

A fast robust method for principal components with applications to chemometrics

Reference:

Hubert, M., Rousseeuw, P.J., Verboven, S.
(2002), *Chemometrics and Intelligent
Laboratory Systems*, 60, 101-111.

Principal Component Analysis (PCA)

Given:

n observations $\mathbf{x}_1, \dots, \mathbf{x}_n$

p variables X_1, \dots, X_p

$\Rightarrow n \times p$ data matrix $X_{n,p} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^t$

Problems:

- Computation time of multivariate methods increases a lot when p is large.
- Often many variables are correlated (e.g. spectra in chemometrics).

Goal of PCA:

to construct a smaller set of uncorrelated

variables Y_1, \dots, Y_k ($k < p$)

with minimal loss of information.

Motivation of our work

- PCA is a frequently used multivariate technique with many applications, e.g. in chemometrics, engineering, image processing,...
- We need robust methods, since real data sets often contain outliers.
- The algorithm of Croux and Ruiz-Gazen (1996) works well, but in high dimensions it needs much computation time and becomes less stable numerically.

Classical PCA

1. Center the data by means of the classical mean $\hat{\boldsymbol{\mu}}^C = (\hat{\mu}_1^C, \dots, \hat{\mu}_p^C)^t$.

2. Define

$$Y_l = a_{l1}(X_1 - \hat{\mu}_1^C) + a_{l2}(X_2 - \hat{\mu}_2^C) + \dots + a_{lp}(X_p - \hat{\mu}_p^C)$$

such that for all $l = 1, \dots, k$:

- $\text{Var}(Y_l)$ is maximal, or equivalently

$$SD \left\{ \mathbf{a}_l^t(\mathbf{x}_1 - \hat{\boldsymbol{\mu}}_1^C), \dots, \mathbf{a}_l^t(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_p^C) \right\}$$

is maximal, with SD the classical standard deviation.

- $\|\mathbf{a}_l\| = 1$
- $\mathbf{a}_l \perp \mathbf{a}_m$ for all $l \neq m$

3. Solution: put $\mathbf{a}_l = \mathbf{v}_l$ where $\mathbf{v}_1, \dots, \mathbf{v}_p$ are the eigenvectors of the classical covariance matrix S , with eigenvalues

$$\lambda_1 \geq \dots \geq \lambda_p \geq 0.$$

Classical decomposition: let

$$P_{p,r} = (\mathbf{v}_1, \dots, \mathbf{v}_r) \text{ and}$$

$$T_{n,r} = (X_{n,p} - M^C)P_{p,r}$$

$$\Rightarrow \boxed{X_{n,p} - M^C = T_{n,r}P_{r,p}^t}$$

with $r = \text{rank}(X_{n,p} - M^C)$

$\hat{\boldsymbol{\mu}}^C = \text{classical mean vector}$

$$M^C = \mathbf{1}_n(\hat{\boldsymbol{\mu}}^C)^t.$$

Problems

- Classical covariance matrix S is highly attracted by outlying observations.



Use a robust covariance estimator:
M-estimator (Devlin et al. 1975),
MCD (Rousseeuw 1984),
S-estimator (Croux and Haesbroeck 1999).

- If $p > n$: calculation of these estimators?



Use Projection Pursuit (PP) to robustify
PCA.

Robust PCA by Projection Pursuit (PP)

proposed by Li and Chen (1985)

1. Center the data by means of a robust location estimator $\hat{\boldsymbol{\mu}}^R = (\hat{\mu}_1^R, \dots, \hat{\mu}_p^R)^t$.
2. Define

$$Y_l = a_{l1}(X_1 - \hat{\mu}_1^R) + a_{l2}(X_2 - \hat{\mu}_2^R) + \dots + a_{lp}(X_p - \hat{\mu}_p^R)$$

such that for all $l = 1, \dots, k$:

- $s^R \{ \mathbf{a}_l^t(\mathbf{x}_1 - \hat{\boldsymbol{\mu}}_1^R), \dots, \mathbf{a}_l^t(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_p^R) \}$ is maximal, with s^R a robust scale estimator.
- $\|\mathbf{a}_l\| = 1$
- $\mathbf{a}_l \perp \mathbf{a}_m$ for all $l \neq m$

C-R Algorithm

Implementation of the general PP-idea by Croux and Ruiz-Gazen (1996, 2000):

1. Take $\hat{\boldsymbol{\mu}}^R$ the L^1 -median, defined as

$$\hat{\boldsymbol{\mu}}^R = \operatorname{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^p} \sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\theta}\|$$

\Rightarrow consider the centered observations

$$(\mathbf{x}_1^{(1)}, \dots, \mathbf{x}_n^{(1)})^t = X_{n,p} - M^R$$

with $\mathbf{x}_i^{(1)} = \mathbf{x}_i - \hat{\boldsymbol{\mu}}^R$

$M^R = \mathbf{1}_n (\hat{\boldsymbol{\mu}}^R)^t$ the robust centering matrix.

2. Choose a scale estimator with high breakdown value and good statistical efficiency: take $s^R = Q_n$ defined as

$$Q_n(z_1, \dots, z_n) = 2.2219 c_n \{ |z_i - z_j|; i < j \}_{(k)}$$

(Rousseeuw and Croux 1993)

with $k = \binom{h}{2} \approx \binom{n}{2} / 4$ for $h = \lfloor \frac{n}{2} \rfloor + 1$,

and c_n a finite-sample correction factor.

3. ‘Eigenvectors’ are constructed one after the other. We maximize the spread over all directions that pass through the origin and a (projected) data point.

(a) The first ‘eigenvector’ is

$$\mathbf{v}_1 = \operatorname{argmax}_{\mathbf{a} \in A_1} Q_n(\mathbf{a}^t \mathbf{x}_1^{(1)}, \mathbf{a}^t \mathbf{x}_2^{(1)}, \dots, \mathbf{a}^t \mathbf{x}_n^{(1)})$$

with $A_1 = \left\{ \mathbf{x}_i^{(1)} / \|\mathbf{x}_i^{(1)}\|; i = 1, \dots, n \right\}$.

(b) To guarantee orthogonality, we project the data points on the orthogonal complement of \mathbf{v}_1 :

$$\mathbf{x}_i^{(2)} = (I_{p,p} - \mathbf{v}_1 \mathbf{v}_1^t) \mathbf{x}_i^{(1)}$$

(c) Repeat steps (a) and (b) until $k \leq r = \operatorname{rank}(X_{n,p} - M^R) \leq \min(n - 1, p)$ eigenvectors are found.

Typical step: suppose $\mathbf{v}_1, \dots, \mathbf{v}_l$ have been constructed, then put

$$\begin{aligned}\mathbf{x}_i^{(l+1)} &= (I_{p,p} - \sum_{j=1}^l \mathbf{v}_j \mathbf{v}_j^t) \mathbf{x}_i^{(1)} \\ &= \mathbf{x}_i^{(l)} - \mathbf{v}_l \mathbf{v}_l^t \mathbf{x}_i^{(1)}\end{aligned}$$

and compute the next eigenvector as

$$\mathbf{v}_{l+1} = \operatorname{argmax}_{\mathbf{a} \in A_{l+1}} Q_n(\mathbf{a}^t \mathbf{x}_1^{(l+1)}, \dots, \mathbf{a}^t \mathbf{x}_n^{(l+1)})$$

with $A_{l+1} = \left\{ \mathbf{x}_i^{(l+1)} / \|\mathbf{x}_i^{(l+1)}\|; i = 1, \dots, n \right\}$.

Moreover, define the robust scale s_l^R for all components $l = 1, \dots, k$ as

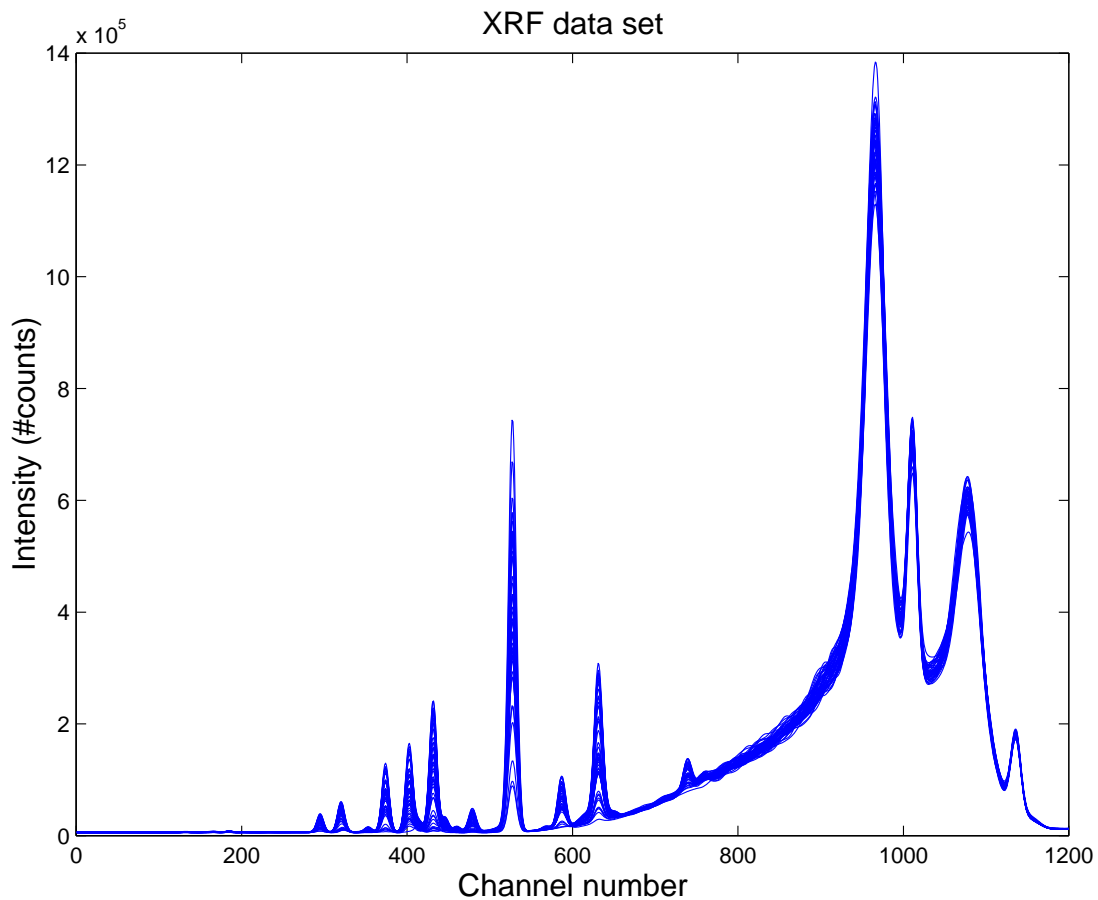
$$\boxed{s_l^R = Q_n(\mathbf{v}_l^t \mathbf{x}_1^{(l)}, \mathbf{v}_l^t \mathbf{x}_2^{(l)}, \dots, \mathbf{v}_l^t \mathbf{x}_n^{(l)})}$$

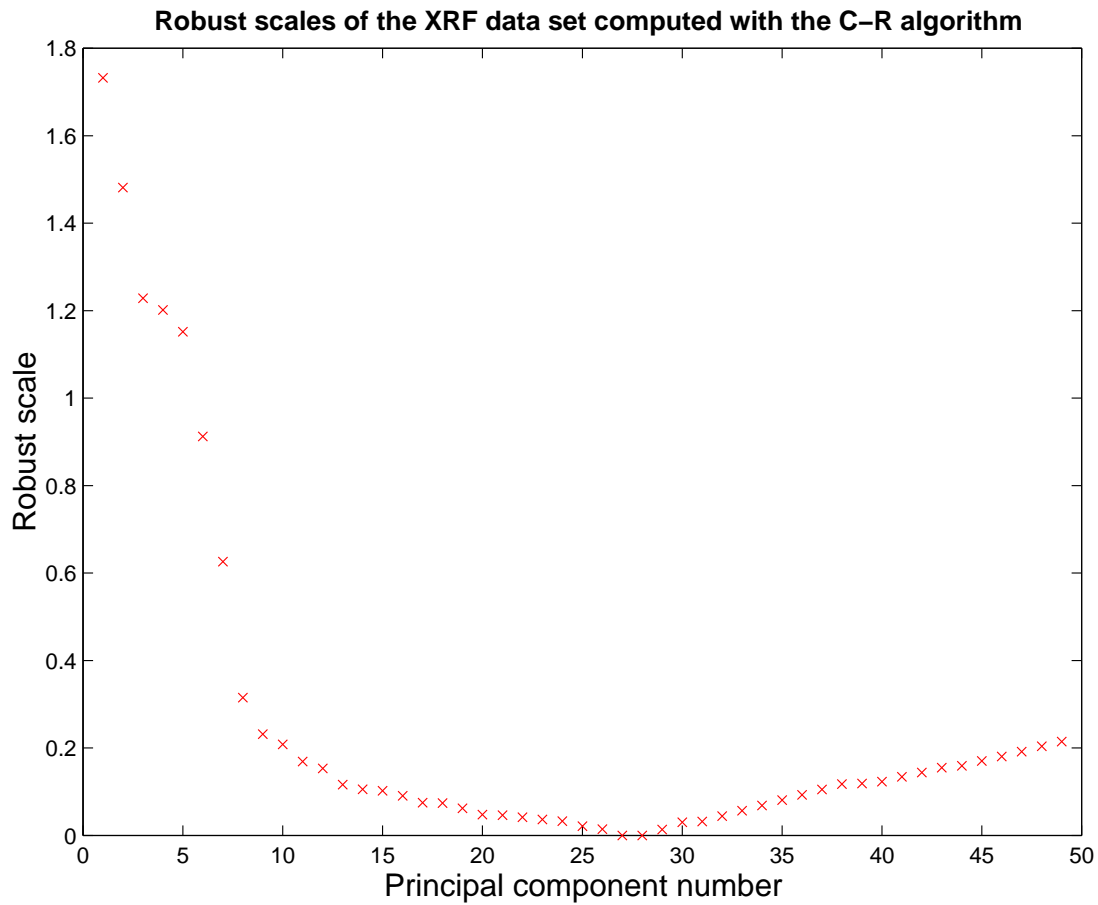
The squared robust scales $(s_l^R)^2$ are called the ‘**eigenvalues**’.

Note that all $\mathbf{x}_i^{(l)} \in \mathbb{R}^p$.

Problem: C-R algorithm may become less numerically stable in high dimensions.

Example: XRF data set with 50 X-Ray Fluorescence spectra of aqueous solutions of metal salts at 1200 channels (Dept. of Chemistry, University of Antwerp). Here $n = 50, p = 1200$.





→ accumulation of round-off errors due to succession of projections in \mathbb{R}^p .

Solution to numerical instability? Reduce the dimension in a slightly different way.

R-step approach

Basic principles as in C-R algorithm.

Improvement:

1. Construct \mathbf{v}_1 as in the C-R algorithm.
2. Transform the data by means of a reflection U_1 such that

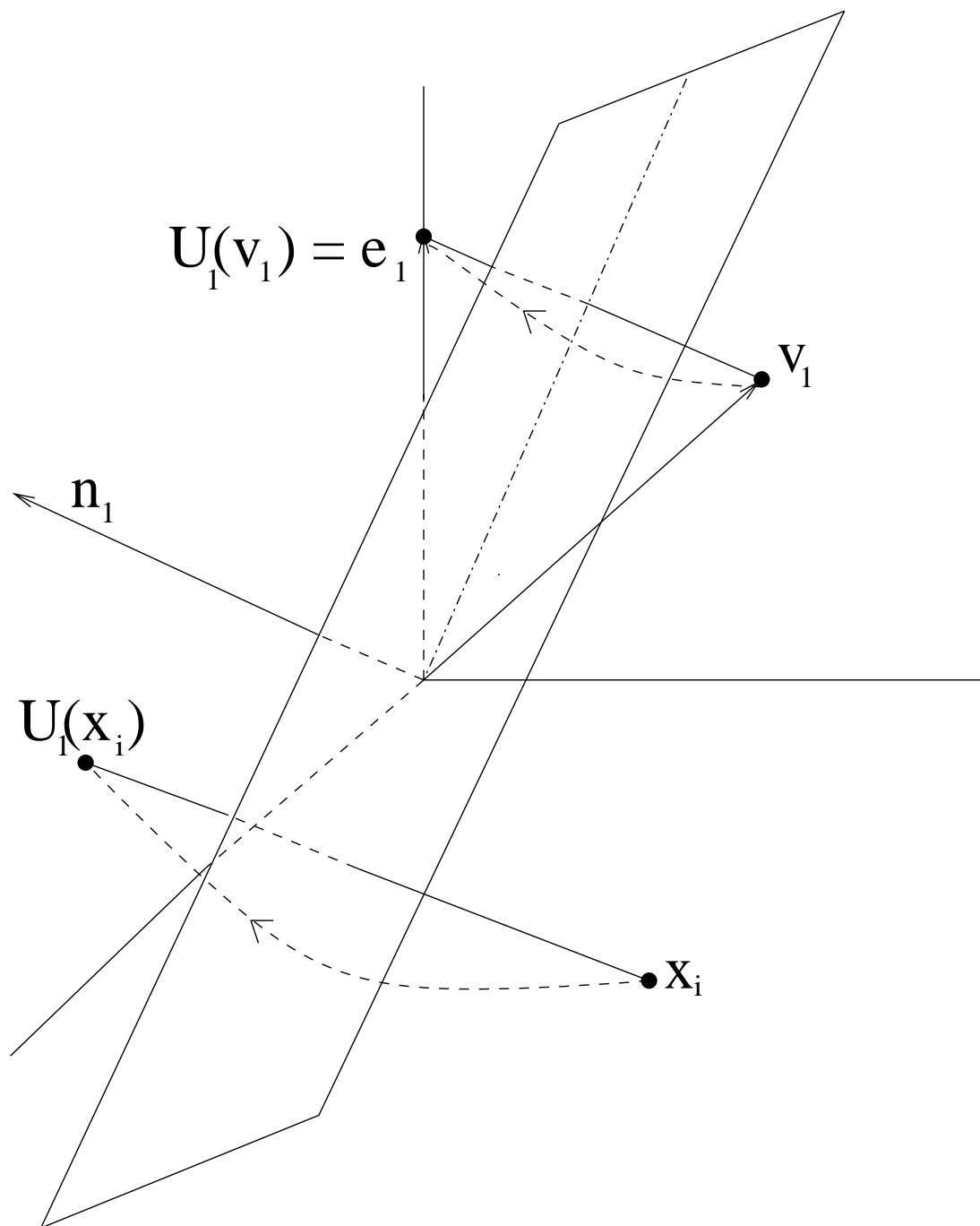
$$U_1(\mathbf{v}_1) = \mathbf{e}_1 = (1, 0, \dots, 0)^t \in \mathbb{R}^p$$

For this we set

$$\begin{aligned} \mathbf{n}_1 &= \frac{\mathbf{e}_1 - \mathbf{v}_1}{\|\mathbf{e}_1 - \mathbf{v}_1\|} \\ \mathbf{x}_i^{(2)} &= U_1(\mathbf{x}_i^{(1)}) \\ &= \mathbf{x}_i^{(1)} - 2\langle \mathbf{x}_i^{(1)}, \mathbf{n}_1 \rangle \mathbf{n}_1 \end{aligned}$$

3. Project data points $\mathbf{x}_i^{(2)}$ onto the orthogonal complement of $U_1(\mathbf{v}_1)$ by omitting their first coordinate:

$$\tilde{\mathbf{x}}_i^{(2)} = \left(x_{i2}^{(2)}, \dots, x_{ip}^{(2)} \right)^t \in \mathbb{R}^{p-1}.$$

R-stepThe reflection U_1

4. Find $\tilde{\mathbf{v}}_2$ in the reduced space \mathbb{R}^{p-1} according to the C-R algorithm:

$$\tilde{\mathbf{v}}_2 = \operatorname{argmax}_{\mathbf{a} \in A_2} Q_n(\mathbf{a}^t \tilde{\mathbf{x}}_1^{(2)}, \dots, \mathbf{a}^t \tilde{\mathbf{x}}_n^{(2)})$$

$$\text{with } A_2 = \left\{ \tilde{\mathbf{x}}_i^{(2)} / \|\tilde{\mathbf{x}}_i^{(2)}\|; i = 1, \dots, n \right\}.$$

5. Repeat Steps 2-4 until k eigenvectors are found. Suppose l eigenvectors have been constructed, then put

$$U_l(\tilde{\mathbf{v}}_l) = \tilde{\mathbf{e}}_l = (1, 0, \dots, 0)^t \in \mathbb{R}^{p-l+1}$$

$$\mathbf{x}_i^{(l+1)} = U_l(\tilde{\mathbf{x}}_i^{(l)}) \in \mathbb{R}^{p-l+1}$$

$$\tilde{\mathbf{x}}_i^{(l+1)} = \left(x_{i,l+1}^{(l+1)}, \dots, x_{i,p}^{(l+1)} \right)^t \in \mathbb{R}^{p-l}$$

and compute the next eigenvector as

$$\tilde{\mathbf{v}}_{l+1} = \operatorname{argmax}_{\mathbf{a} \in A_{l+1}} Q_n(\mathbf{a}^t \tilde{\mathbf{x}}_1^{(l+1)}, \dots, \mathbf{a}^t \tilde{\mathbf{x}}_n^{(l+1)})$$

$$\text{with } A_{l+1} = \left\{ \tilde{\mathbf{x}}_i^{(l+1)} / \|\tilde{\mathbf{x}}_i^{(l+1)}\|; i = 1, \dots, n \right\}$$

6. Finally, backtransform $\tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \dots$ to \mathbb{R}^p .

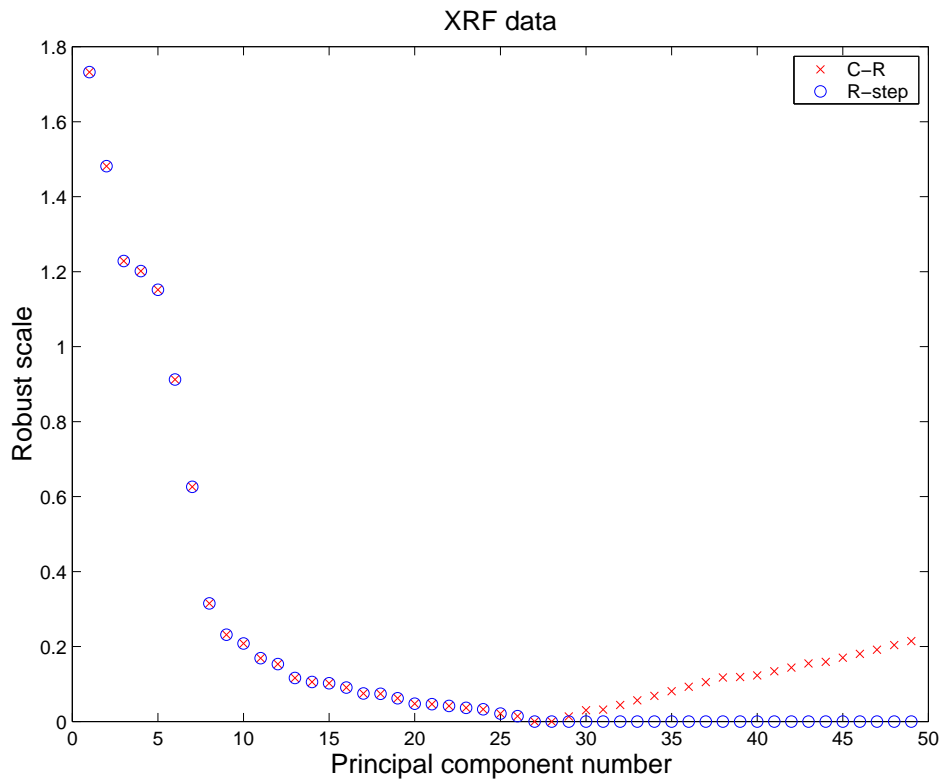
Robust decomposition: let

$$P_{p,k} = (\mathbf{v}_1, \dots, \mathbf{v}_k)$$

$$\Rightarrow \boxed{(X_{n,p} - M^R)P_{p,k} = T_{n,k}}$$

Effect of R-Step

Since all operations are mathematically equivalent we should *in principle* obtain the same solution as with the C-R algorithm, but:



RAPCA algorithm

Faster version of the R-step approach.

(RAPCA = *Reflection-based Algorithm for Principal Components Analysis*.)

Step 1:

Reduce the data space to the affine subspace spanned by the n observations. For this we may use classical PCA without loss of robustness:

$$X_{n,p} - M^C = \tilde{T}_{n,r} \tilde{P}_{r,p}^t \quad (1)$$

with $r = \text{rank}(X_{n,p} - M^C) \leq n - 1$

$\hat{\boldsymbol{\mu}}^C =$ classical mean vector

$$M^C = \mathbf{1}_n (\hat{\boldsymbol{\mu}}^C)^t.$$

Is particularly beneficial when $p \gg n$.

Example: XRF data set with $X_{n,p} = X_{50,1200}$

$$\rightarrow \tilde{T}_{n,r} = \tilde{T}_{50,49}.$$

Step 2:

Applying the R-step approach to the scores matrix $\tilde{T}_{n,r}$ yields

$$(\tilde{T}_{n,r} - M_T^R)\tilde{P}_{r,k} = T_{n,k} \quad (2)$$

with $k \leq r$.

Combining (1) and (2) and using the orthogonal equivariance of the L^1 -median, we obtain

$$\boxed{(X_{n,p} - M^R)P_{p,k} = T_{n,k}}$$

with $P_{p,k} := \tilde{P}_{p,r}\tilde{P}_{r,k}$ still column orthogonal.

Implementation of Step 1:

Singular value decomposition (SVD):

$$X^C := X_{n,p} - M^C = U_{n,r} \Lambda_{r,r} V_{r,p}^t$$

with

$$\begin{aligned} r &= \text{rank}(X^C) \\ U^t U &= V^t V = I_{r,r} \\ \Lambda &= \text{diag}(l_1, \dots, l_r) \end{aligned}$$

↓

$$(X^C)^t X^C = V \Lambda^2 V^t$$

thus $V = (\mathbf{v}_1, \dots, \mathbf{v}_r)$ with \mathbf{v}_i the eigenvectors of $(X^C)^t X^C = (n-1)S$, and

$\Lambda^2 = \text{diag}((n-1)\lambda_1, \dots, (n-1)\lambda_r)$ with λ_i the eigenvalues of S .

We can thus apply SVD on X^C or $(X^C)^t X^C$, and then we set

$$\begin{aligned} \tilde{P}_{p,r} &= V \\ \tilde{T}_{n,r} &= X^C \tilde{P}_{p,r} \end{aligned}$$

But if $p > n$, $(X^C)^t X^C$ has dimension $(p \times p)$ which is much larger than $(n \times n)$, the dimension of $X^C (X^C)^t$.

We then apply SVD on

$$X^C (X^C)^t = U \Lambda^2 U^t$$

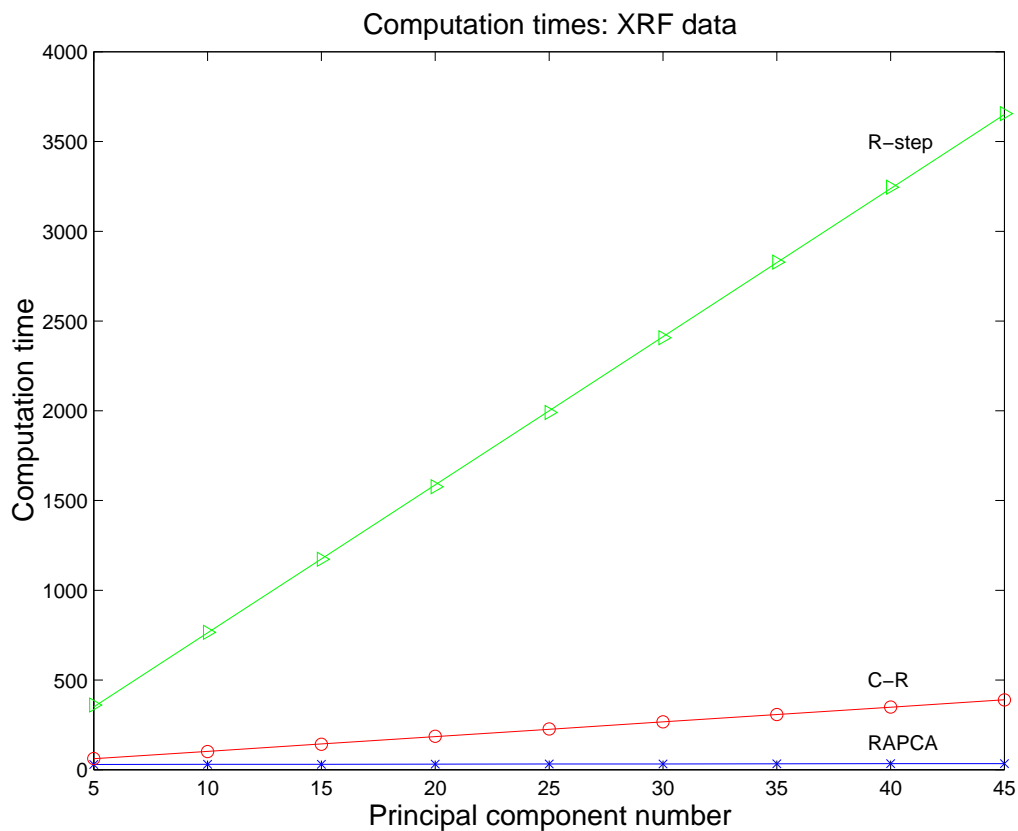
since

$$\boxed{V = (X^C)^t U}$$

and the eigenvalues remain the same.

Conclusions:

- RAPCA yields same results as R-step approach.
- RAPCA is much faster than C-R and R-step approach. Example:

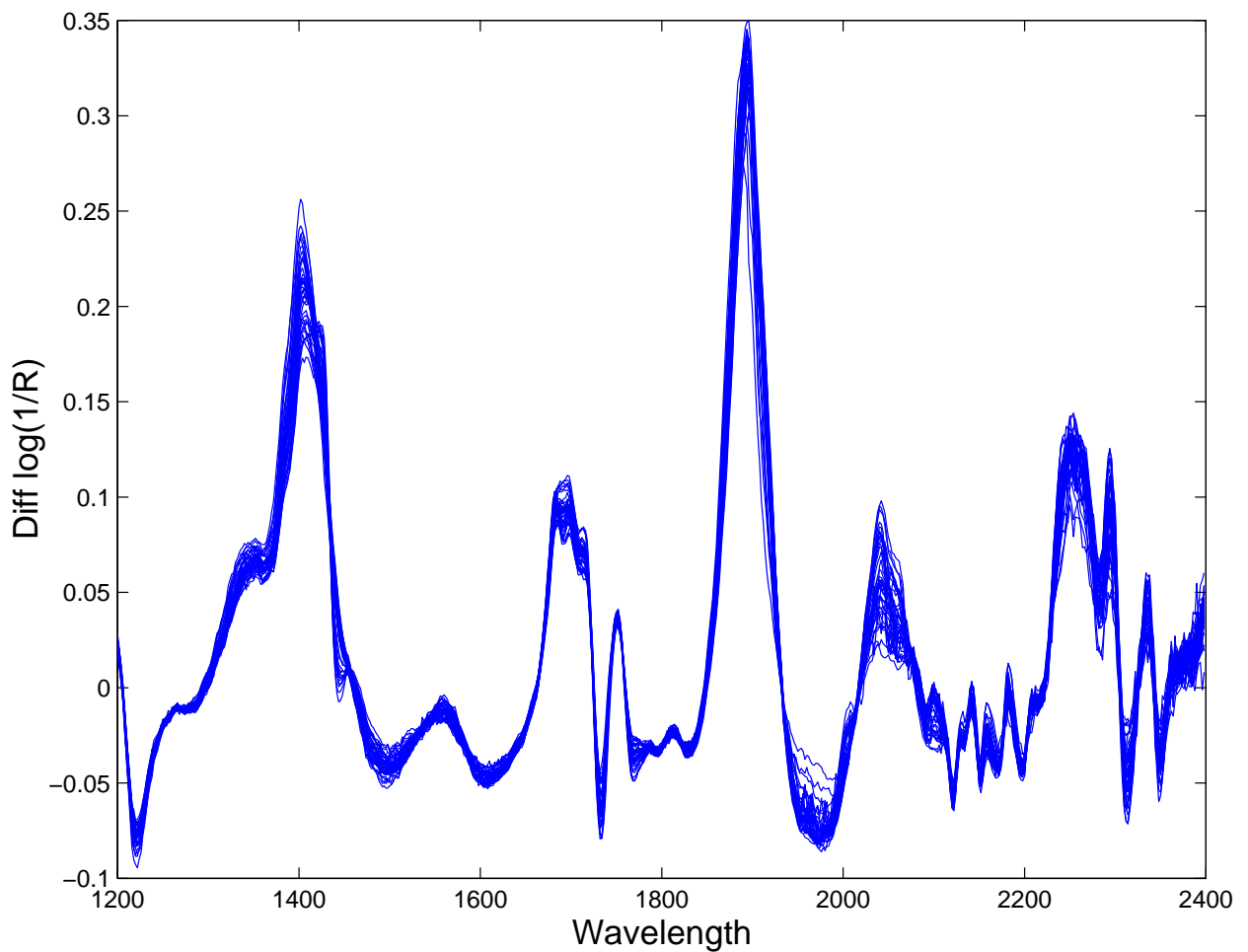


- RAPCA has been implemented in MATLAB and the code can be obtained from the authors.

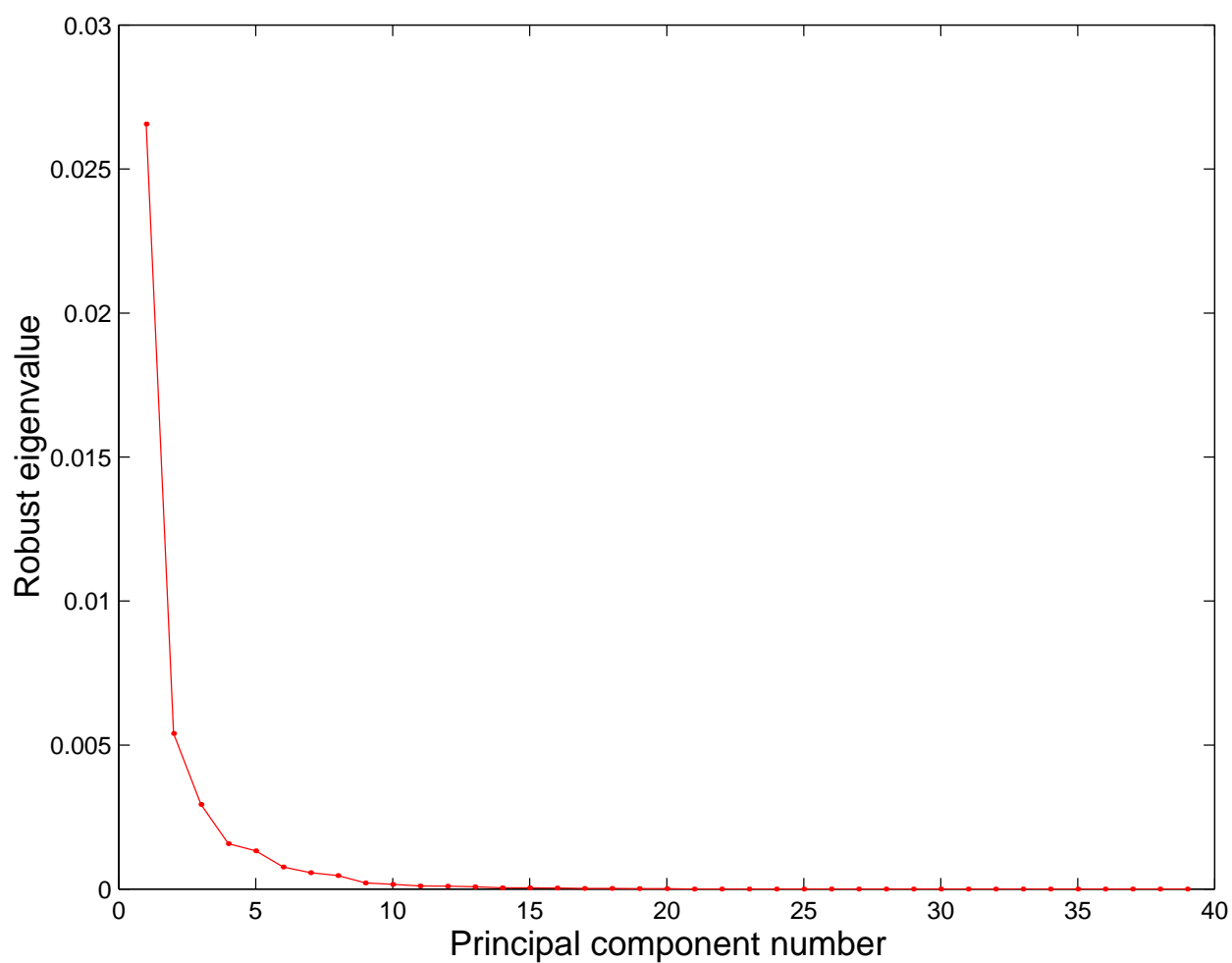
Biscuit Dough data set

Data set (Osborne et al. 1984) contains NIR spectra of biscuit dough ($n = 40, p = 600$).

1. Data are preprocessed with a logarithmic transformation. We take first differences to eliminate drift and background scatter.



2. RAPCA yields screeplot of robust eigenvalue versus principal component number:

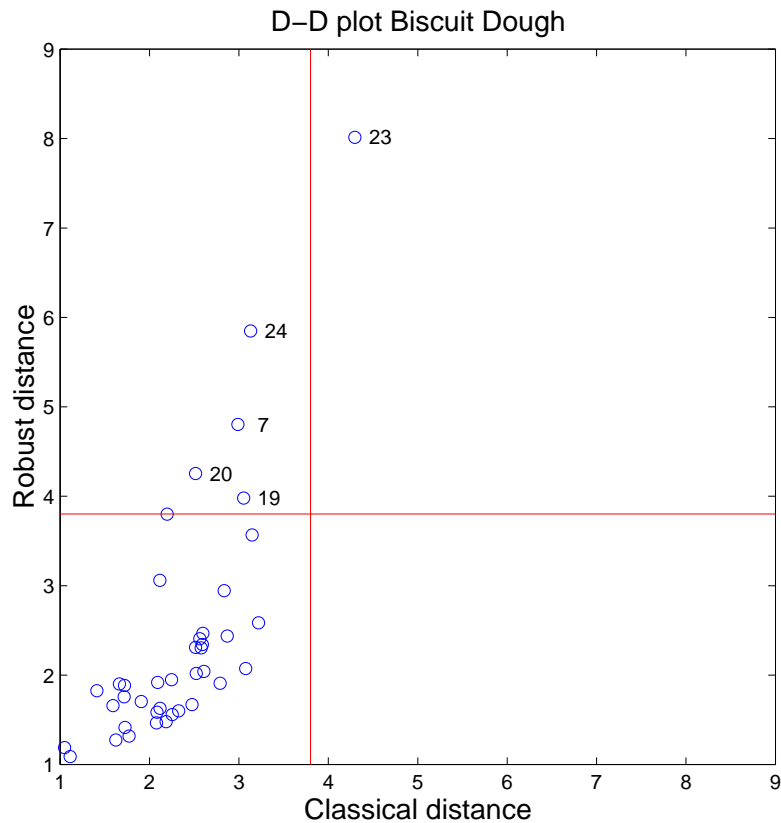


We decide to keep $k = 6$ latent variables.

3. To detect outliers we look at the Distance-Distance plot (Rousseeuw and Van Driessen 1999). It plots the robust distances RD_i versus the classical distances CD_i defined as

$$CD_i = \sqrt{\sum_{j=1}^k \left(\frac{t_{ij}^C}{s_j^C} \right)^2} \quad RD_i = \sqrt{\sum_{j=1}^k \left(\frac{t_{ij}^R}{s_j^R} \right)^2}$$

Distances which exceed the cut-off value $\sqrt{\chi_{k,0.975}^2}$ flag outliers.

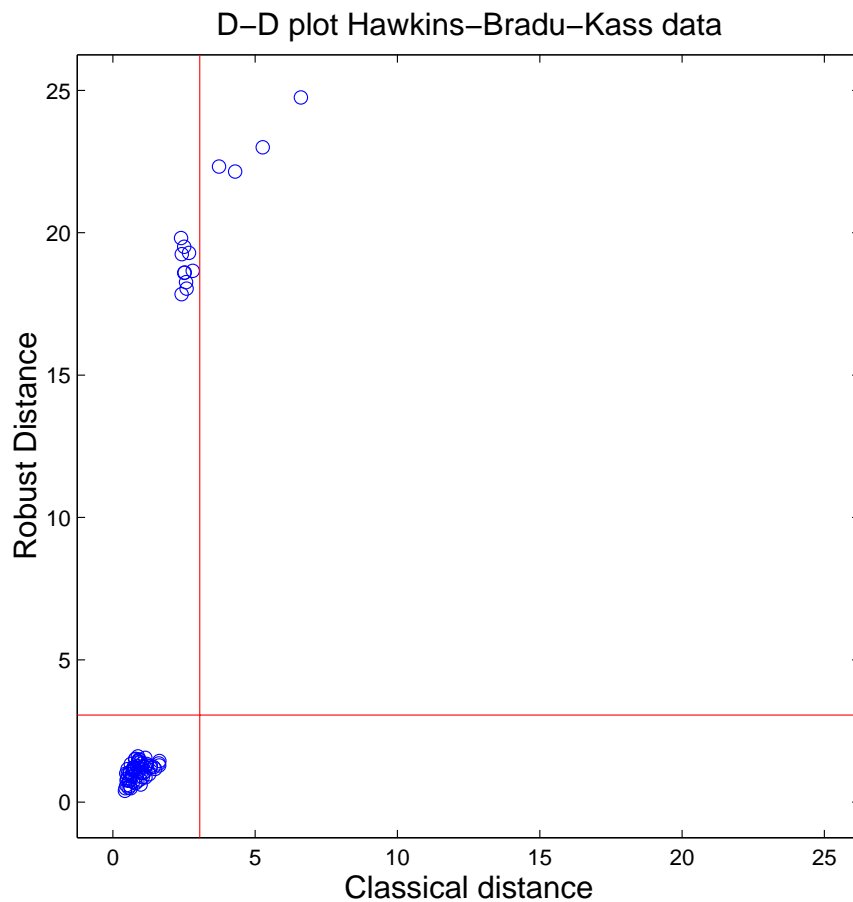


Hawkins-Bradu-Kass data set

$n = 75, p = 4$ (Technometrics 1984)

Variables X_1, X_2, X_3, Y (meant for regression)

Objects 1-14 are known outliers: first ten have outlying \mathbf{x}_i and y_i values and the last four are good leverage points (only \mathbf{x}_i is outlying).

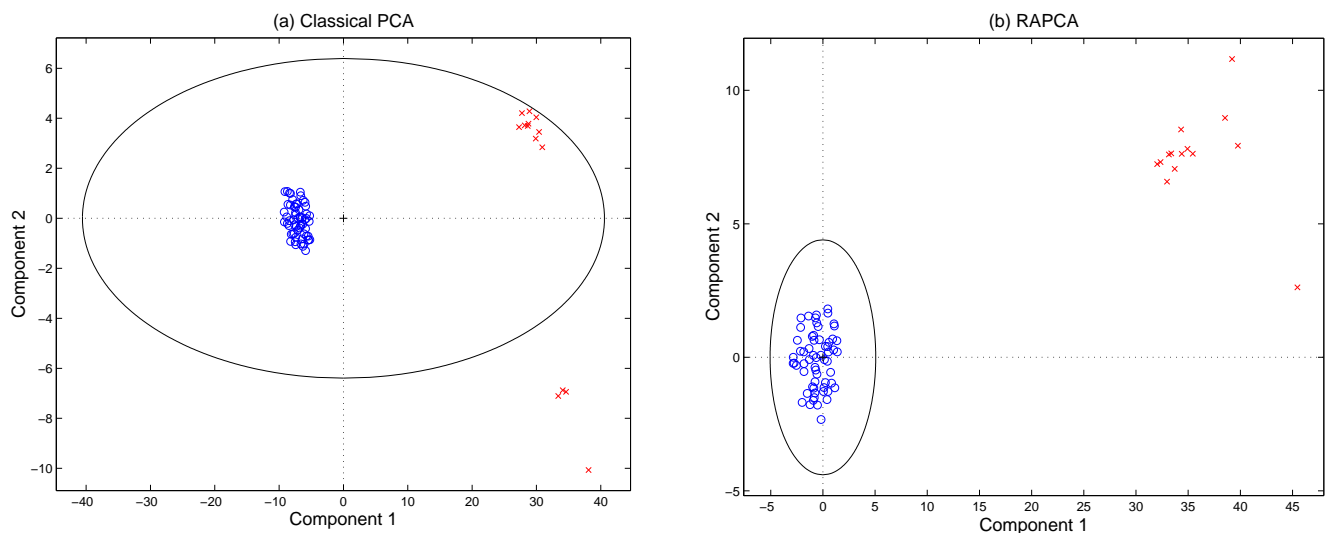


On the score plots we superimpose the 97.5% tolerance ellipse defined by:

$$\left(\frac{t_{i1}}{s_1}\right)^2 + \left(\frac{t_{i2}}{s_2}\right)^2 = 7.38 \quad (= \chi_{2,0.975}^2)$$

Observations outside the tolerance ellipse are flagged as outliers.

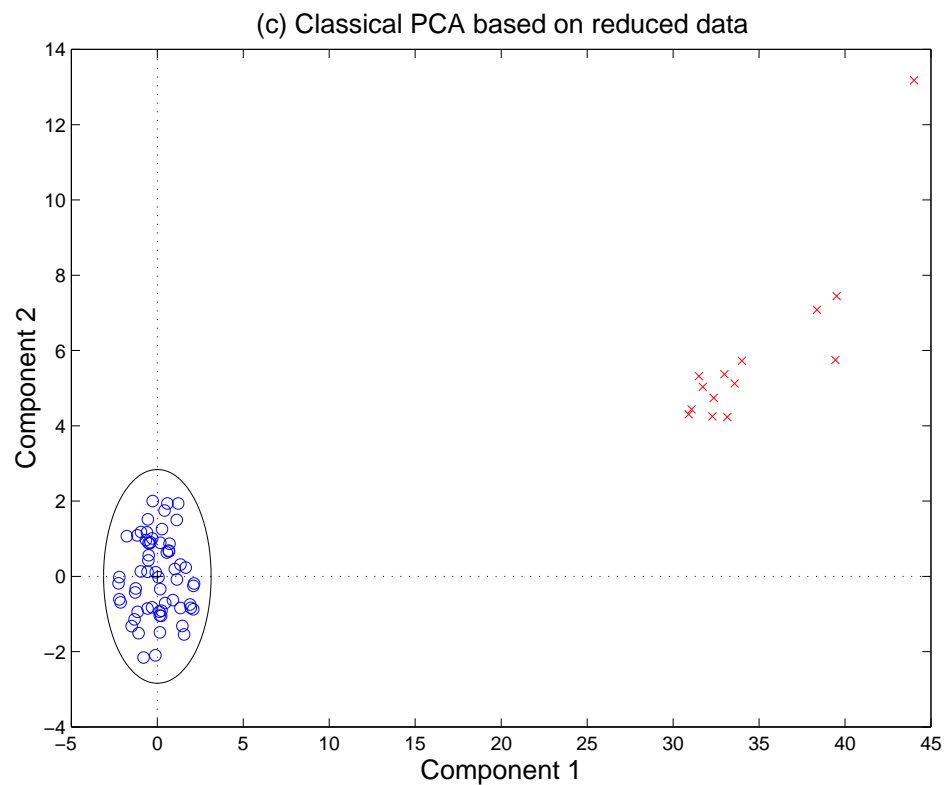
Classical PCA and RAPCA scores:



- PCA identifies 4 outliers.
- RAPCA finds all 14 outliers.
- RAPCA obtains center of good data, PCA does not.

Classical PCA on the reduced data set without outliers (PCA_{red}):

Computations based on remaining 61 points.



- all outliers are identified.
- the good data points are well centered.
- comparable with RAPCA_{full} but eigenvalues are smaller.

RAPCA on reduced set (RAPCA_{red}):

Similar scores plot. Eigenvalues:

	PCA_{full}	RAPCA_{full}	PCA_{red}	RAPCA_{red}
s_1^2	223.12	3.47	1.33	1.60
s_2^2	5.54	2.63	1.09	1.33
s_3^2	1.69	2.47	0.96	1.24
s_4^2	0.91	0.67	0.30	0.37

\Rightarrow eigenvalues close to those of PCA_{red} .

Note that PCA_{full} finds one very large eigenvalue, so would keep only **one** latent variable. The robust results all yield **three** latent variables.

Conclusions

- RAPCA deals with situations where $p \gg n$.
- It combines numerical accuracy with computation speed.
- In the presence of outliers, RAPCA describes variability of the good data points.
- RAPCA induces a robust covariance matrix, defined as

$$S = \sum_l s_l^2 \mathbf{v}_l \mathbf{v}_l^t$$

with eigenvectors \mathbf{v}_l and eigenvalues s_l^2 .

Applications

- ROBPCA: robust PCA combining projection pursuit in high dimensions and MCD in low dimensions
- Robust PCR, combining ROBPCA and robust regression
- Robust PLS (Partial Least Squares) based on ROBPCA
- Applications in bio-informatics
- Model selection in PCA, PCR, and PLS: fast robust cross-validation
- Robust discriminant analysis (classification) based on MCD