# The pcaMethods Package

Wolfram Stacklies and Henning Redestig
Max Planck Institute for Molecular Plant Physiology
Potsdam, Germany
`http://bioinformatics.mpimp-golm.mpg.de/`

September 18, 2006

## Overview

The `pcaMethods` package 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 eigenvectors are used for projection this can be written as:

$$y = Cx + v \tag{1}$$

Where $y$ denotes the observations, $C$ the transformation matrix (consisting of the eigenvectors of the covariance matrix), $x$ are the latent variables or scores, and $v$ is the noise.

Missing values may be estimated by projecting the scores back into the original space.

$$y_{esti} = xC^T \tag{2}$$

Optimally, this produces an estimate of the missing data based on the underlying correlation structure, thereby ignoring noise.

In order to calculate the transformation matrix $C$ one needs to determine the covariance matrix between variables or alternatively calculate $C$ 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.

## 1 PCA algorithms

All methods use a common class called `pcaRes` to return 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 [1]. 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 the incomplete gene 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. 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].

## 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.
The algorithm seems to be tolerant to relatively high amounts of missing data ($>$ 10%).

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

[1]The term "Eigengenes" denotes the loadings when PCA was applied considering genes as observations.

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

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

### Nipals PCA

Nipals (Nonlinear Estimation by Iterative Partial Least Squares) [4] is an algorithm at the root of PLS regression. It is tolerant to small amounts (generally not more than 5%) of missing data.

## 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)
```

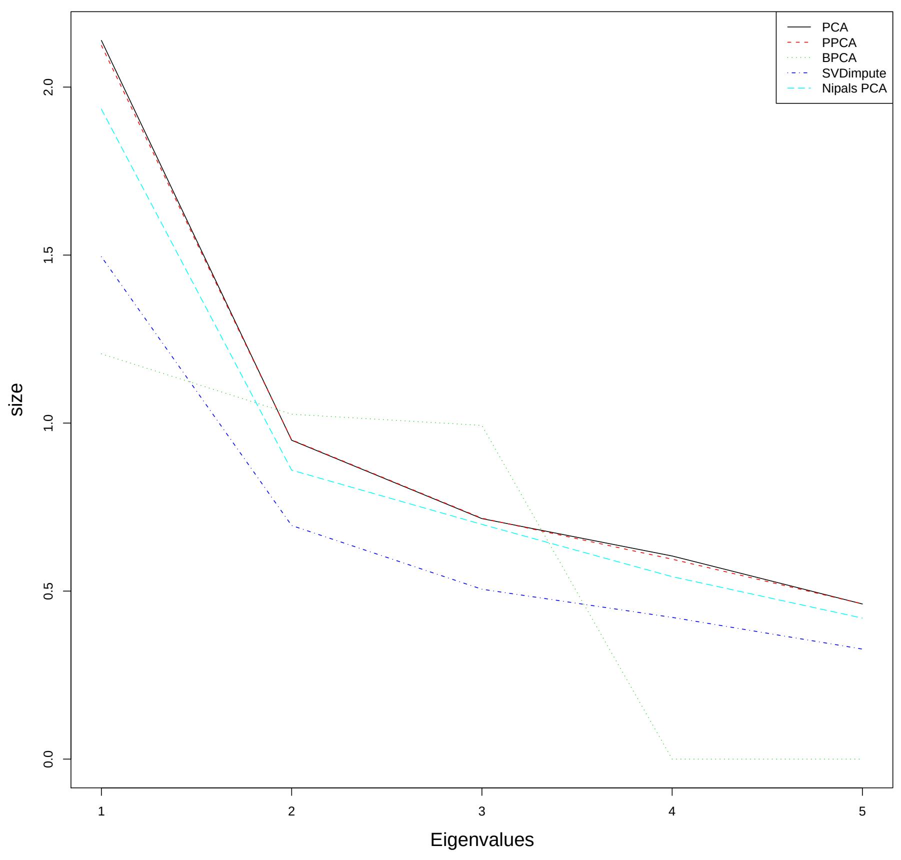

Figure 1: Eigenvalue structure as obtained with different methods

Figure 1 shows a plot of the eigenvalue structure (`pcaRes@sDev`). If most of the variance is captured with few eigenvectors PCA is likely to produce good missing value estimation results. For the sample data all methods show similar similar eigenvalues. One can also see that most of the variance is already captured by the first eigenvector, thus estimation is likely to work fine on this data. For BPCA, the eigenvalues are scaled differently because the scaling for scores / loadings differs from the other methods. But this is always a linear factor, see Figure 2. The order of the eigenvectors remains the same. I will try to address this issue in future releases.

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 2 shows this for BPCA on the sample data.

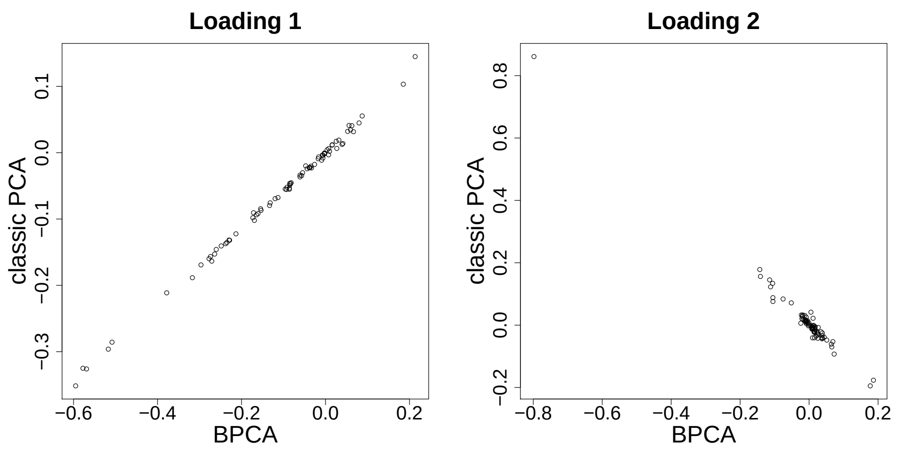

Figure 2: Loading 1 and 2 calculated with BPCA plotted against those calculated with standard PCA.

## 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 principal components. Cross validation is performed by either 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 $X$.

$$\text{SSX} = \sum (x_{i,k})^2$$

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

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

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

```
> q2BPCA <- Q2(resBPCA, mdC, nPcs = 5, nruncv = 1, fold = 10)
> q2Nipals <- Q2(resNipals, mdC, nPcs = 5, nruncv = 1, fold = 10)
> q2SVDI <- Q2(resSVDI, mdC, nPcs = 5, nruncv = 1, fold = 10)
> q2PPCA <- Q2(resPPCA, mdC, nPcs = 5, nruncv = 1, fold = 10)
```

The second method called kEstimate uses cross validation to estimate the optimal number of components for missing value estimation. The NRMSEP (normalised root mean square error) [2] is used to define the average error of prediction. This error normalises the square difference between real and estimated values for a certain gene by the variance of this gene. The idea behind this normalisation is that the error of prediction will automatically be higher if the

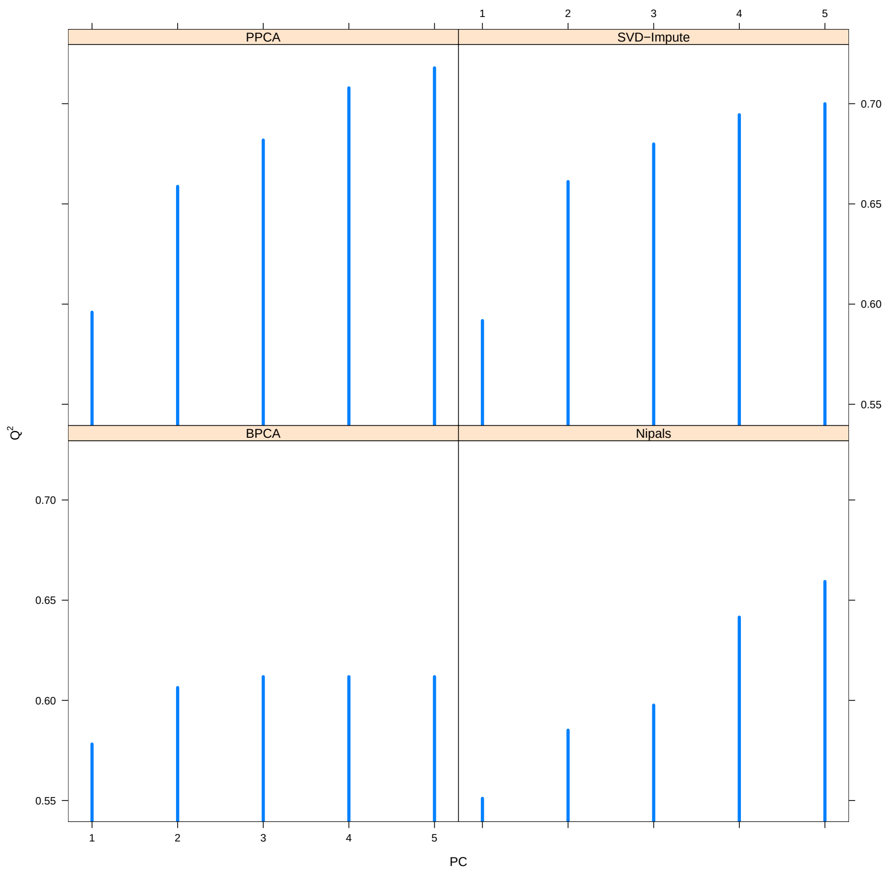

Figure 3: Boxplot of the `Q2` results for BPCA, Nipals PCA, SVDimpute and PPCA PPCA and SVDimpute both deliver better results than BPCA and Nipals in this example.

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:

$$NRMSEP_k = \sqrt{\frac{1}{p} \sum_{j \in P} \frac{\sum_{i \in T_j} (y_{ij} - ye_{ijk})^2}{t_j s_{y_j}^2}} \tag{3}$$

where $s_{y_j}^2 = \sum_{i=1}^{n} (y_{ij} - \overline{y}_j)^2/(n-1)$, this is the variance within a certain gene. Further, $P$ denotes the set of incomplete genes, $p$ is the number of incomplete genes. $T_j$ is the set of missing observations in gene $j$ and $t_j$ is the number of missing observations in gene $j$. $ye_{ijk}$ stands for the estimate of value $i$ of gene $j$ using $k$ components. See Figure 4 for an example.

```
> errEsti <- kEstimate(md, method = "ppca", maxPcs = 5, nruncv = 1)
```

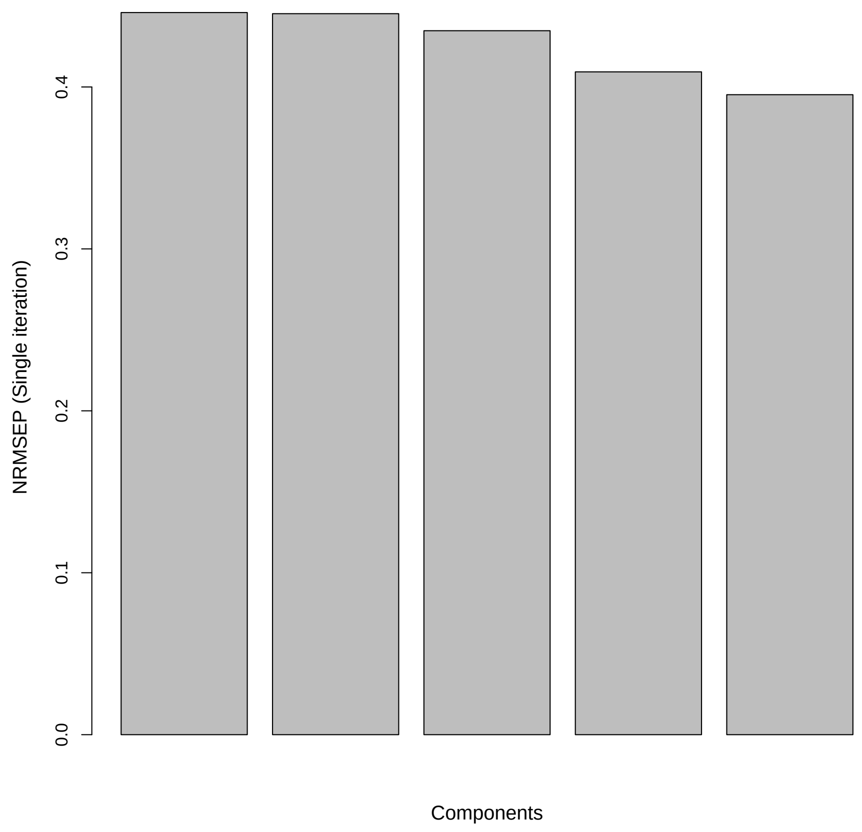

Figure 4: Boxplot showing the `NRMSEP` versus the number of components. In this example only one iteration was performed, it is recommendable to allow more iterations to obtain better estimates of NRMSEP.

## 5 Visualisation of the results

Some methods for display of scores and loadings are also provided. `slplot()` is a convenient way to present a PCA result for two components. An example is shown in Figure 5.
Another method called `plotPcs()` allows to visualise many PCs plotted against each other, see Figure 6.

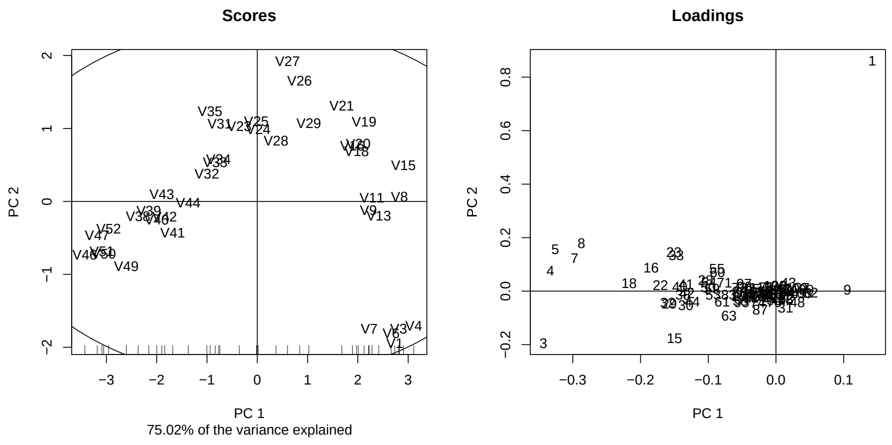

Figure 5: `slplot` for scores and loadings obtained with SVDimpute.

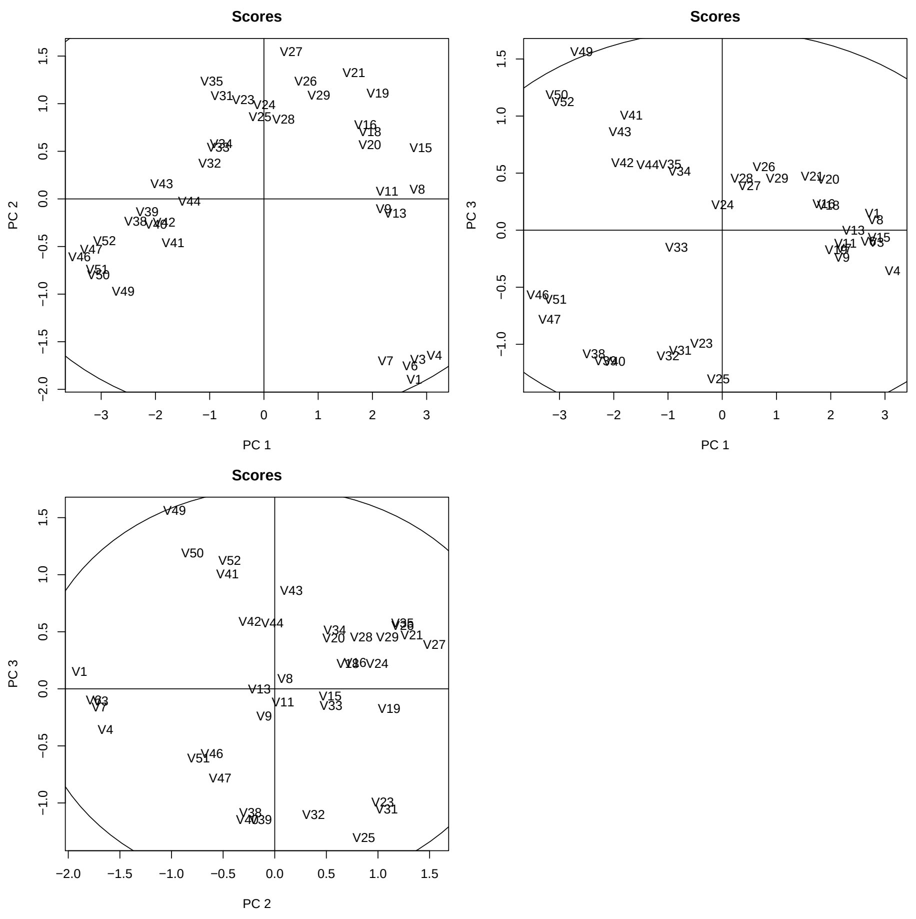
Figure 6: A plot of score 1:3 for PPCA created with `plotPcs()`

## References

[1] 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):520-5.

[2] G. Feten and T. Almoy and A.H. Aastveit *Prediction of Missing Values in Microarray and Use of Mixed Models to Evaluate the Predictors.*, Stat. Appl. Genet. Mol. Biol. 2005;4(1):Article10

[3] 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.

[4] 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.