The pcaMethods Package

Wolfram Stacklies and Henning Redestig
CAS-MPG Partner Institute for Computational Biology (PICB)
Shanghai, P.R. China
and
Max Planck Institute for Molecular Plant Physiology
Potsdam, Germany
http://bioinformatics.mpimp-golm.mpg.de/

May 3, 2016

Overview

The `pcaMethods` package [1] provides a set of different PCA implementations, together with tools for cross validation and visualisation of the results. The methods basically allow to perform PCA on incomplete data and thus may also be used for missing value estimation.

When doing PCA one assumes that the data is restricted to a subspace of lower dimensionality, e.g. correlation patterns between jointly regulated genes. PCA aims to extract these structures thereby filtering noise out. If only the most significant loadings (eigenvectors, also referred to as principal components) are used for projection this can be written as:

$$X = 1 \times \bar{x}^T + TP^T + V \tag{1}$$

Where the term $1 \times \bar{x}^T$ represents the original variable averages, X denotes the observations, $T = t_1, t_2, \ldots, t_k$ the latent variables or scores, $P = p_1, p_2, \ldots, p_k$ the transformation matrix (consisting of the most significant eigenvectors of the covariance matrix) and V are the residuals.

Missing values may be estimated by projecting the scores back into the original space using $\hat{X} = 1 \times \bar{x}^T + TP^T$. Optimally, this produces an estimate of the missing data based on the underlying correlation structure, thereby ignoring noise. This will only produce reasonable results if the residuals V are sufficiently small, implying that most of the important information is captured by the first k components.

In order to calculate the transformation matrix P one needs to determine the covariance matrix between variables or alternatively calculate P directly via SVD. In both cases, this can only be done on complete matrices. However, an approximation may be obtained by use of different regression methods. The PCA methods provided in this package implement algorithms to accurately estimate the PCA solution on incomplete data.

Although the focus of this package is clearly to provide a collection of PCA methods we also provide a cluster based method for missing value imputation. This allows to better rate and compare the results.

1 Algorithms

All methods return a common class called `pcaRes` as a container for the results. This guarantees maximum flexibility for the user. A wrapper function called `pca()` is provided that receives the desired type of pca as a string.

svdPca

This is a wrapper function for $R's$ standard `prcomp` function. It delivers the results as a `pcaRes` object for compatibility with the rest of the package.

svdImpute

This implements the SVDimpute algorithm as proposed by Troyanskaya et al [3]. The idea behind the algorithm is to estimate the missing values as a linear combination of the k most significant eigengenes[1]. The algorithm works iteratively until the change in the estimated solution falls below a certain threshold. Each step the eigengenes of the current estimate are calculated and used to determine a new estimate.

An optimal linear combination is found by regressing an incomplete variable against the k most significant eigengenes. If the value at position j is missing, the j^{th} value of the eigengenes is not used when determining the regression coefficients.
SVDimpute seems to be tolerant to relatively high amount of missing data ($>$ 10%).

Probabilistic PCA (ppca)

Probabilistic PCA combines an EM approach for PCA with a probabilistic model. The EM approach is based on the assumption that the latent variables as well as the noise are normal distributed.

In standard PCA data which is far from the training set but close to the principal subspace may have the same reconstruction error, see Figure 1 for explanation.

PPCA defines a likelihood function such that the likelihood for data far from the training set is much lower, even if they are close to the principal subspace. This allows to improve the estimation accuracy.
PPCA is tolerant to amounts of missing values between 10% to 15%. If more data is missing the algorithm is likely not to converge to a reasonable solution.

The method was implemented after the draft "`EM Algorithms for PCA and Sensible PCA`" written by Sam Roweis and after the Matlab `ppca` script implemented by *Jakob Verbeek*[2].

Please check also the PPCA help file.

Bayesian PCA (bpca)

Similar to probabilistic PCA, Bayesian PCA uses an EM approach together with a Bayesian model to calculate the likelihood for a reconstructed value.

[1] The term "eigengenes" denotes the loadings when PCA was applied considering variables (here the genes) as observations.

[2] `http://lear.inrialpes.fr/~verbeek/`

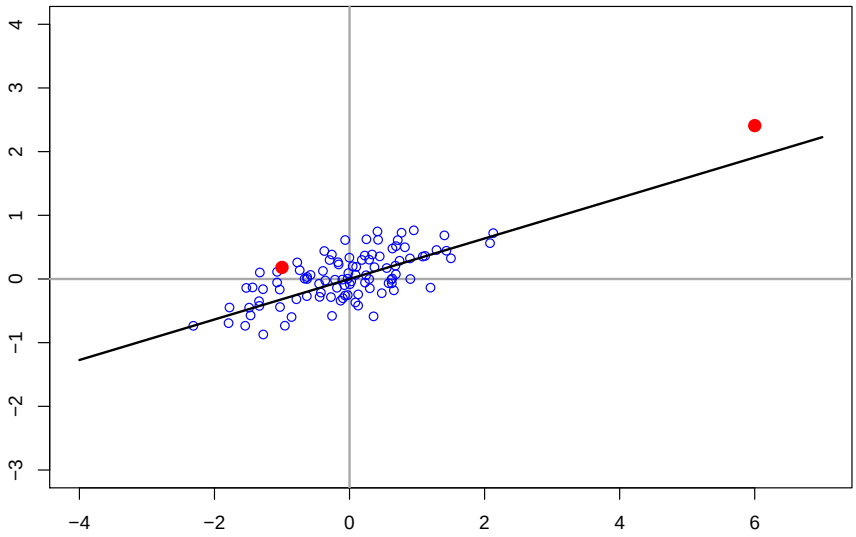

Figure 1: Normal distributed data with the first loading plotted in black. The two red points have the same reconstruction error because PCA does not define a density model. Thus the only measure of how well new data fits the model is the distance from the principal subspace. Data points far from the bulk of data but still close to the principal subspace will have a low reconstruction error.

The algorithm seems to be tolerant to relatively high amounts of missing data ($> 10\%$). Scores and loadings obtained with Bayesian PCA slightly differ from those obtained with conventional PCA. This is because BPCA was developed especially for missing value estimation and is based on a variational Bayesian framework (VBF), with automatic relevance determination (ARD). In BPCA, ARD leads to a different scaling of the scores, loadings and eigenvalues when compared to standard PCA or PPCA. The algorithm does not force orthogonality between loadings. However, the authors of the BPCA paper found that including an orthogonality criterion made the predictions worse. They also state that the difference between "real" and predicted Eigenvalues becomes larger when the number of observation is smaller, because it reflects the lack of information to accurately determine true loadings from the limited and noisy data. As a result, weights of factors to predict missing values are not the same as with conventional PCA, but the missing value estimation is improved.

BPCA was proposed by Oba et al [5]. The method available in this package is a port of the `bpca` Matlab script also provided by the authors[3].

Inverse non-linear PCA (NLPCA)

NLPCA [2] is especially suitable for data from experiments where the studied response is non-linear. Examples of such experiments are ubiquitous in biology – enzyme kinetics are inherently non-linear as are gene expression responses influenced by the cell cycle or diurnal oscillations. NLPCA is based on training

[3] `http://hawaii.aist-nara.ac.jp/%7Eshige-o/tools/`

an auto-associative neural network composed of a component layer which serves as the "bottle-neck", a hidden non-linear layer and an output layer corresponding to the reconstructed data. The loadings can be seen as hidden in the network. Missing values in the training data are simply ignored when calculating the error during back-propagation. Thus NLPCA can be used to impute missing values in the same way as for conventional PCA. The only difference is that the loadings P are now represented by a neural network.
A shortcoming of the current implementation is that there is no reasonable stop criterion. The quality of the estimated solution depends on the number of iterations. This should in most cases be somewhat between 500 and 1500. We recommend to use `kEstimate` or `kEstimateFast` to determine this parameter.

Nipals PCA

Nipals (Nonlinear Estimation by Iterative Partial Least Squares) [6] is an algorithm at the root of PLS regression which can execute PCA with missing values by simply leaving those out from the appropriate inner products. It is tolerant to small amounts (generally not more than 5%) of missing data.

1.1 Local least squares (LLS) imputation

The package provides an algorithm called `llsImpute` for missing value estimation based on a linear combination of the k nearest neighbours of an incomplete variable (in Microarray experiments normally a gene). The distance between variables is defined as the absolute value of the Pearson, Spearman or Kendall correlation coefficient. The optimal linear combination is found by solving a local least squares problem as described in [7]. In tests performed in the cited paper the llsImpute algorithm is able to outperform knnImpute[3] and competes well with BPCA.

In the current implementation two slightly different ways for missing value estimation are provided. The first one is to restrict the neighbour searching to the subset of complete variables. This is preferable when the number of incomplete variables is relatively small.

The second way is to consider all variables as candidates. Here, missing values are initially replaced by the columns wise mean. The method then iterates, using the current estimate as input for the LLS regression until the change between new and old estimate falls below a certain threshold (0.001).

2 Getting started

Installing the package. To install the package first download the appropriate file for your platform from the Bioconductor website (`http://www.bioconductor.org/`). For Windows, start `R` and select the `Packages` menu, then `Install package from local zip file`. Find and highlight the location of the zip file and click on `open`.

For Linux/Unix, use the usual command `R CMD INSTALL` or set the option `CRAN` to your nearest mirror site and use the command `install.packages` from within an `R` session.

Loading the package: To load the `pcaMethods` package in your `R` session, type `library(pcaMethods)`.

Help files: Detailed information on `pcaMethods` package functions can be obtained from the help files. For example, to get a description of `bpca` type `help("bpca")`.

Sample data: Two sample data sets are coming with the package. `metaboliteDataComplete` contains a complete subset from a larger metabolite data set. `metaboliteData` is the same data set but with 10 % values removed from an equal distribution.

3 Some examples

To load the package and the two sample data sets type:

```
> library(pcaMethods)
> data(metaboliteData)
> data(metaboliteDataComplete)
```

Now centre the data

```
> md  <- prep(metaboliteData, scale="none", center=TRUE)
> mdC  <- prep(metaboliteDataComplete, scale="none", center=TRUE)
```

Run SVD pca, PPCA, BPCA, SVDimpute and nipalsPCA on the data, using the `pca()` wrapper function. The result is always a `pcaRes` object.

```
> resPCA  <- pca(mdC, method="svd", center=FALSE, nPcs=5)
> resPPCA  <- pca(md, method="ppca", center=FALSE, nPcs=5)
> resBPCA  <- pca(md, method="bpca", center=FALSE, nPcs=5)
> resSVDI  <- pca(md, method="svdImpute", center=FALSE, nPcs=5)
> resNipals  <- pca(md, method="nipals", center=FALSE, nPcs=5)
> resNLPCA <- pca(md, method="nlpca", center=FALSE, nPcs=5, maxSteps=300)
```

Figure 2 shows a plot of the eigenvalue structure (`sDev(pcaRes)`). If most of the variance is captured with few loadings PCA is likely to produce good missing value estimation results. For the sample data all methods show similar eigenvalues. One can also see that most of the variance is already captured by the first loading, thus estimation is likely to work fine on this data. For BPCA, the eigenvalues are scaled differently for reasons discussed above, see Figure 3. The order of the loadings remains the same.

To get an impression of the correctness of the estimation it is a good idea to plot the scores / loadings obtained with classical PCA and one of the probabilistic methods against each other. This of course requires a complete data set from which data is randomly removed. Figure 3 shows this for BPCA on the sample data.

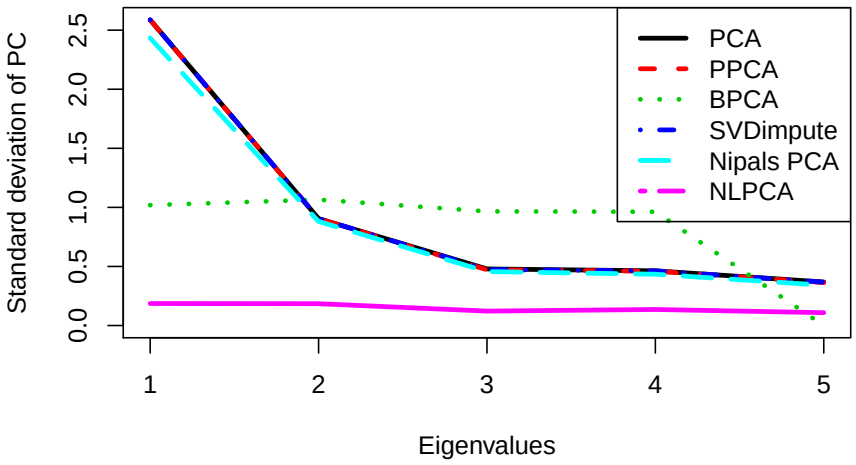

Figure 2: Eigenvalue structure as obtained with different methods

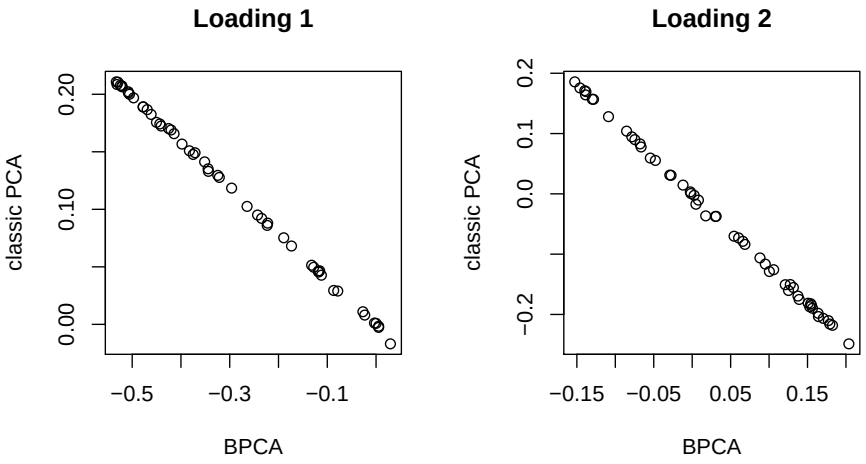

Figure 3: Loading 1 and 2 calculated with BPCA plotted against those calculated with standard PCA.

4 Cross validation

`Q2` is the goodness measure used for internal cross validation. This allows to estimate the level of structure in a data set and to optimise the choice of number of loadings. Cross validation is performed by removing random elements of the data matrix, then estimating these using the PCA algorithm of choice and then calculating Q^2 accordingly. At the moment, cross-validation can only be performed with algorithms that allow missing values (i.e. not SVD). Missing value independent cross-validation is scheduled for implementation in later versions. Q^2 is defined as following for the mean centered data (and possibly scaled) matrix X.

$$\text{SSX} = \sum (x_{ij})^2$$

$$\text{PRESS} = \sum (x_{ij} - \hat{x}_{ij})^2$$

$$Q^2 = 1 - \text{PRESS}/\text{SSX}$$

The maximum value for Q^2 is thus 1 which means that all variance in X is represented in the predictions; $X = \hat{X}$.

```
> q2SVDI <- Q2(resSVDI, mdC, fold=10)
> q2PPCA <- Q2(resPPCA, mdC, fold=10)
```

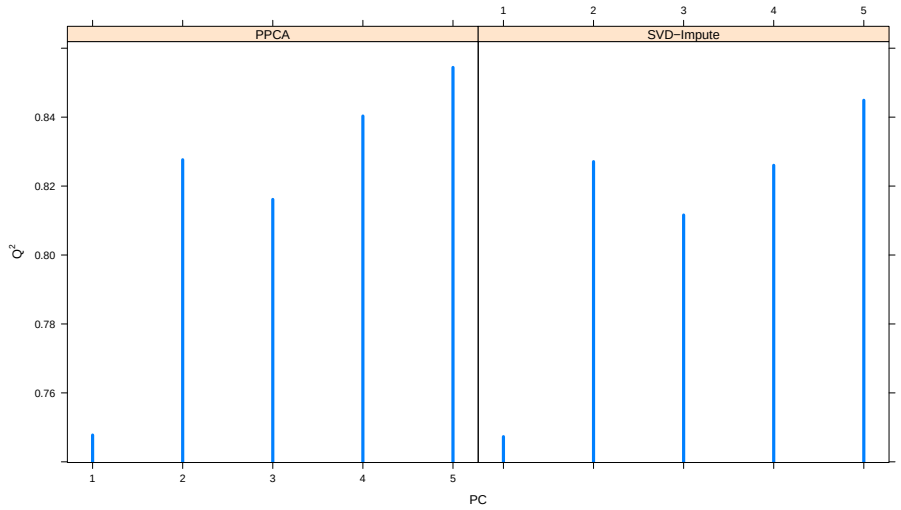

Figure 4: Boxplot of the `Q2` results for BPCA, Nipals PCA, SVDimpute and PPCA. PPCA and SVDimpute both deliver better results than BPCA and Nipals in this example.

The second method called `kEstimate` uses cross validation to estimate the optimal number of loadings for missing value estimation. The `NRMSEP` (normalised root mean square error of prediction) [4] or Q2 can be used to define the average error of prediction. The NRMSEP normalises the square difference between real and estimated values for a certain variable by the variance within

this variable. The idea behind this normalisation is that the error of prediction will automatically be higher if the variance is higher. The `NRMSEP` for mean imputation is $\sqrt{\frac{nObs}{nObs-1}}$ when cross validation is used, where $nObs$ is the number of observations. The exact definition is:

$$NRMSEP_k = \sqrt{\frac{1}{g}\sum_{j \in G} \frac{\sum_{i \in O_j}(x_{ij} - \hat{x}_{ijk})^2}{o_j s^2_{x_j}}} \tag{2}$$

where $s^2_{x_j} = \sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2/(n-1)$, this is the variance within a certain variable. Further, G denotes the set of incomplete variables, g is the number of incomplete varialbes. O_j is the set of missing observations in variable j and o_j is the number of missing observations in variable j. $\hat{x}_{ijk}$ stands for the estimate of value i of variable j using k loadings. See Figure 5 for an example. The NRMSEP should be the error measure of choice. But if the number of observations is small, the variance within a certain variable may become and unstable criterion. If so or if variance scaling was applied we recommend to use Q2 instead.

```
> errEsti <- kEstimate(md, method = "ppca", evalPcs=1:5, nruncv=1, em="nrmsep")
```

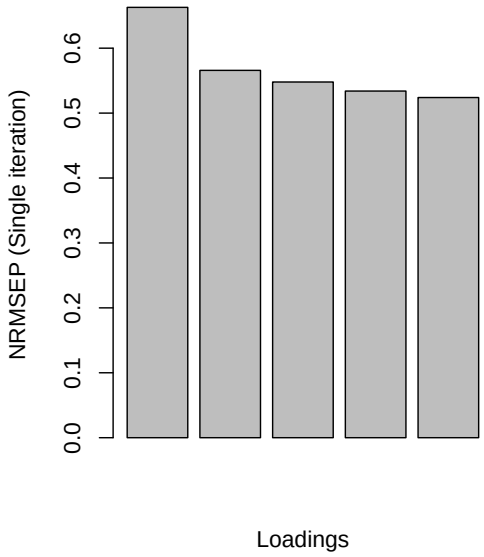

Figure 5: Boxplot showing the `NRMSEP` versus the number of loadings. In this example only 1 iteration of the whole cross validation were performed. It is normally advisable to do more than just one iteration.

`kEstimate` also provides information about the estimation error for individual variables. The Q^2 distance or the NRMSEP are calculated separately for each variable. See the manpage for `kEstimate` and `kEstimateFast` for details. Plotting the variable - wise results gives information about for which variables missing value estimation makes sense, and for which no imputation or mean imputation is preferable, see Figure 6. If you are not interested in variable - wise information we recommend to use the faster `kEstimateFast` instead.

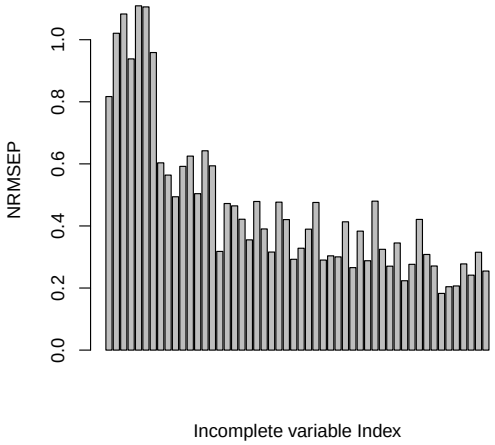

Figure 6: Boxplot showing the `NRMSEP` for all incomplete variables in the data set. For the first 7 variables missing value imputation does not seem to make too much sense.

5 Visualization of the results

Some methods for display of scores and loadings are also provided. The function `slplot()` aims to be a simple way to quickly visualize scores and loadings in an intuitive way, see Figure 7. Barplots are provided when plotting only one PC and colors can be specified differently for the scores and loadings plots. For a more specific scatter plot it is however recommended to access scores and loadings slots and define own plot functions.

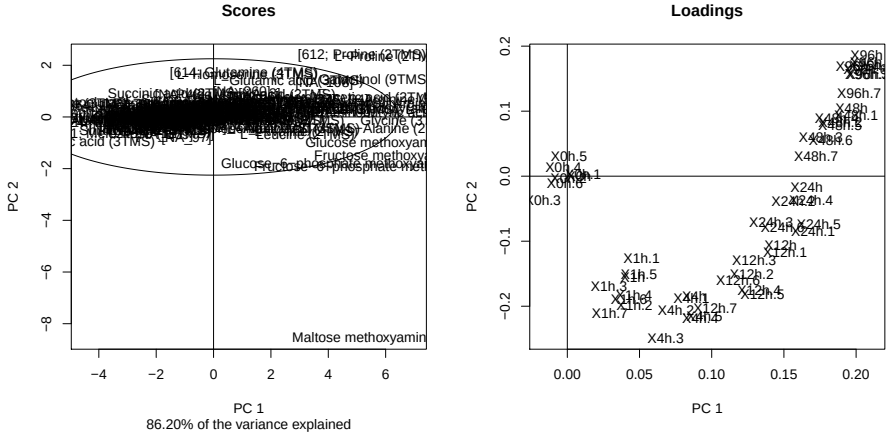

Figure 7: `slplot` for scores and loadings obtained with classical SVD based PCA.

Another method called `plotPcs()` allows to visualise many PCs plotted against each other, see Figure 8.

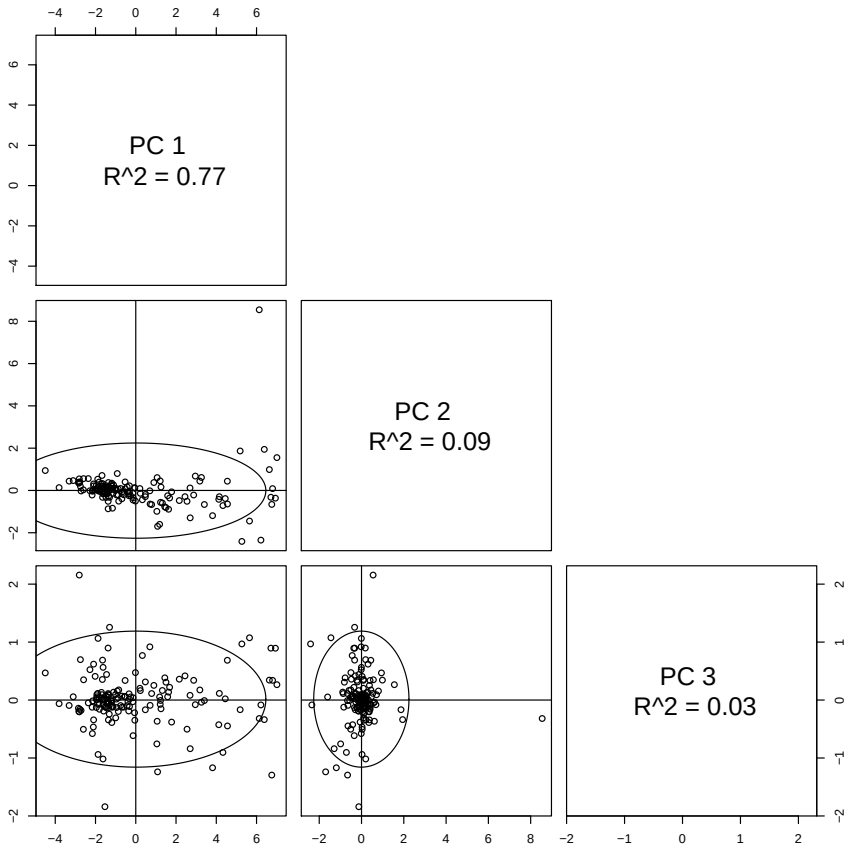

Figure 8: A plot of score 1:3 for PPCA created with `plotPcs()`

References

[1] Stacklies W., Redestig H., Scholz M., and Walther D., and Selbig J. *pcaMethods – a Bioconductor package providing PCA methods for incomplete data* Bioinformatics. 2007, 23, 1164-1167. *Non-linear PCA: a missing data approach.* Bioinformatics. 2005, 21, 3887-3895.

[2] Scholz, M. , Kaplan, F., Guy, C.L., Kopka, J. and Selbig, J. *Non-linear pca: a missing data approach.* Bioinformatics. 2005, 21, 3887-3895.

[3] Troyanskaya O. and Cantor M. and Sherlock G. and Brown P. and Hastie T. and Tibshirani R. and Botstein D. and Altman RB. *Missing value estimation methods for DNA microarrays.* Bioinformatics. 2001 Jun;17(6):520-525.

[4] Feten G. and Almoy T. and Aastveit A.H. *Prediction of Missing Values in Microarray and Use of Mixed Models to Evaluate the Predictors.*, Stat. Appl. Genet. Mol. Biol. 2005;4(1):Article 10

[5] Oba S. and Sato MA. and Takemasa I. and Monden M. and Matsubara K. and Ishii S. *A Bayesian missing value estimation method for gene expression profile data.* Bioinformatics. 2003 Nov 1;19(16):2088-96.

[6] Wold H. Estimation of principal components and related models by iterative least squares. In Multivariate Analysis (Ed. P.R. Krishnaiah), Academic Press, NY, 391-420.

[7] Kim H. and Golub G.H. and Park H. *Missing value estimation for DNA microarray gene expression data: local least squares imputation* Bioinformatics. 2005 21(2) :187-198