Project 2 NLA: SVD applications

Jaume Sánchez

December 2023

Contents

1 Introduction

This report corresponds to the study carried out solving the second Numerical Linear Algebra project. In this, you can find both theoretical answers and the explanation and analysis of the results of the practical part (code part).

The goal of this project is to discuss three common applications of the SVD decomposition: LS problem, graphics compression and PCA.

2 A little explanation

Let us briefly introduce what we have delivered to the professor. For each section of the project we have delivered one file (*.py*):

1. *Prac2_least_squares.py* for the first section.
2. *Prac2_graphics_compression.py* for the second section.
3. *Prac2_PCA.py* for the third section.

Later (in the corresponding section) we will explain in more detail what each function is for.
We also delivered 3 original images used in section 2 (and the ones generated) and some plots that we got while executing the code provided in this delivery.

3 Least Squares problem

In this first section we need to read data from two different datasets and then apply QR factorisation and SVD methodology, comparing them, in order to solve the LS problem.
To do this, we have created four functions:

1. *read_dades(degree)*: reads the *dades* file and generates the corresponding matrix A and vector b.
2. *read_csv()*: reads the *dades_regressio* file and returns the corresponding matrix A and vector b.
3. *QR(A,b)*: given a matrix A and a vector b returns the QR solution, x, that solves $QRx = b$.
4. *SVD(A,b)*: given a matrix A and a vector b returns the solution $x = A^{+}b$ where A^{+} is the pseudoinvers calculated using SVD.

Let us talk about the first dataset: *dades*. It consists of 500 points (x, y) (see Figure 1) , which we want to use to compute the best polynomial fitting. To do so and given a degree d, we create a matrix (using *read_dades(degree)*) A and a vector b following the next scheme:

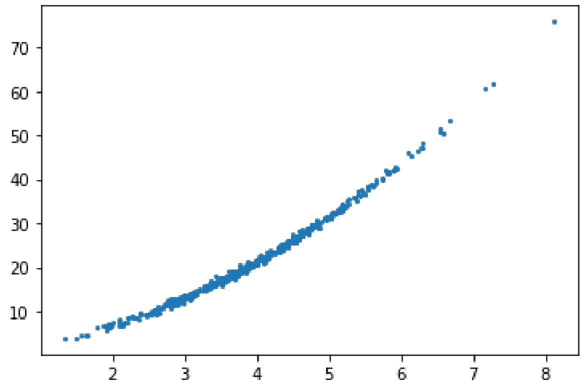


Figure 1: Data from *dades* file.

$$A = \begin{pmatrix} 1 & x_1^1 & \cdots & x_1^{d-1} \\ 1 & x_2^1 & \cdots & x_2^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & \cdots & x_n^{d-1} \end{pmatrix} \; ; \; b = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

We first tried for different degrees, from 2 to 20, both for QR and for SVD. We selected the degree that had the smaller error, that is, the smaller value of $||Ax - b||_2$. The results obtained were the following:

1. QR method, best degree: 11, error value: 10.818267948014958
2. SVD method, best degree: 12, error value: 10.79578096823541
3. SVD method, degree: 11, error value: 10.818267951572489

Although in the SVD method we got that 12 was the best degree, we can see that the difference with degree 11 is minimal. This is why, in order to be able to compare both solutions, we computed the solution with 11 as a degree. Note that the difference (the norm of the difference vector), with 11 as a degree, between the two methods is only of $1,42 \cdot 10^{-4}$.

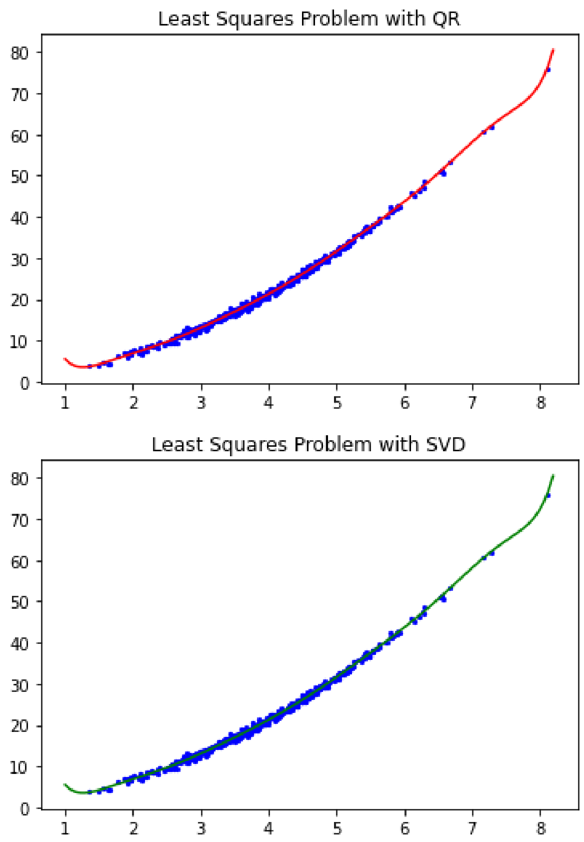


Figure 2: Solution to Least Squares problem using QR and SVD, both with degree $d = 11$.

About the second dataset, *dades_regressio*, the data is already given, that is, we do not need to create any matrix; just read it from the file. Although the norm of the difference vector, using QR and SVD, is approximately 11.75 both solutions give us a similar error of 1.14 identical to the tenth decimal.

4 Graphics compression

Let us first answer the theoretical questions of this section. We will properly state two statements and then prove them.
Definition 4.1 Let $A \in \mathbb{R}^{m\times n}$ be a real matrix with $m \geq n$, and let $r \geq 0$ be the rank of A. Then, for $k = 1, ..., r$ we define the k rank approximation of A as the following matrix:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T$$

where u_i and v_i denote the i-th column of U and V respectively and σ_i is the i-th singular value of A.

Theorem 4.2 Let $A \in \mathbb{R}^{m\times n}$ be a real matrix with $m \geq n$, and that $A = U\Sigma V^T$ is the singular value decomposition of A. Then, the best rank k approximation to A in the Frobenius norm, $||\cdot||_F$, is given by A_k, that is:

$$||A - A_k||_F \leq ||A - B||_F$$

for any matrix B of rank at most k.
Proof. To start with, note that as the norm is invariant for othogonal matrices (U and V), we have that

$$||A - A_k||_F^2 = ||\Sigma - \Sigma_k||_F^2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2$$

Also, by the triangle inequality we have that, for every matrix X, Y and being $\sigma_i(C)$ the i-singular value of the matrix C:

$$\sigma_i(X) + \sigma_j(Y) \geq \sigma_{i+j-1}(X + Y)$$

Now, taking into account that $\sigma_{k+1}(B) = 0$ (remember that B is a matrix of at most rank k), if we choose X to be $A - B$ and Y to be B we obtain that for $i \geq 1$, $j = k + 1$:

$$\sigma_i(A - B) \geq \sigma_{k+1}(A)$$

Finally,

$$||A - B||_F^2 = \sum_{i=1}^{n} \sigma_i(A - B)^2 \geq \sum_{i=k+1}^{n} \sigma_i(A)^2 = ||A - A_k||_F^2$$

□

Theorem 4.3 Let $A \in \mathbb{R}^{m\times n}$ be a real matrix with $m \geq n$, and that $A = U\Sigma V^T$ is the singular value decomposition of A. Then, the best rank k approximation to A in the 2-norm, $||\cdot||_2$, is given by A_k, that is:

$$||A - A_k||_2 \leq ||A - B||_2$$

for any matrix B of rank at most k.
Proof. Reasoning similarly to the previous proof we have that

$$||A - A_k||_2^2 = ||\Sigma - \Sigma_k||_2^2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\|_2^2 = \sigma_{k+1}^2$$

Since we have that $\sigma_i \geq \sigma_j$ if $i < j$. As before, suppose B is a matrix of rank k; then there exist a vector that belongs to the intersection between the space generated by the first $k+1$ columns of V and the null space of B, whose dimension is $n-k$. Without loss of generality we can scale this vector, w, so that $||w||_2 = 1$. Suppose then that $w = \sum_{i=1}^{k+1} \alpha_i v_i$ such that v_i are the columns of V and $\sum_{i=1}^{k+1} \alpha_i = 1$. Then,

$$||A-B||_2^2 \geq ||(A-B)w||_2^2 = ||Aw||_2^2 = \textstyle\sum_{i=1}^{k+1} \alpha_i^2 \sigma_i^2 \geq \sigma_{k+1}^2 = ||A-A_k||_2^2$$

$\square$

Once we have dealt with the theoretical questions, let us focus with the practical questions and the obtained results.

In this section we have used the SVD method to obtain a lossy compressed graphic image from a .jpeg graphic file. To do so, we have created just one function SVD(image,k,flag), that receives the image path, the k singular values that we want to keep and a flag, that will be set to 1 if we want the new image to be saved and any other value if not. This function returns the error done while compressing the image, the the amount of norm that is conserved in the compression process, and also, if the flag is set to 1, saves the compressed image in the corresponding folder.

Let us first see the original images.

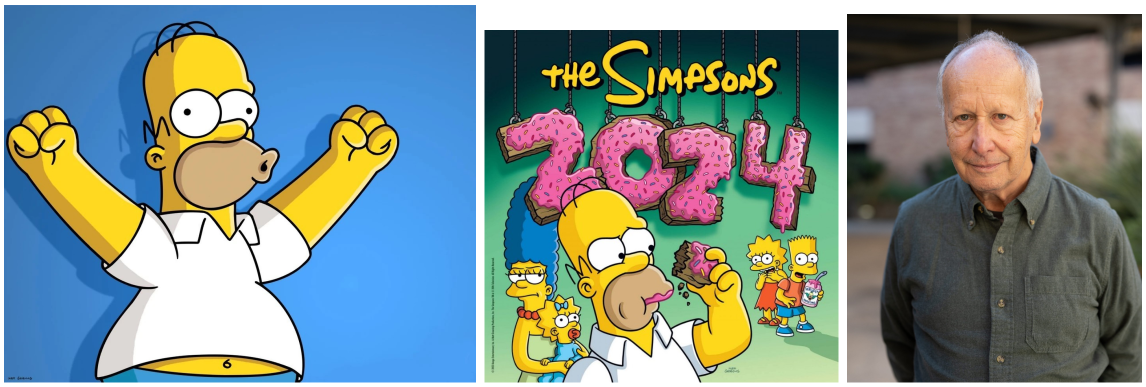


Figure 3: Original images to be compressed.

The names of these images are the following: *Homer, Letters and Nobel.* Note that they have different sizes: 1200x960, 796x793 and 2048x2560 respectively and also different weights: 111 KB, 183KB and 371 KB respectively.
We have selected a few images:

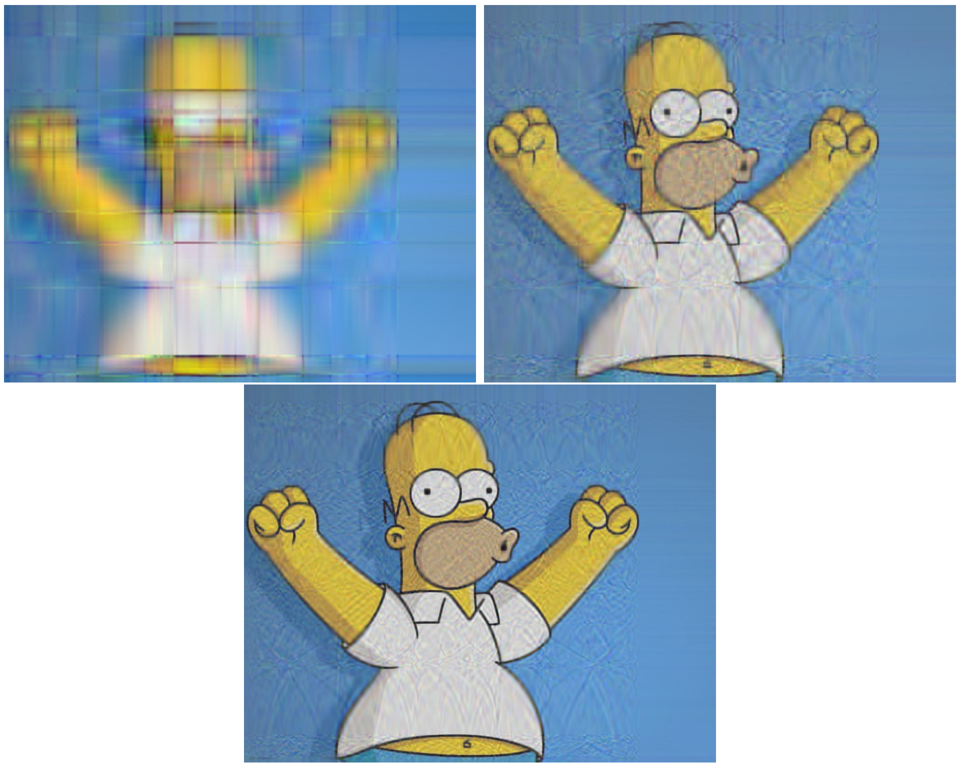


Figure 4: $k = 5$, $k = 25$, $k = 50$ from left to right.

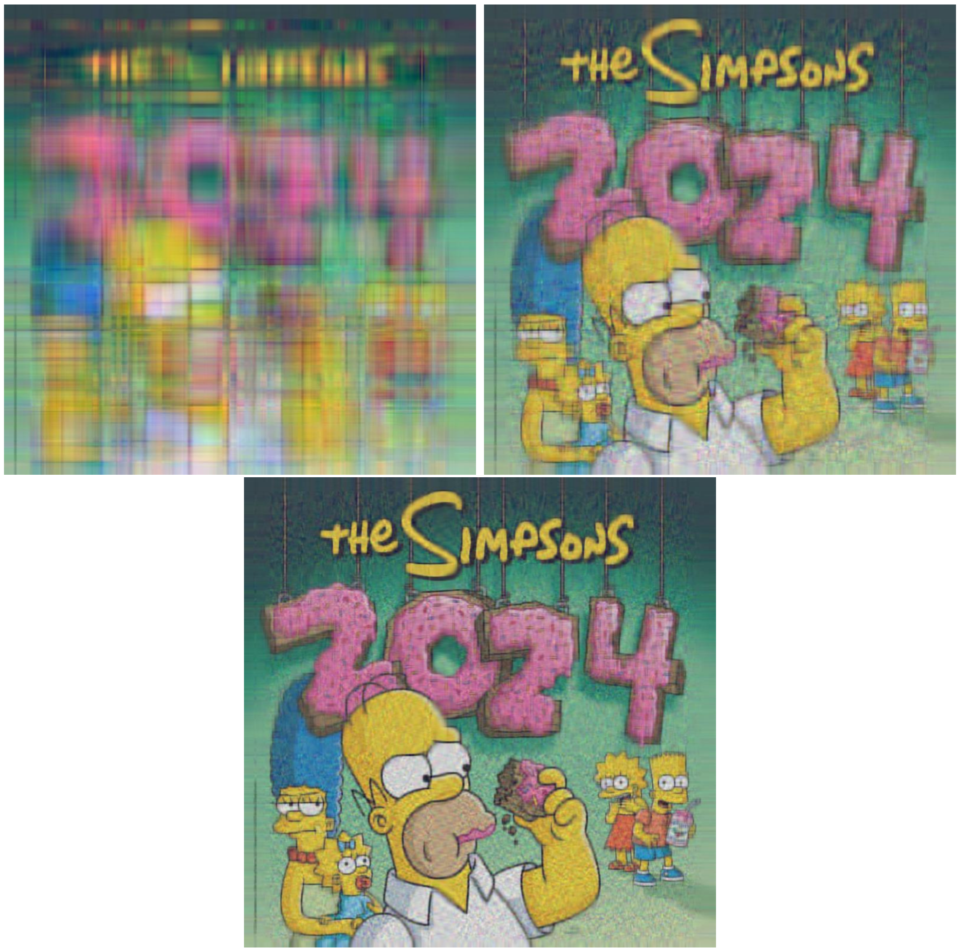


Figure 5: $k = 5$, $k = 25$, $k = 50$ from left to right.

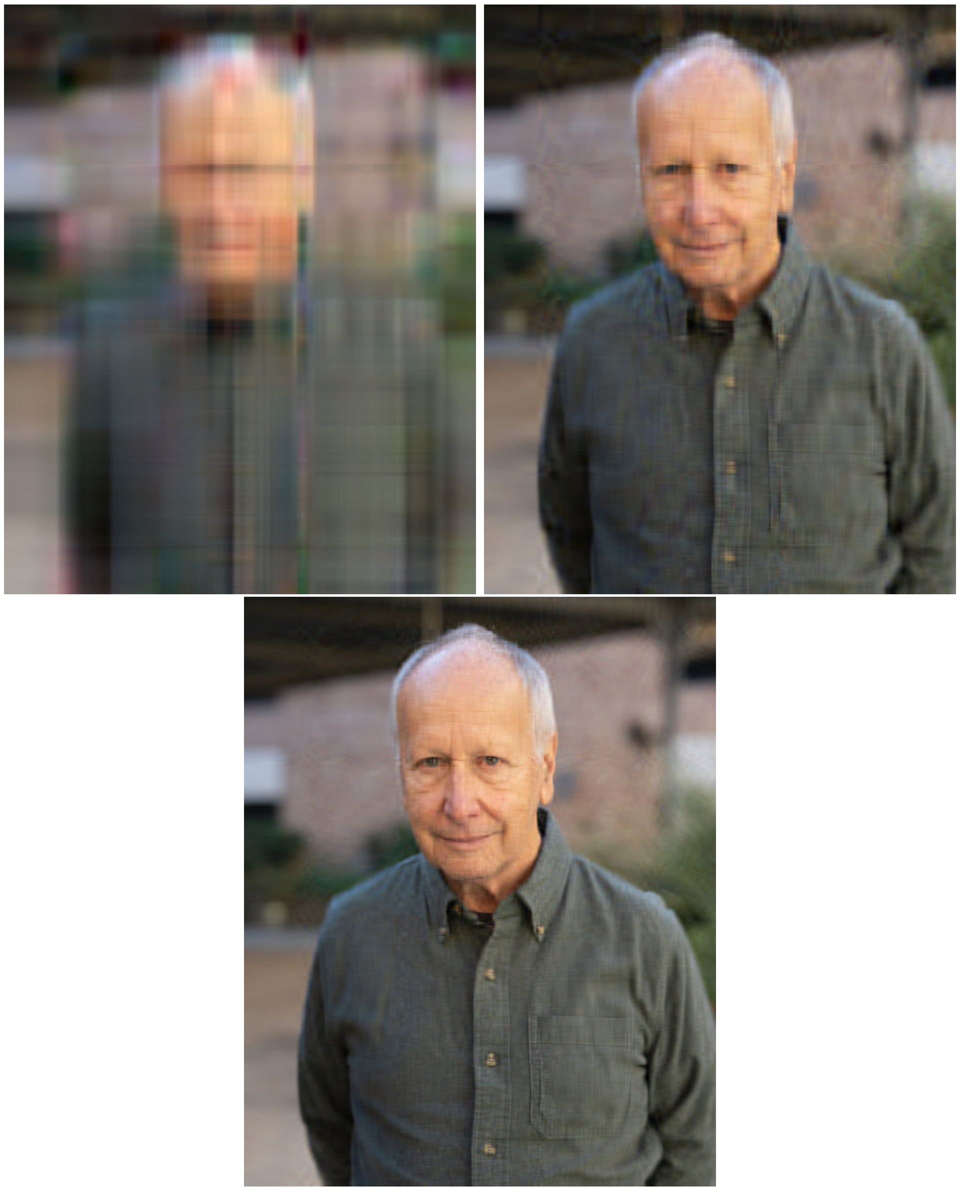


Figure 6: $k = 5$, $k = 25$, $k = 50$ from left to right.

We can clearly see (as expected) how as the value of k increases the image looks more similar to the original one. Let us see the error plots of each image:

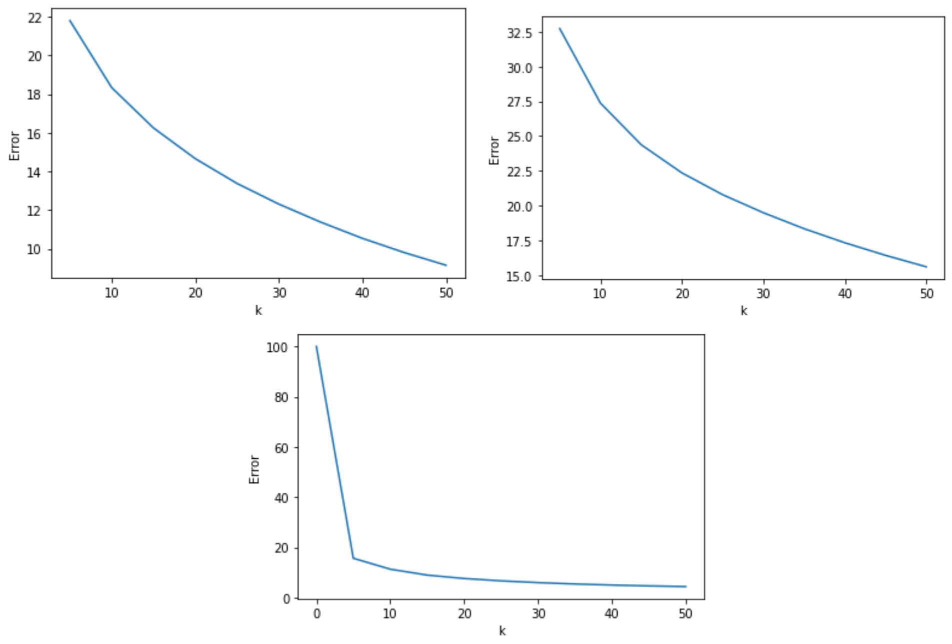


Figure 7: Error plots while k increases.

The order of the plots is as follows: *Homer, Letters, Nobel*. We can see that both shares the same: the error decreases as k increases. Note that although the same k criteria has been used, the last ”Letters” compressed image looks ”blurrier” than the ”Nobel” last compressed image, so of course, the results will depend on the selected image. That is, it depends on the characteristics or details of the image it may need to keep more k singular values.

5 Principal component analysis (PCA)

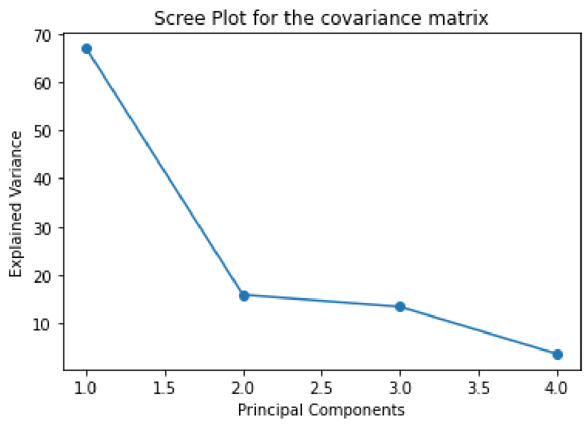
In this last section of the project, we want to use the principal component analysis technique to detect the main components of a data set in order to reduce it into fewer dimensions retaining the relevant information. To do that, we have created the following functions:

1. *read_txt()*: reads the *example.dat* file and transposes the data.
2. *read_csv()*: reads the *RCsGoff.csv* and returns only the relevant information.
3. *PCA(data,case)*: given the first dataset or the second dataset (*data* it generates the PCA analysis taking into account the covariance matrix or the correlation matrix (*case*). That is, it returns portion of the total variance accumulated in each of the principal components, the standard deviation of each of the principal components and the expression of the original dataset in the new PCA coordinates.
4. *Kaiser(S)*: given a vector it returns the amount of the selected principal components by the *Kaiser* rule and also which PC they are.
5. *rule_34(vec)*: given a vector it returns the amount of the selected principal components by the $\frac{3}{4}$ rule and also how much variance they add up.

Let us talk about the first dataset. It consists of 16 observations of 4 variables, that is, in order to be consistent with the theoretical comments in the project we will have to transpose this data. Although it is not a huge dataset we would want to know if we can reduce this data.
Once we apply the corresponding analysis to the covariance or correlation matrix we have three "rules" that can help us to decide whether retaining a dimensions is worth it or not, or in other words,to determine how many principal components to retain. They are the following ones:

1. *Kaiser rule*: The rule suggests that you should retain only those principal components whose eigenvalues are greater than 1. Note that the eigenvalues represent the amount of variance explained by each principal component.
2. The *3/4 of the total variance* rule suggests that you should only keep components until you reach the 75 % of the total variance.
3. *Scree plot*: The main idea is to look at the "knee" or "elbow" on the graph, which is the point where the slope of the curve drops significantly. This point is where the eigenvalues stop decreasing rapidly, and retaining more components will not provide a substantial improvement in explained variance. The commonly used decision rule in the Scree plot is to retain the principal components whose eigenvalues are above the elbow.

With all this being said, let us have a look at the results of the first dataset.
We obtain the following Scree plots:



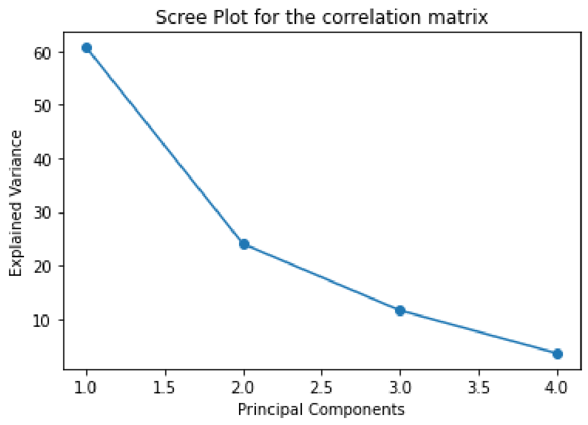


Figure 8: Scree plots for the first dataset

Note that the total variance accumulated in each of the principal components is the following: [60.76, 24.03, 11.62, 3.6] for the first plot and [66.98, 15.89, 13.45, 3.68] for the second one. Thus, following the Scree plot rule previously mentioned, we would select the first two principal components in both cases. Following the Kaiser rule, obtained that in the first case we would choose 3 principal components (the last three) and in the second case just two principal components (the second and the third one). And last but not least, the 3/4 of the total variance rule suggest us to choose the first two principal components in both cases as $60.76 + 24.03 > 75$ and $66.98 + 15.89 > 75$.
As for the second dataset we obtained the following results:

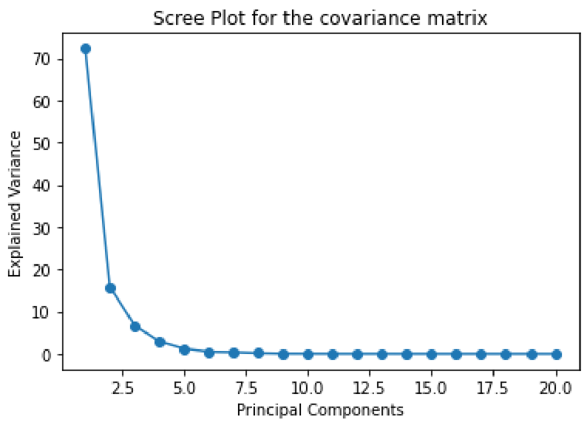


Figure 9: Scree plot for the second dataset

Again, the total variance accumulated for the first 10 principal components is: [72.3, 15.78, 6.69, 2.89, 1.23, 0.46, 0.36, 0.16, 0.04, 0.04]. Trying to find the "elbow" one could say that we should keep the first four or five principal components. Also, Kaiser rule now suggest us to keep 19 principal components (and hence just deleting one) and as $72.3 + 15.78 > 75$ the last criterion would suggest us to take just 2 principal components.
Finally, and in order to corroborate the results obtained, we now show the plot shown in the project:

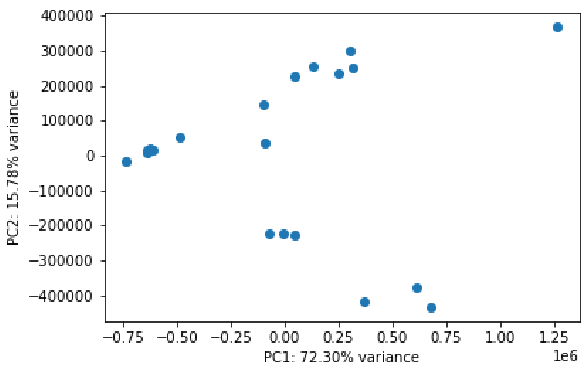


Figure 10: PC1 and PC2

To finish this section, we answer the last question in the format that the project asks us to do. (See the last line code of the main).

6 Dificulties

In general terms, I consider that this practice has been an enjoyable project. We have been able to see direct applications of the SVD method. If I had to highlight any difficulty it would be in the last exercise where on some occasions the data had to be transposed.