Robust Principal Component Analysis Based on Projection Pursuit

Claus Reichel*

June 4, 1997

*http://stud1.tuwien.ac.at/~e9300654/

1 Introduction

1.1 Notation

$\mathbf{\Gamma}$	an uppercase bold symbol stands for a matrix
$\boldsymbol{\delta}$	and a lowercase bold sybol is used for a vector
$\boldsymbol{x}_{i.}$	represents the i-th row of the two dimensional matrix $\boldsymbol{x}$
Diag($x_1, \ldots, x_n$)	$\begin{pmatrix} x_1 & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & x_n \end{pmatrix}$
diag($\mathbf{\Sigma}$)	$(\sigma_{11}, \ldots, \sigma_{nn})$

1.2 Principal Component Analysis

The Principal Component Analysis (PCA) goes back to Karl Pearson (1901) and was developed by Hotelling (1933). The aim is to reduce the dimensions of a collection of observations. This is done by finding linearcombinations of the variables with a minimum loss of information. This method builds the base for many other multivariant methods. Take a look at [Filzmoser, 1997, Chapter 3].

1.2.1 Theory

There is

$$\boldsymbol{x} = (x_1, \ldots, x_p)^T \tag{1.1}$$

an p-dimenstional random vector with a middle vector $\boldsymbol{\mu}$ and a covariance matrix

$$\mathbf{\Sigma} = E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]. \tag{1.2}$$

Further is

$$\mathbf{\Gamma} = (\boldsymbol{\gamma}_{.1}, \ldots, \boldsymbol{\gamma}_{.p}) \tag{1.3}$$

a (pxp)-matix with fix coefficients and the conditions that $\boldsymbol{\gamma}_{.i}$ are unitvectors ($\boldsymbol{\gamma}_{.i}^T\boldsymbol{\gamma}_{.i} = 1$ for $i = 1, \ldots, p$) and $\mathbf{\Gamma}$ is orthogonal ($\mathbf{\Gamma}^T = \mathbf{\Gamma}^{-1}$ and $\boldsymbol{\gamma}_{.i}^T\boldsymbol{\gamma}_{.j} = 0$ for $i \neq j$). We'll take a look at the linear transformation

$$\boldsymbol{z} = \mathbf{\Gamma}^T(\boldsymbol{x} - \boldsymbol{\mu}) \tag{1.4}$$

where

$$\boldsymbol{z} = (z_1, \ldots, z_p)^T \tag{1.5}$$

and

$$z_i = \boldsymbol{\gamma}_{.i}^T(\boldsymbol{x} - \boldsymbol{\mu}). \tag{1.6}$$

The variance for one element z_i of this new random vector is defined as

$$Var(z_i) = E[\boldsymbol{\gamma}_{.i}^T(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\gamma}_{.i}] = \boldsymbol{\gamma}_{.i}^T\boldsymbol{\Sigma}\boldsymbol{\gamma}_{.i}. \tag{1.7}$$

We need a transformation to maximize the variance with the condition of unitvectors. We get the following lagrange term which we derive by $\boldsymbol{\gamma}_{.i}$

$$\phi_i = \boldsymbol{\gamma}_{.i}^T\boldsymbol{\Sigma}\boldsymbol{\gamma}_{.i} - a_i(\boldsymbol{\gamma}_{.i}^T\boldsymbol{\gamma}_{.i} - 1) \tag{1.8}$$

$$\frac{\delta\phi_i}{\delta\boldsymbol{\gamma}_{.i}} = 2\boldsymbol{\Sigma}\boldsymbol{\gamma}_{.i} - 2a_i\boldsymbol{\gamma}_{.i} = 0 \tag{1.9}$$

$$(\boldsymbol{\Sigma} - a_i\boldsymbol{I})\boldsymbol{\gamma}_{.i} = 0 \tag{1.10}$$

$$\boldsymbol{\Sigma\Gamma} = \boldsymbol{\Gamma A} \tag{1.11}$$

Where $\boldsymbol{A}$ is the same as $\mathrm{Diag}(a_1, \ldots, a_p)$ and $a_1 \geq \cdots \geq a_p$. The equation 1.11 is also known as eigenvalueproblem. The a_is represents the eigenvalues with the coresponding eigenvectors $\boldsymbol{\gamma}_{.i}$'s. All eigenvalues are real non-negative because $\boldsymbol{\Sigma}$ is positve definit.

$$E(\boldsymbol{z}) = \boldsymbol{\Gamma}^T[E(\boldsymbol{x} - \boldsymbol{\mu})] = \boldsymbol{0} \tag{1.12}$$

$$Cov(\boldsymbol{z}) = \boldsymbol{\Gamma}^T Cov(\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^T\boldsymbol{\Sigma\Gamma} = \boldsymbol{A} \tag{1.13}$$

So we can say that the variance of the i^{th} PC is equal to a_i the i^{th} eigenvalue of $\boldsymbol{\Sigma}$. By the way $Cor(z_i, z_j) = 0$ for $i \neq j$ and the sum of the variance is equal to the trace of $\boldsymbol{\Sigma}$.

The covarince between the variables and the PC's is defined as

$$Cov(\boldsymbol{x}, \boldsymbol{z}) = E[\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{z}^T] = E[\boldsymbol{\Gamma z z}^T] = \boldsymbol{\Gamma A}. \tag{1.14}$$

The correlation between the variables and the PC's is defined as

$$Corr(\boldsymbol{x}, \boldsymbol{z}) = \frac{Cov(\boldsymbol{x}, \boldsymbol{z})}{\sqrt{Var(\boldsymbol{x})}\sqrt{Var(\boldsymbol{z})}} \tag{1.15}$$

$$\lambda_{ij} = \frac{Cov(x_i, z_j)}{\sqrt{Var(x_i)}\sqrt{Var(z_j)}} = \frac{\gamma_{ij}a_j}{\sigma_{ii}^{\frac{1}{2}}a_j^{\frac{1}{2}}} = \gamma_{ij}\frac{\sqrt{a_j}}{\sqrt{\sigma_{ii}}} = \gamma_{ij}\sqrt{\frac{a_j}{\sigma_{ii}}} \tag{1.16}$$

In matrix form this is equal to

$$\boldsymbol{\Lambda} = (Diag(\boldsymbol{\Sigma}))^{-\frac{1}{2}}\boldsymbol{\Gamma A}^{\frac{1}{2}} \tag{1.17}$$

1.2.2 Outliers

This method is not really robust gainst outliers because we use the arithmetic mean as estimation of $\boldsymbol{\mu}$. This estimator has a breakpoint[1] of 0 %. As seen in section 1.2.1 there are different computations in the PCA which are based on this estimation too. Further is the observed data used for the computation of $\boldsymbol{\Sigma}$ and for the variance.

1.2.3 Geometry

If we use the PCA for a given data set we will receive all PC's at once as shown in figure 1. The 1^{st} PC represents the greatest variance in the data, the 2^{nd} .... As we will see in the examples (6 and 8) the 1^{st} PC is mainly influenced by outliers.

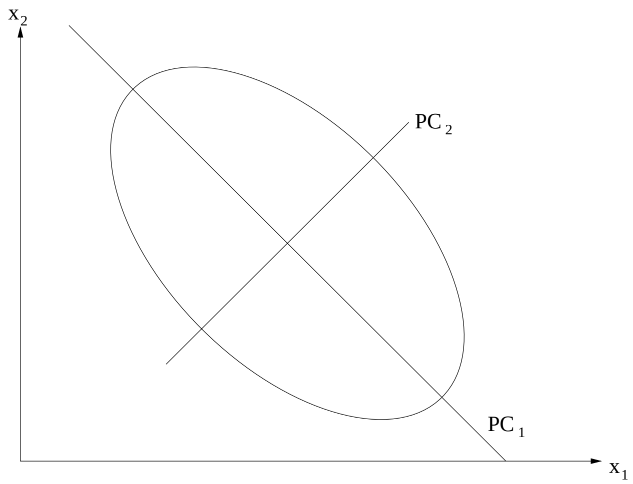

Figure 1: Principal Component Analysis

[1]Measurement for the maximum %age of outliers allowed in the observed data without influence of the estimation if the sample size $n \to \infty$.

2 Robust Principal Components based on Projection Pursuit

The difference to the classic PCA is that with this method we calculate each PC seperate, so we have the possibility to stop after n PC's. For the robustness of the method we use the

$$medmed(\boldsymbol{y}) = \underset{i}{med}|y_i - \underset{j}{med}(y_j)|. \tag{2.18}$$

We have observations

$$X = (x_1, \ldots, x_n) \text{ in } \ R^p. \tag{2.19}$$

As mentioned in the name of the method there is a projection matrix. We suppose that the first $k-1$ projection directions are already known. $\boldsymbol{P}_k$ stands for the projection matrix.

$$\boldsymbol{P}_1 = \boldsymbol{I}_p \text{ and } \quad \boldsymbol{P}_k = \boldsymbol{I}_p - \sum_{j=1}^{k-1} \boldsymbol{\gamma}_{.j}\boldsymbol{\gamma}_{.j}^T. \tag{2.20}$$

We are searching for a projection direction $\boldsymbol{a}$ for which the function

$$\boldsymbol{a} \rightarrow S(\boldsymbol{X}\boldsymbol{P}_k\boldsymbol{a}) = S\left(\begin{pmatrix} \boldsymbol{x}_{1.}^T \\ \vdots \\ \boldsymbol{x}_{n.}^T \end{pmatrix} \boldsymbol{P}_k\boldsymbol{a}\right) \tag{2.21}$$

is maximal with the condtions

$$\boldsymbol{a}^T\boldsymbol{a} = 1 \text{ and } \quad \boldsymbol{P}_k\boldsymbol{a} = \boldsymbol{a}. \tag{2.22}$$

The last condition in 2.22 represents the demand for orthoganlity to all previous found projection directions.
To save computing time the space of possible directions $\boldsymbol{a}$ is reduced to

$$A_{n,k} = \left(\frac{\boldsymbol{P}_k(\boldsymbol{X}_{1.} - \widehat{\boldsymbol{\mu}}_n)}{||\boldsymbol{P}_k(\boldsymbol{X}_{1.} - \widehat{\boldsymbol{\mu}}_n)||}, \ldots, \frac{\boldsymbol{P}_k(\boldsymbol{X}_{n.} - \widehat{\boldsymbol{\mu}}_n)}{||\boldsymbol{P}_k(\boldsymbol{X}_{n.} - \widehat{\boldsymbol{\mu}}_n)||}\right). \tag{2.23}$$

$\widehat{\boldsymbol{\mu}}_n$ stands for the L1-Median. So we are just searching kn directions if we like to know the first k robust PC's and n^2 for all PC's. [Filzmoser, 1997, Chapter 4]

3 Examples

The 1^{st} variable of the given data in the plots in figure 2 and 3 was multipled with 10^4 to avoid problems with the tolerance[2] in different computations. In figure 2 we can see how outliers influence the PC's if we use the classic PCA. Figure 3 shows the result of the Projection-Pursuit-Algorithmus (uses the L1median[3]) and robust PC where $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\boldsymbol{\mu}}$ was calculated with the MVE[4]. Figure 4 and 5 shows the same computation as figure 2 and 3 but with scaled data.
The figures 6 and 7 show the difference between the classic PCA and the robust PCA (PP-Algorthmus). In the boxplots in the figures 6 and 7 you can see that in the classic PCA the 1^{st} PC is mainly influenced by outliers. With a robust PCA the outliers are more distributed to all other PC's too, see [Prat, 1996, page 211-216].

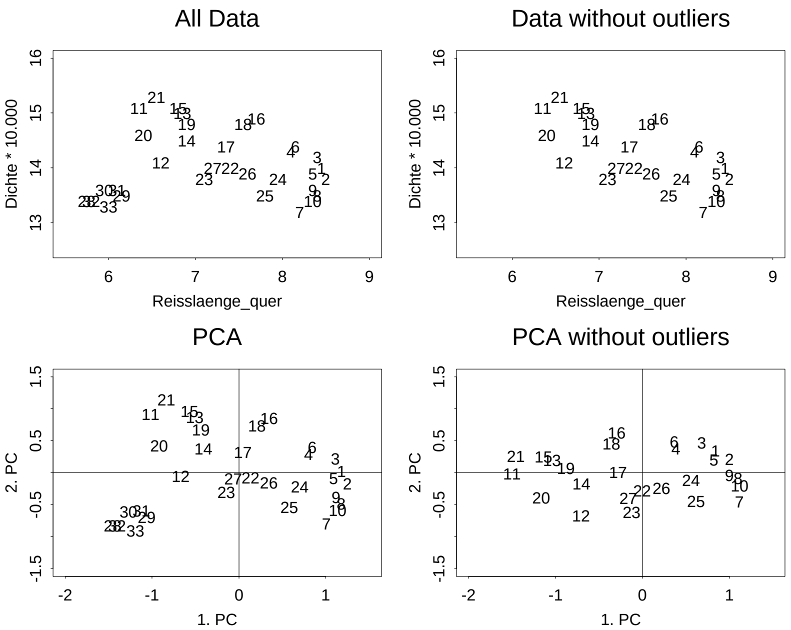

Figure 2: Data from Gölles & Dutter not scaled with classic PCA

[2]see line 10 in A.2
[3]The corrected version of the L1median is included in appendix A.1
[4]Minimum-Volume-Ellipsoid, Rousseeuw 1983. Breakpoint 50 %

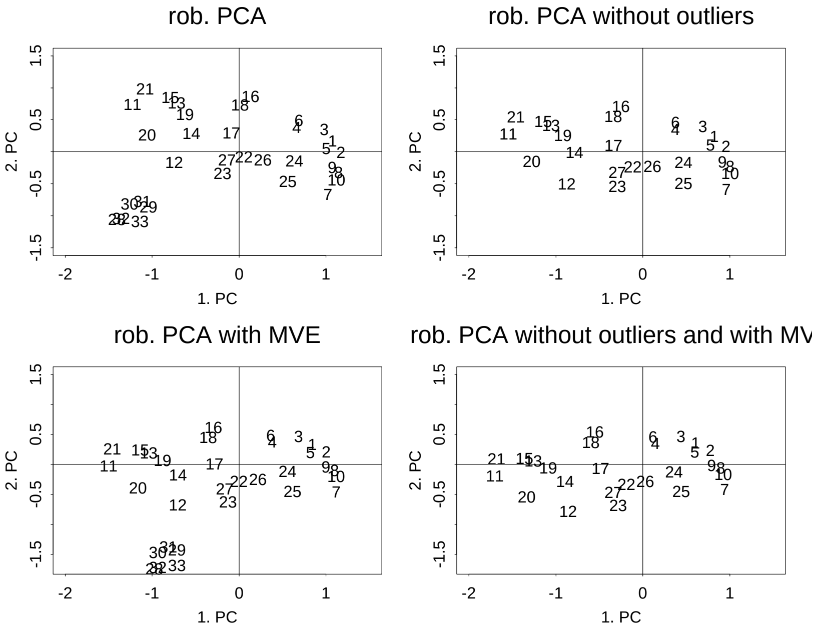

Figure 3: Data from Gölles & Dutter not scaled
1^{st} row: calculated with the Projection-Pursuit-Algorithmus A.2
2^{nd} row: $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\boldsymbol{\mu}}$ calculated with the MVE

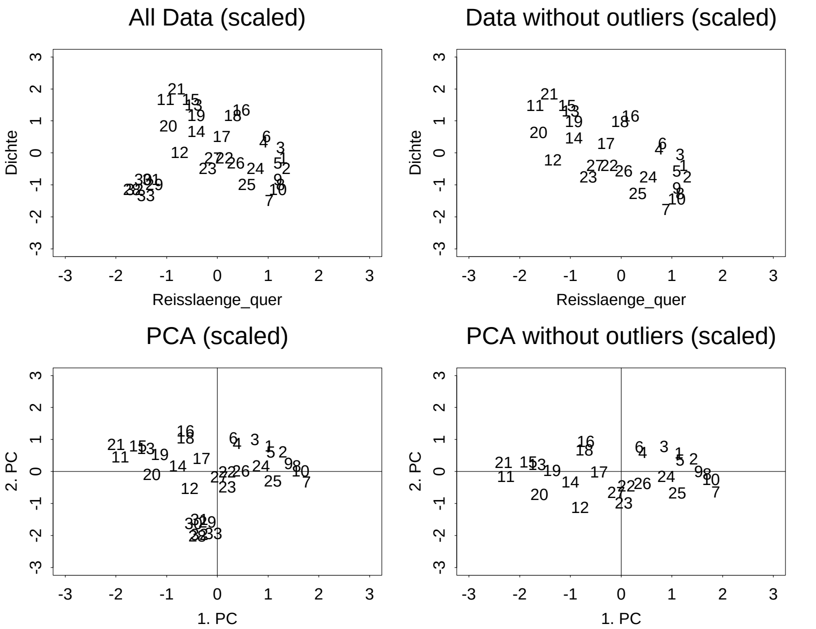

Figure 4: Same as figure 2 but with scaled data

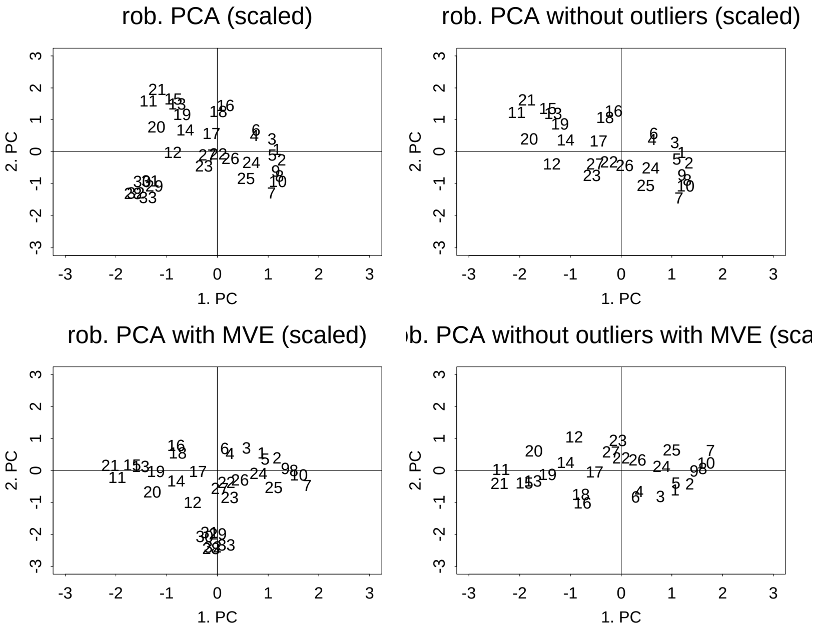

Figure 5: Same as figure 3 but with scaled data

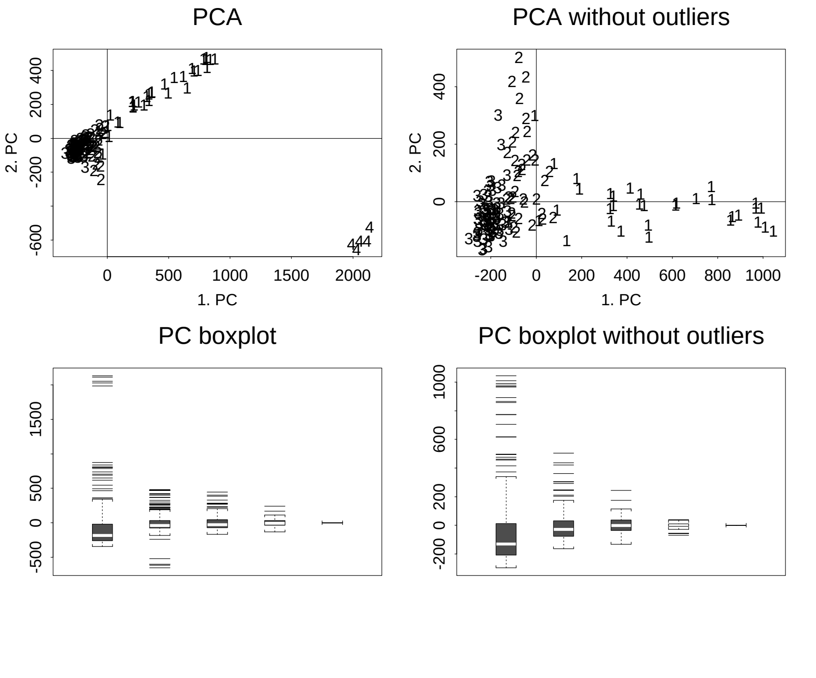

Figure 6: Diabetes data with additional outliers (group 4)
2^{nd} row: boxplot of the scores splitted by the 5 PC's

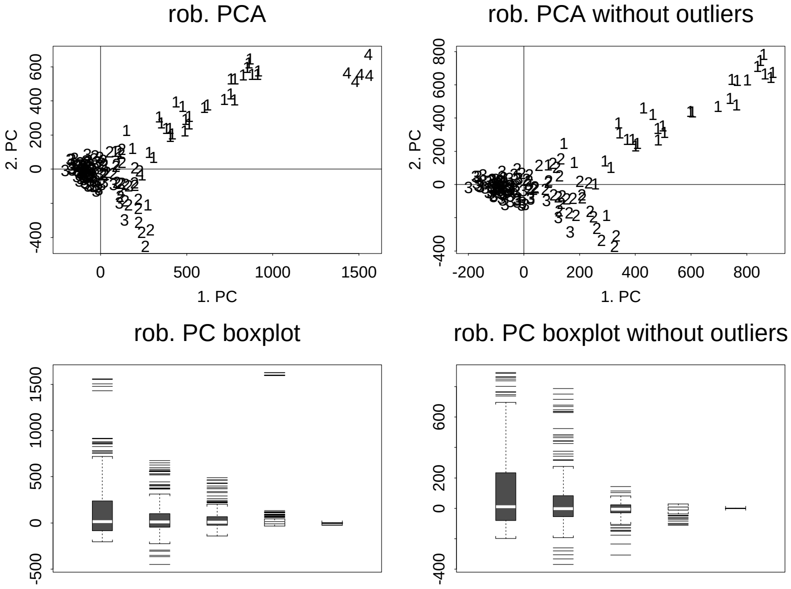

Figure 7: Same as in figure 6 but with robust PC's

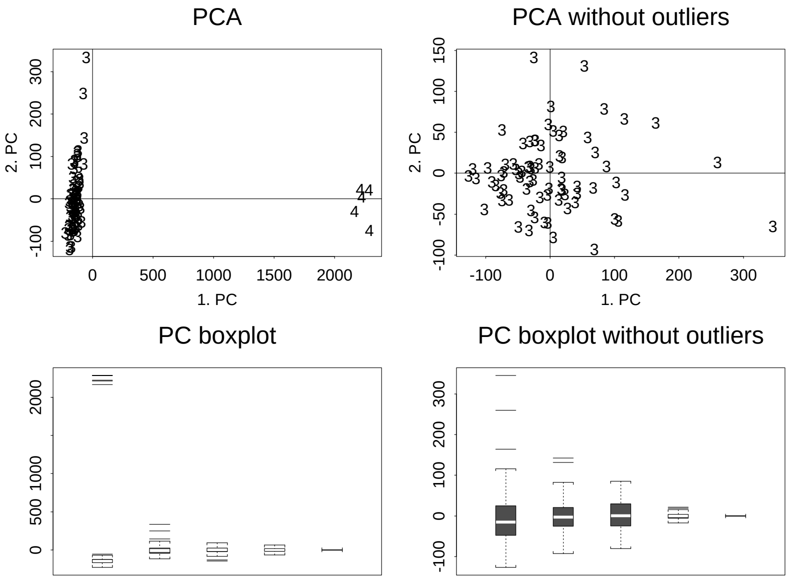

Figure 8: Diabetes data (only group 3) with additional outliers (group 4) 2^{nd} row: boxplot of the scores splited by the 5 PC's

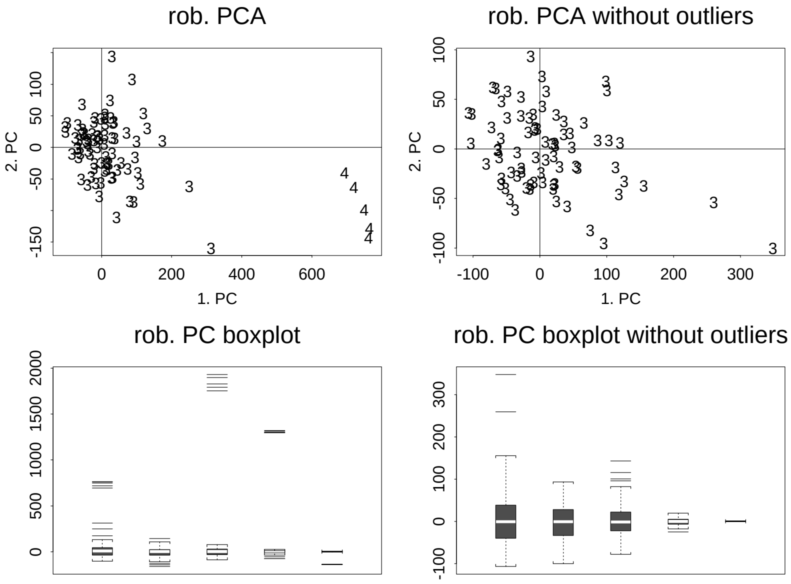

Figure 9: Same as in figure 8 but with robust PC's

A S-Plus-Code

A.1 L1median

```
function(X, tolexp = -3, maxstep = 200)
{
      ittol <- 10^(tolexp)
      m <- apply(X, 2, median)
      k <- 0
      while(k < maxstep) {
            k <- k + 1
            mold <- m
            Y <- scale(X, center = m, scale = F)
            Ynorm <- apply(Y, 1, vecnorm)
            w <- ifelse(Ynorm == 0, 0, 1/Ynorm)
            W <- diag(w)
            delta <- apply(W %*% Y, 2, sum)/sum(w)
            nd <- vecnorm(delta)
            if(nd < ittol)
                  maxhalf <- 0
            else maxhalf <- log(nd/ittol)/log(2)
            m <- mold + delta
            nstep <- 0
            while((obj(X, m) >= obj(X, mold)) && (nstep <= maxhalf)) {
                  nstep <- nstep + 1
                  m <- mold + (delta/(2^nstep))
            }
            d <- delta/(2^nstep)
            if(nstep >= maxhalf)
                  return(as.matrix(m))
      }
      if(k == maxstep)
            return(as.matrix(m))
      if(k > maxstep)
            stop("iteration failed")
}
```

A.2 prcomp.rob

```
function(X, sca = NULL, scores = F)
{
      n <- nrow(X)
      p <- ncol(X)
      p1 <- min(n, p)
      s <- rep(1, n)
      S <- rep(1, p1)
      V <- matrix(1:(p * p1), ncol = p1, nrow = p)
      P <- diag(p)
```

```
      m <- l1median(X)
      for(k in 1:p1) {
            Xcentr <- scale(X, center = m, scale = F)
            B <- P %*% t(Xcentr)
            Bnorm <- apply(B, 2, vecnorm)
            A <- B %*% diag(1/Bnorm)
            Y <- X %*% t(P) %*% A
            if(sca == "mad")
                  s <- apply(Y, 2, mad)
            if(sca == "tau")
                  s <- apply(Y, 2, scale.tau)
            if(sca == "A")
                  s <- apply(Y, 2, scale.a)
            j <- order(s)[n]
            S[k] <- s[j]
            V[, k] <- A[, j]        #if(V[1, k] < 0)
            V[, k] <- (-1) * V[, k]
            P <- P - (V[, k] %*% t(V[, k]))
      }
      if(scores) {
            Y <- scale(X, center = m, scale = F)
            list(scale = S, rotation = V, scores = Y %*% V)
      }
      else list(scale = S, rotation = V)
}
```

A.3 Gölles & Dutter data plots not scaled

```
function()
{
#
# global settings
#
      par(mfrow = c(2, 2))    #
      tol <- -3       #
# Input G"olles & Dutter
#
      x <- matrix(scan("se/testdust.ori"), ncol = 2, byrow = T)
      x6 <- matrix(scan("se/testdust.-6"), ncol = 2, byrow = T)       #
# there are some problems (internal comp.) without transforming the
# 1st col
#
      x[, 1] <- (10^4) * x[, 1]
      x6[, 1] <- (10^4) * x6[, 1]     #
# calculate the MVE
#
      xmve <- cov.mve(x)
      x6mve <- cov.mve(x6)    #
```

```
# all data
#
      plot(x[, 2], x[, 1], ylab = "Dichte * 10.000", xlab =
            "Reisslaenge_quer", type = "n", xlim = c(5.5, 9),
            ylim = c( 12.5, 16))
      text(x[, 2], x[, 1], labels = seq(1, dim(x)[1]))
      title("All Data")
      dev.ask(T)     #
# data without outliers
#
      plot(x6[, 2], x6[, 1], ylab = "Dichte * 10.000", xlab =
            "Reisslaenge_quer", type = "n", xlim = c(5.5, 9),
            ylim = c( 12.5, 16))
      text(x6[, 2], x6[, 1], labels = seq(1, dim(x6)[1]))
      title("Data without outliers")
      dev.ask(T)     #
# classic PCA
#
      xpc <- princomp(x)
      plot(xpc$scores, xlab = "1. PC", ylab = "2. PC", type = "n",
            xlim = c( -2, 1.5), ylim = c(-1.5, 1.5))
      text(xpc$scores, labels = seq(1, dim(x)[1]))
      title("PCA")
      abline(h = 0, v = 0)
      dev.ask(T)     #
# classic PCA without outliers
#
      xpc6 <- princomp(x6)
      plot(xpc6$scores, xlab = "1. PC", ylab = "2. PC", type = "n",
            xlim = c( -2 , 1.5), ylim = c(-1.5, 1.5))
      title("PCA without outliers")
      text(xpc6$scores, labels = seq(1, dim(x6)[1]))
      abline(h = 0, v = 0)
      dev.ask(T)     #
# rob PCA PP-Algo
#
      xpcr <- prcomp.rob(x, sca = "mad", scores = T)
      plot( - xpcr$scores[, 1],  - xpcr$scores[, 2], xlab = "1. PC",
            ylab = "2. PC", type = "n", xlim = c(-2, 1.5),
            ylim = c(-1.5, 1.5))
      title("rob. PCA")
      text( - xpcr$scores[, 1],  - xpcr$scores[, 2], labels = seq(1,
            dim(x)[1]))
      abline(h = 0, v = 0)
      dev.ask(T)     #
# rob PCA without outliers PP-Algo
#
      xpcr6 <- prcomp.rob(x6, sca = "mad", scores = T)
      plot(xpcr6$scores[, 1],  - xpcr6$scores[, 2], xlab = "1. PC",
```

```
            ylab = "2. PC", type = "n", xlim = c(-2, 1.5),
            ylim = c(-1.5, 1.5))
      title("rob. PCA without outliers")
      text(xpcr6$scores[, 1],  - xpcr6$scores[, 2], labels = seq(1,
            dim(x6)[1]))
      abline(h = 0, v = 0)
      dev.ask(T)      #
# rob PC with MVE
#
#
      xev <- eigen(xmve$cov)$vectors
      xmpc <- (x - seq(1, by = 0, length = dim(x)[1]) %*%
            t(xmve$center)) %*% xev
      plot(xmpc[, 1], xmpc[, 2], xlab = "1. PC", ylab = "2. PC",
            type = "n", xlim = c(-2, 1.5), ylim = c(-1.7, 1.5))
      title("rob. PCA with MVE")
      text(xmpc[, 1], xmpc[, 2], labels = seq(1, dim(x)[1]))
      abline(h = 0, v = 0)
      dev.ask(T)      #
# rob PC with MVE without outliers
#
#
      x6ev <- eigen(x6mve$cov)$vectors
      x6mpc <- (x6 - seq(1, by = 0, length = dim(x6)[1]) %*%
            t(x6mve$center)) %*% x6ev
      plot(x6mpc, xlab = "1. PC", ylab = "2. PC", type = "n",
            xlim = c(-2, 1.5), ylim = c(-1.7, 1.5))
      title("rob. PCA without outliers and with MVE")
      text(x6mpc, labels = seq(1, dim(x6)[1]))
      abline(h = 0, v = 0)
      dev.ask(T)      #
#
}
```

A.4 Gölles & Dutter data plots scaled

Substitute the given lines in the code of section A.3.

```
15    x <- scale(x)
16    x6 <- scale(x6)
```

A.5 Diabetes

```
function()
{
#
# global settings
#
```

```
      par(mfrow = c(2, 2))    #
# Input diabetes with outliers
#
      d5 <- matrix(scan("se/diabet"), ncol = 6, byrow = T)
      d <- matrix(scan("se/diabet.aus"), ncol = 6, byrow = T)
      names <- c("Gewicht", "Plasmaglukose", "Glukoseflaeche",
            "Insulinflaeche", "Widerstandskraft", "Objektnamen")
      dimnames(d5) <- list(NULL, names)
      dimnames(d) <- list(NULL, names)        #
# classic PCA
#
      x <- princomp(d[, 1:5])
      x5 <- princomp(d5[, 1:5])
      plot(x$scores[, 1], x$scores[, 2], type = "n", xlab = "1. PC",
            ylab = "2. PC")
      text(x$scores[, 1], x$scores[, 2], d[, 6])
      title("PCA")
      abline(h = 0, v = 0)
      dev.ask(T)      #
# classic PCA without outliers
#
      plot(x5$scores[, 1], x5$scores[, 2], type = "n", xlab = "1. PC",
             ylab = "2. PC")
      text(x5$scores[, 1], x5$scores[, 2], d5[, 6])
      title("PCA without outliers")
      abline(h = 0, v = 0)
      dev.ask(T)      #
# corresponding boxplots
#
      boxplot(x$scores[, 1], x$scores[, 2], x$scores[, 3],
            x$scores[, 4], x$scores[, 5])
      title("PC boxplot")
      dev.ask(T)
      boxplot(x5$scores[, 1], x5$scores[, 2], x5$scores[, 3],
            x5$scores[, 4], x5$scores[, 5])
      title("PC boxplot without outliers")
      dev.ask(T)      #
# rob PCA same as before
#
      x <- prcomp.rob(d[, 1:5], sca = "mad", scores = T)
      x5 <- prcomp.rob(d5[, 1:5], sca = "mad", scores = T)
      plot(x$scores[, 1], x$scores[, 2], type = "n", xlab = "1. PC",
            ylab = "2. PC")
      text(x$scores[, 1], x$scores[, 2], d[, 6])
      title("rob. PCA")
      abline(h = 0, v = 0)
      dev.ask(T)
      plot(x5$scores[, 1], x5$scores[, 2], type = "n", xlab = "1. PC",
             ylab = "2. PC")
```

```
        text(x5$scores[, 1], x5$scores[, 2], d5[, 6])
        title("rob. PCA without outliers")
        abline(h = 0, v = 0)
        dev.ask(T)
        boxplot(x$scores[, 1], x$scores[, 2], x$scores[, 3],
              x$scores[, 4], x$scores[, 5])
        title("rob. PC boxplot")
        dev.ask(T)
        boxplot(x5$scores[, 1], x5$scores[, 2], x5$scores[, 3],
              x5$scores[, 4], x5$scores[, 5])
        title("rob. PC boxplot without outliers")
        dev.ask(T)
}
```

B About the document

This document was created with LaTeX [Michel Goossens, 1994] under Linux (Kernel 2.0.11). The following styles were used: isolatin1, amstex, psfig, moreverb, float and fancyheadings. The file contains 19935 characters in 563 lines and includes 9 postscript files.

References

[Filzmoser, 1997] Filzmoser, P. (1997). *Multivariate Methoden.* Institut für Statistik und Wahrscheinlichkeitstheorie, Wien.

[Michel Goossens, 1994] Michel Goossens, F. M. u. A. S. (1994). *Der LaTeXBegleiter.* Addison-Wesley, Bonn; Paris, erste Auflage / erster korrigierter Nachdruck 1996.

[Prat, 1996] Prat, A., editor (1996). *Proceedings in computational statistics.* Physica-Verlag, Heidelberg. Comstat 1996.