our methods in simulations due to time constraint of the project. Some limitations to the simulations include restricting the maximum number of steps to 1000 for the NLPCA method, and restricting the number of replications in each simulation to 100. Without taking these measures, the simulations would take significantly longer and put strain on our machines. As a good practice, we found X matrices need to be centered and scaled before the PCA methods were applied. Interestingly, Table 1 also shows that the Bartlett method tends to overestimate $\hat{k}$ consistently, which was our biggest finding. For future research, we recommend considering a different method of selecting k for this data, increasing the number of NLPCA iterations, and running additional simulation repetitions for better accuracy.

References

Bartlett, M. S. (1950). Tests of significance in factor analysis. *British journal of psychology*.

Erichson, N. B., P. Zheng, and S. Aravkin (2018). *sparsepca: Sparse Principal Component Analysis (SPCA)*. R package version 0.1.2.

Erichson, N. B., P. Zheng, K. Manohar, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin (2020). Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics 80*(2), 977–1002.

Raiche, G. and D. Magis (2020). *nFactors: Parallel Analysis and Other Non Graphical Solutions to the Cattell Scree Test*. R package version 2.4.1.

Shen, H. and J. Z. Huang (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis 99*(6), 1015–1034.

Stacklies, W., H. Redestig, M. Scholz, D. Walther, and J. Selbig (2007). pcamethods – a bioconductor package providing pca methods for incomplete data. *Bioinformatics 23*, 1164–1167.

Wang, Y., S. Van Aelst, H. C. Valdiviezo, and T. Reynkens (2019). Package 'ltsspca'.