An Object-Oriented Solution for Robust Factor Analysis *

Ying-Ying Zhang
Chongqing University

Abstract

Taking advantage of the S4 class system of the programming environment R, which facilitates the creation and maintenance of reusable and modular components, an object-oriented solution for robust factor analysis is developed. The solution resides in the package **robustfa**. In the solution, new S4 classes `"Fa"`, `"FaClassic"`, `"FaRobust"`, `"FaCov"`, `"SummaryFa"` are created. The robust factor analysis function `FaCov()` can deal with maximum likelihood estimation, principal component analysis, and principal factor analysis methods. Consider the factor analysis methods (classical or robust), the data input (data or the scaled data), and the running matrix (covariance or correlation) all together, there are 8 combinations. We recommend classical and robust factor analysis using the correlation matrix of the scaled data as the running matrix for theoretical and computational reasons. The design of the solution follows common statistical design patterns. The application of the solution to multivariate data analysis is demonstrated on the `hbk` data and the `stock611` data which themselves are parts of the package **robustfa**.

Keywords: robustness, factor analysis, object-oriented solution, R, statistical design patterns.

1. Introduction

Outliers exist in virtually every data set in any application domain. In order to avoid the masking effect, robust estimators are needed. The classical estimators of multivariate location and scatter are the sample mean $\bar{\mathbf{x}}$ and the sample covariance matrix $\mathbf{S}$. These estimates are optimal if the data come from a multivariate normal distribution but are extremely sensitive to the presence of even a few outliers. If outliers are present in the input data they will influence the estimates $\bar{\mathbf{x}}$ and $\mathbf{S}$ and subsequently worsen the performance of the classical factor analysis (Pison *et al.* 2003). Therefore it is important to consider robust alternatives to these estimators. There are several robust estimators in the literature: MCD (Rousseeuw 1985; Rousseeuw and Driessen 1999), OGK (Maronna and Zamar 2002), MVE (Rousseeuw 1985), M (Rocke 1996), S (Davies 1987; Ruppert 1992; Woodruff and Rocke 1994; Rocke 1996; He and Wang 1997; Salibian-Barrera and Yohai 2006) and Stahel-Donoho (Stahel 1981a,b; Donoho 1982; Maronna and Yohai 1995). Substituting the classical location and scatter estimates by their robust analogues is the most straightforward method for robustifying many multivariate procedures (Maronna *et al.* 2006; Todorov and Filzmoser 2009), which is our choice for robustifying the factor analysis procedure.

*The research was supported by Natural Science Foundation Project of CQ CSTC CSTC2011BB0058.

Taking advantage of the new S4 class system (Chambers 1998) of R (R Development Core Team 2009) which facilitate the creation of reusable and modular components, an object-oriented solution for robust factor analysis is implemented. The application of the solution to multivariate data analysis is demonstrated on the `hbk` data and the `stock611` data which themselves are parts of the package **robustfa** (Zhang 2013). We follow the object-oriented paradigm as applied to the R object model (naming conventions, access methods, coexistence of S3 and S4 classes, usage of UML, etc.) which is described in (Todorov and Filzmoser 2009).

The rest of the paper is organized as follows. In the next Section 2 the design approach and structure of the solution are given. Section 3 describes the robust factor analysis method, some theoretical results, the computations, and the implementations. The Sections 3.1, 3.2, 3.3, and 3.4 are dedicated to the object model, method of robust factor analysis, a `hbk` data example, and a stocks data example, respectively. Section 4 concludes.

2. Design approach and structure of the solution

We follow the route of (Todorov and Filzmoser 2009). The main part of the solution is implemented in the package **robustfa** but it relies on codes in the packages **rrcov** (Todorov 2013), **robustbase** (Rousseeuw *et al.* 2013), and **pcaPP** (Filzmoser *et al.* 2013). The structure of the solution and its relation to other R packages are shown in Figure 1. The package **robustfa** extends **rrcov** with options for dealing with robust factor analysis problems like **robust** (Wang *et al.* 2013).

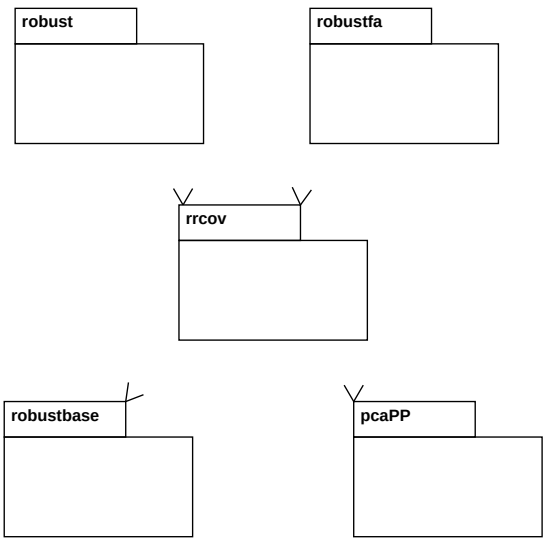


Figure 1: Class diagram: structure of the solution and relation to other R packages.

3. Robust factor analysis

3.1. Object model

The object model for the S4 classes and methods implementing the robust factor analysis follows the Unified Modeling Language (UML) (OMG 2009a,b) class diagram and is presented in Figure 2. A class is denoted by a box with three compartments which contain the name, the attributes (slots) and operations (methods) of the class, respectively. The class names, `Fa` and `FaRobust`, in italics indicate that the classes are abstract. Each attribute is followed by its type and each operation is followed by the type of its return value. We use the R types like `numeric`, `vector`, `matrix`, etc. but the type can be also a name of an S4 class (`Fa`, `FaClassic`, or `FaCov`). The classes `Ulogical`, `Unumeric` etc. are class unions for optional slots, e.g., for definition of slots which will be computed on demand. Relationships between classes are denoted by arrows with different form. The inheritance relationship is depicted by a large empty triangular arrowhead pointing to the base class. We see that both `FaClassic` and `FaRobust` inherit from `Fa`, and `FaCov` inherits from `FaRobust`. Composition means that one class contains another one as a slot. This relation is represented by an arrow with a solid diamond on the side of the composed class. We see that `SummaryFa` is a composed class of `Fa`.

As in (Todorov and Filzmoser 2009), all UML diagrams of the solution were created with the open source UML tool **ArgoUML** (Robbins 1999; Robbins and Redmiles 2000) which is available for download from http://argouml.tigris.org/. The naming conventions of the solution follow the recommended Sun's Java coding style. See http://java.sun.com/docs/codeconv/.

3.2. Factor analysis based on a robust covariance matrix

As in (Todorov and Filzmoser 2009), the most straightforward and intuitive method to obtain robust factor analysis is to replace the classical estimates of location and covariance by their robust analogues. The package **stats** in base R contains the function `factanal()` which performs a factor analysis on a given numeric data matrix and returns the results as an object of S3 class `factanal`. This function has an argument `covmat` which can take a covariance matrix, or a covariance list as returned by `cov.wt`, and if supplied, it is used rather than the covariance matrix of the input data. This allows to obtain robust factor analysis by supplying the covariance matrix computed by `cov.mve` or `cov.mcd` from the package **MASS**. The reason to include such type of function in the solution is the unification of the interfaces by leveraging the object orientation provided by the S4 classes and methods. The function `FaCov()` computes robust factor analysis by replacing the classical covariance matrix with one of the robust covariance estimators available in the package **rrcov**—MCD, OGK, MVE, M, S or Stahel-Donoho, i.e., the parameter `cov.control` can be any object of a class derived from the base class `CovControl`. This control class will be used to compute a robust estimate of the covariance matrix. If this parameter is omitted, MCD will be used by default. Of course any newly developed estimator following the concepts of the framework in (Todorov and Filzmoser 2009) can be used as input to the function `FaCov()`.

There are three methods to perform a factor analysis: maximum likelihood estimation (MLE), principal component analysis (PCA), and principal factor analysis (PFA). The function `factanal()` performs a factor analysis using MLE. The function `FaCov()` deals with three

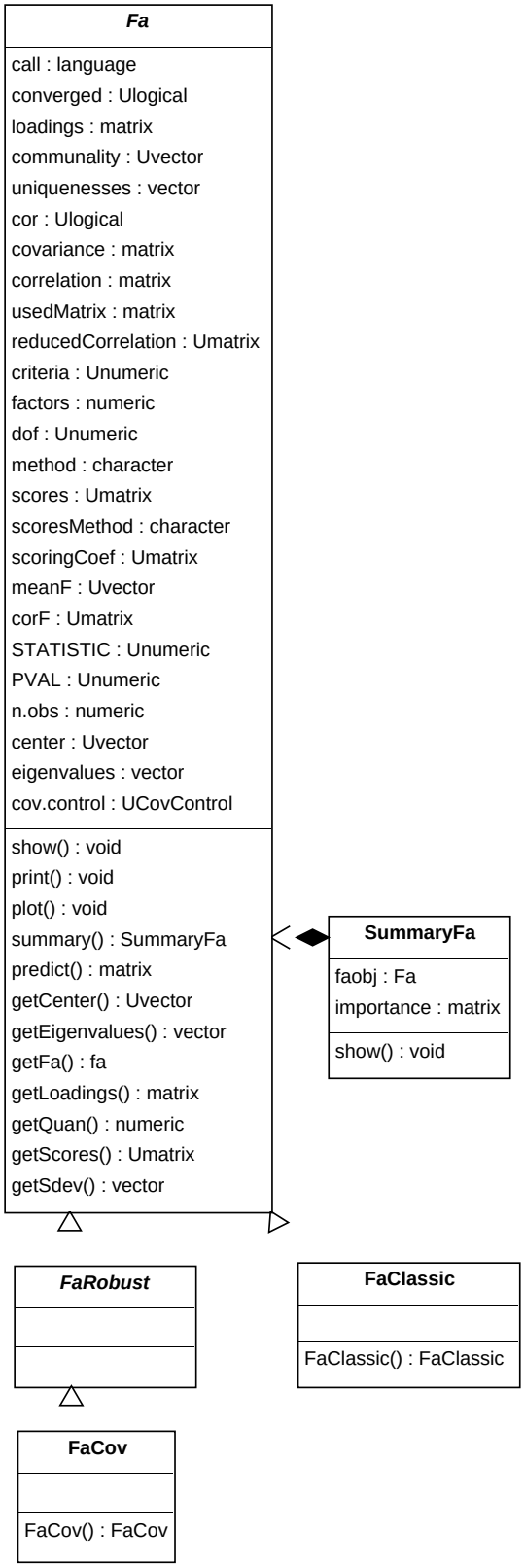


Figure 2: Object model for robust factor analysis.

methods using an argument '`method = c("mle", "pca", "pfa")`'. When the data set is large, `factanal()` usually generates errors, and thus one can only resort to the PCA and PFA methods.

When compare the classical and robust factor analysis methods, there are two other issues to consider. That is, whether we should use `data` or `scale(data)` as the data input; whether we should use the covariance matrix or the correlation matrix as the running matrix (used matrix)? Consider the factor analysis methods, the data input, and the running matrix all together, there are 8 combinations, i.e.,

```
(1) classical, x = data, cor = FALSE
(2) classical, x = data, cor = TRUE
(3) classical, x = scale(data), cor = FALSE
(4) classical, x = scale(data), cor = TRUE
(5) robust, x = data, cor = FALSE
(6) robust, x = data, cor = TRUE
(7) robust, x = scale(data), cor = FALSE
(8) robust, x = scale(data), cor = TRUE
```

Here `cor` is a logical value indicating whether the calculation should use the covariance matrix (`cor = FALSE`) or the correlation matrix (`cor = TRUE`).

There are 4 classical and robust factor analysis comparisons, i.e., (1) vs (5), (2) vs (6), (3) vs (7), and (4) vs (8). We recommend (4) vs (8). The reasons are as follows. First, when the variables have different units, it is common to standardize the variables, the sample covariance matrix of the standardized variables is the correlation matrix of the original variables. Thus it is common to use the correlation matrix as the running matrix. Second, we need to explain the factors from the loading matrix. The entries of the loading matrix from the sample covariance matrix are not limitted between 0 and 1, which makes the explanations of the factors hard. The first two reasons suggest us to choose (2) vs (6) and (4) vs (8). However, (2) and (4) ((6) and (8)) have the same running matrices, eigenvalues, loadings, uniquenesses, scoring coefficients, scaled data matrices, and score matrices, see Theorem 3.2. That is, (2) vs (6) and (4) vs (8) give us the same comparisons. We can choose any pair to do the comparison. Third, we may not be able to compute the robust covariance matrix, and thus the robust correlation matrix of the original data, as the stocks data example will illustrate. Consequently, we should choose (4) vs (8).

The running matrices of the 8 combinations are given in Table 1. In the table,

$$\text{correlation} = \text{cov2cor(covariance)}.$$

Let $\boldsymbol{S} = (s_{ij})_{p\times p}$ be a square matrix. Let $\boldsymbol{v} = \text{diag}\,(\boldsymbol{S}) = (s_{11}, s_{22}, \cdots, s_{pp})^\top$ be a vector holding the diagonal elements of $\boldsymbol{S}$. Let

$$\boldsymbol{D} = \text{diag}\,(\boldsymbol{v}) = \text{diag}\,(\text{diag}\,(\boldsymbol{S})) = \begin{pmatrix} s_{11} & & & \\ & s_{22} & & \\ & & \ddots & \\ & & & s_{pp} \end{pmatrix}$$

Table 1: The running matrices.

		Classical	Robust
data	covariance	(1) $\boldsymbol{S}^c$	(5) $\boldsymbol{S}^r$
	correlation	(2) $\boldsymbol{R}^c$	(6) $\boldsymbol{R}^r$
scale(data)	covariance	(3) $\tilde{\boldsymbol{S}}^c$	(7) $\tilde{\boldsymbol{S}}^r$
	correlation	(4) $\tilde{\boldsymbol{R}}^c$	(8) $\tilde{\boldsymbol{R}}^r$

be a diagonal matrix whose diagonal is the vector $\boldsymbol{v}$. In fact,

$$\begin{aligned} \text{diag}\,(\text{matrix}) &= \text{vector}, \\ \text{diag}\,(\text{vector}) &= \text{diagonal matrix}. \end{aligned}$$

Let

$$\boldsymbol{X}_{n\times p} = \begin{pmatrix} \boldsymbol{X}_{(1)}^\top \\ \boldsymbol{X}_{(2)}^\top \\ \vdots \\ \boldsymbol{X}_{(n)}^\top \end{pmatrix}$$

be the original data matrix, where $\boldsymbol{X}_{(k)} = (x_{k1}, x_{k2}, \cdots, x_{kp})^\top$ $(k = 1, 2, \cdots, n)$ is the kth observation, n is the number of observations. Let M be the number of regular observations (excluding the outliers), without loss of generality, we can assume $\boldsymbol{X}_{(1)}, \boldsymbol{X}_{(2)}, \cdots, \boldsymbol{X}_{(M)}$ as the regular observations. Let

$$\bar{\boldsymbol{X}} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{X}_{(k)} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_p)^\top$$

be the sample mean of $\boldsymbol{X}$, where

$$\bar{x}_j = \frac{1}{n}\sum_{k=1}^{n} x_{kj}, \;\; j = 1, 2, \cdots, p.$$

Let

$$\boldsymbol{S} = (s_{ij})_{p\times p} = \frac{1}{n-1}\sum_{k=1}^{n} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right)\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right)^\top$$

be the sample covariance matrix, where

$$s_{ij} = \frac{1}{n-1}\sum_{k=1}^{n} \left(x_{ki} - \bar{x}_i\right)\left(x_{kj} - \bar{x}_j\right), \; i, j = 1, 2, \cdots, p.$$

Let $\boldsymbol{X}^*$ be the scaled matrix of $\boldsymbol{X}$, where

$$\begin{aligned}
\boldsymbol{X}^* &= \begin{pmatrix} \boldsymbol{X}_{(1)}^{*\top} \\ \boldsymbol{X}_{(2)}^{*\top} \\ \vdots \\ \boldsymbol{X}_{(n)}^{*\top} \end{pmatrix} = \begin{pmatrix} \left(\boldsymbol{D}^{-\frac{1}{2}}\left(\boldsymbol{X}_{(1)} - \bar{\boldsymbol{X}}\right)\right)^\top \\ \left(\boldsymbol{D}^{-\frac{1}{2}}\left(\boldsymbol{X}_{(2)} - \bar{\boldsymbol{X}}\right)\right)^\top \\ \vdots \\ \left(\boldsymbol{D}^{-\frac{1}{2}}\left(\boldsymbol{X}_{(n)} - \bar{\boldsymbol{X}}\right)\right)^\top \end{pmatrix} \\
&= \begin{pmatrix} \left(\boldsymbol{X}_{(1)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{D}^{-\frac{1}{2}} \\ \left(\boldsymbol{X}_{(2)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{D}^{-\frac{1}{2}} \\ \vdots \\ \left(\boldsymbol{X}_{(n)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{D}^{-\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_{(1)}^\top - \bar{\boldsymbol{X}}^\top \\ \boldsymbol{X}_{(2)}^\top - \bar{\boldsymbol{X}}^\top \\ \vdots \\ \boldsymbol{X}_{(n)}^\top - \bar{\boldsymbol{X}}^\top \end{pmatrix} \boldsymbol{D}^{-\frac{1}{2}} \\
&= \left(\boldsymbol{X} - \mathbf{1}\bar{\boldsymbol{X}}^\top\right) \boldsymbol{D}^{-\frac{1}{2}},
\end{aligned}$$

$$\mathbf{1} = \mathbf{1}_{n\times 1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{n\times 1}, \ \boldsymbol{D} = \operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}\right)\right) = \begin{pmatrix} s_{11} & & & \\ & s_{22} & & \\ & & \ddots & \\ & & & s_{pp} \end{pmatrix},$$

$$\boldsymbol{D}^{\frac{1}{2}} = \left[\operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}\right)\right)\right]^{\frac{1}{2}} = \operatorname{diag}\left(\left[\operatorname{diag}\left(\boldsymbol{S}\right)\right]^{\frac{1}{2}}\right) = \begin{pmatrix} \sqrt{s_{11}} & & & \\ & \sqrt{s_{22}} & & \\ & & \ddots & \\ & & & \sqrt{s_{pp}} \end{pmatrix},$$

$$\boldsymbol{D}^{-\frac{1}{2}} = \left[\operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}\right)\right)\right]^{-\frac{1}{2}} = \operatorname{diag}\left(\left[\operatorname{diag}\left(\boldsymbol{S}\right)\right]^{-\frac{1}{2}}\right) = \begin{pmatrix} 1/\sqrt{s_{11}} & & & \\ & 1/\sqrt{s_{22}} & & \\ & & \ddots & \\ & & & 1/\sqrt{s_{pp}} \end{pmatrix},$$

$$\begin{aligned}
\boldsymbol{X}_{(k)}^* &= \boldsymbol{D}^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right) \\
&= \begin{pmatrix} 1/\sqrt{s_{11}} & & & \\ & 1/\sqrt{s_{22}} & & \\ & & \ddots & \\ & & & 1/\sqrt{s_{pp}} \end{pmatrix} \begin{pmatrix} x_{k1} - \bar{x}_1 \\ x_{k2} - \bar{x}_2 \\ \vdots \\ x_{kp} - \bar{x}_p \end{pmatrix} \\
&= \begin{pmatrix} \left(x_{k1} - \bar{x}_1\right)/\sqrt{s_{11}} \\ \left(x_{k2} - \bar{x}_2\right)/\sqrt{s_{22}} \\ \vdots \\ \left(x_{kp} - \bar{x}_p\right)/\sqrt{s_{pp}} \end{pmatrix}.
\end{aligned}$$

To compare the 8 combinations, we summarize some useful results in Tables 2 and 3. There are some points in Tables 2 and 3 that are proved here.

Table 2: Results of (1), (2), (5), and (6).

	$\boldsymbol{X}$, classical (1), (2)	$\boldsymbol{X}$, robust (5), (6)
data matrix	$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_{(1)}^\top \\ \boldsymbol{X}_{(2)}^\top \\ \vdots \\ \boldsymbol{X}_{(n)}^\top \end{pmatrix}$	$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_{(1)}^\top \\ \boldsymbol{X}_{(2)}^\top \\ \vdots \\ \boldsymbol{X}_{(n)}^\top \end{pmatrix}$
number of used observations	n	M
sample used	$\boldsymbol{X}_{(1)}, \boldsymbol{X}_{(2)}, \cdots, \boldsymbol{X}_{(n)}$	$\boldsymbol{X}_{(1)}, \boldsymbol{X}_{(2)}, \cdots, \boldsymbol{X}_{(M)}$
kth observation	$\boldsymbol{X}_{(k)}$	$\boldsymbol{X}_{(k)}$
sample mean	$\bar{\boldsymbol{X}}^c = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{X}_{(k)}$	$\bar{\boldsymbol{X}}^r = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{X}_{(k)}$
sample covariance	$\boldsymbol{S}^c = \frac{1}{n-1}\sum_{k=1}^{n} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right)\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right)^\top$	$\boldsymbol{S}^r = \frac{1}{M-1}\sum_{k=1}^{M} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)^\top$
sample correlation	$\boldsymbol{R}^c$	$\boldsymbol{R}^r$
$\boldsymbol{D}$	$\boldsymbol{D}^c = \mathrm{diag}\left(\mathrm{diag}\left(\boldsymbol{S}^c\right)\right)$	$\boldsymbol{D}^r = \mathrm{diag}\left(\mathrm{diag}\left(\boldsymbol{S}^r\right)\right)$
scaledX	$\boldsymbol{X}^{*c} = \left(\boldsymbol{X} - \boldsymbol{1}\left(\bar{\boldsymbol{X}}^c\right)^\top\right)\left(\boldsymbol{D}^c\right)^{-\frac{1}{2}} = \begin{pmatrix} \left(\boldsymbol{X}_{(1)}^{*c}\right)^\top \\ \left(\boldsymbol{X}_{(2)}^{*c}\right)^\top \\ \vdots \\ \left(\boldsymbol{X}_{(n)}^{*c}\right)^\top \end{pmatrix}$	$\boldsymbol{X}^{*r} = \left(\boldsymbol{X} - \boldsymbol{1}\left(\bar{\boldsymbol{X}}^r\right)^\top\right)\left(\boldsymbol{D}^r\right)^{-\frac{1}{2}} = \begin{pmatrix} \left(\boldsymbol{X}_{(1)}^{*r}\right)^\top \\ \left(\boldsymbol{X}_{(2)}^{*r}\right)^\top \\ \vdots \\ \left(\boldsymbol{X}_{(n)}^{*r}\right)^\top \end{pmatrix}$
kth scaled observation	$\boldsymbol{X}_{(k)}^{*c} = \left(\boldsymbol{D}^c\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right)$	$\boldsymbol{X}_{(k)}^{*r} = \left(\boldsymbol{D}^r\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)$
sample mean of scaledX	$\bar{\boldsymbol{X}}^{*c} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{X}_{(k)}^{*c} = \boldsymbol{0}$	$\bar{\boldsymbol{X}}^{*r} = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{X}_{(k)}^{*r} = \boldsymbol{0}$ (a)
cov(scaledX)	$\boldsymbol{S}^{*c} = \boldsymbol{R}^c = \frac{1}{n-1}\sum_{k=1}^{n}\left(\boldsymbol{X}_{(k)}^{*c} - \bar{\boldsymbol{X}}^{*c}\right)\left(\boldsymbol{X}_{(k)}^{*c} - \bar{\boldsymbol{X}}^{*c}\right)^\top = \frac{1}{n-1}\sum_{k=1}^{n}\boldsymbol{X}_{(k)}^{*c}\left(\boldsymbol{X}_{(k)}^{*c}\right)^\top = \left(\boldsymbol{D}^c\right)^{-\frac{1}{2}}\boldsymbol{S}^c\left(\boldsymbol{D}^c\right)^{-\frac{1}{2}}$	$\boldsymbol{S}^{*r} = \boldsymbol{R}^r = \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{X}_{(k)}^{*r} - \bar{\boldsymbol{X}}^{*r}\right)\left(\boldsymbol{X}_{(k)}^{*r} - \bar{\boldsymbol{X}}^{*r}\right)^\top = \frac{1}{M-1}\sum_{k=1}^{M}\boldsymbol{X}_{(k)}^{*r}\left(\boldsymbol{X}_{(k)}^{*r}\right)^\top = \left(\boldsymbol{D}^r\right)^{-\frac{1}{2}}\boldsymbol{S}^r\left(\boldsymbol{D}^r\right)^{-\frac{1}{2}}$ (b)

Table 3: Results of (3), (4), (7), and (8).

	$\boldsymbol{Y}$, classical (3), (4)	$\boldsymbol{Y}$, robust (7), (8)
data matrix	$\boldsymbol{Y} = \text{scale}\left(\boldsymbol{X}\right) = \boldsymbol{X}^{*c}$ $= \left(\boldsymbol{X} - \mathbf{1}\left(\bar{\boldsymbol{X}}^{c}\right)^{\top}\right)\left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}$	$\boldsymbol{Y} = \text{scale}\left(\boldsymbol{X}\right) = \boldsymbol{X}^{*c}$ $= \left(\boldsymbol{X} - \mathbf{1}\left(\bar{\boldsymbol{X}}^{c}\right)^{\top}\right)\left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}$
number of used observations	n	M
sample used	$\boldsymbol{Y}_{(1)}, \boldsymbol{Y}_{(2)}, \cdots, \boldsymbol{Y}_{(n)}$	$\boldsymbol{Y}_{(1)}, \boldsymbol{Y}_{(2)}, \cdots, \boldsymbol{Y}_{(M)}$
kth observation	$\boldsymbol{Y}_{(k)} = \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^{c}\right) = \boldsymbol{X}^{*c}_{(k)}$	$\boldsymbol{Y}_{(k)} = \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^{c}\right) = \boldsymbol{X}^{*c}_{(k)}$
sample mean	$\bar{\boldsymbol{Y}}^{c} = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{Y}_{(k)} = \mathbf{0}$ (c)	$\bar{\boldsymbol{Y}}^{r} = \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{Y}_{(k)}$ $= \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\bar{\boldsymbol{X}}^{r} - \bar{\boldsymbol{X}}^{c}\right)$ (f)
sample covariance	$\tilde{\boldsymbol{S}}^{c} = \frac{1}{n-1}\sum_{k=1}^{n}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{c}\right)\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{c}\right)^{\top}$ $= \frac{1}{n-1}\sum_{k=1}^{n}\boldsymbol{Y}_{(k)}\boldsymbol{Y}_{(k)}^{\top}$	$\tilde{\boldsymbol{S}}^{r} = \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)^{\top}$
sample correlation	$\tilde{\boldsymbol{R}}^{c}$	$\tilde{\boldsymbol{R}}^{r}$
$\boldsymbol{D}$	$\tilde{\boldsymbol{D}}^{c} = \text{diag}\left(\text{diag}\left(\tilde{\boldsymbol{S}}^{c}\right)\right) = \boldsymbol{E}$ (d)	$\tilde{\boldsymbol{D}}^{r} = \text{diag}\left(\text{diag}\left(\tilde{\boldsymbol{S}}^{r}\right)\right)$
scaledX	$\boldsymbol{Y}^{*c} = \left(\boldsymbol{Y} - \mathbf{1}\left(\bar{\boldsymbol{Y}}^{c}\right)^{\top}\right)\left(\tilde{\boldsymbol{D}}^{c}\right)^{-\frac{1}{2}}$ $= \begin{pmatrix} \left(\boldsymbol{Y}^{*c}_{(1)}\right)^{\top} \\ \left(\boldsymbol{Y}^{*c}_{(2)}\right)^{\top} \\ \vdots \\ \left(\boldsymbol{Y}^{*c}_{(n)}\right)^{\top} \end{pmatrix}$	$\boldsymbol{Y}^{*r} = \left(\boldsymbol{Y} - \mathbf{1}\left(\bar{\boldsymbol{Y}}^{r}\right)^{\top}\right)\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}$ $= \begin{pmatrix} \left(\boldsymbol{Y}^{*r}_{(1)}\right)^{\top} \\ \left(\boldsymbol{Y}^{*r}_{(2)}\right)^{\top} \\ \vdots \\ \left(\boldsymbol{Y}^{*r}_{(n)}\right)^{\top} \end{pmatrix}$
kth scaled observation	$\boldsymbol{Y}^{*c}_{(k)} = \left(\tilde{\boldsymbol{D}}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{c}\right) = \boldsymbol{Y}_{(k)}$ (e)	$\boldsymbol{Y}^{*r}_{(k)} = \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)$
sample mean of scaledX	$\bar{\boldsymbol{Y}}^{*c} = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{Y}^{*c}_{(k)} = \mathbf{0}$	$\bar{\boldsymbol{Y}}^{*r} = \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{Y}^{*r}_{(k)} = \mathbf{0}$ (g)
cov(scaledX)	$\tilde{\boldsymbol{S}}^{*c} = \tilde{\boldsymbol{R}}^{c}$ $= \frac{1}{n-1}\sum_{k=1}^{n}\left(\boldsymbol{Y}^{*c}_{(k)} - \bar{\boldsymbol{Y}}^{*c}\right)\left(\boldsymbol{Y}^{*c}_{(k)} - \bar{\boldsymbol{Y}}^{*c}\right)^{\top}$ $= \frac{1}{n-1}\sum_{k=1}^{n}\boldsymbol{Y}^{*c}_{(k)}\left(\boldsymbol{Y}^{*c}_{(k)}\right)^{\top}$ $= \left(\tilde{\boldsymbol{D}}^{c}\right)^{-\frac{1}{2}}\tilde{\boldsymbol{S}}^{c}\left(\tilde{\boldsymbol{D}}^{c}\right)^{-\frac{1}{2}}$	$\tilde{\boldsymbol{S}}^{*r} = \tilde{\boldsymbol{R}}^{r}$ $= \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{Y}^{*r}_{(k)} - \bar{\boldsymbol{Y}}^{*r}\right)\left(\boldsymbol{Y}^{*r}_{(k)} - \bar{\boldsymbol{Y}}^{*r}\right)^{\top}$ $= \frac{1}{M-1}\sum_{k=1}^{M}\boldsymbol{Y}^{*r}_{(k)}\left(\boldsymbol{Y}^{*r}_{(k)}\right)^{\top}$ $= \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\tilde{\boldsymbol{S}}^{r}\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}$ (h)

Proof. (a)

$$\begin{aligned}\bar{\boldsymbol{X}}^{*r} &= \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{X}_{(k)}^{*r}\\ &= \frac{1}{M}\sum_{k=1}^{M}\left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)\\ &= \left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\frac{1}{M}\sum_{k=1}^{M}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)\\ &= \left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\left(\bar{\boldsymbol{X}}^{r}-\bar{\boldsymbol{X}}^{r}\right)\\ &= \mathbf{0}.\end{aligned}$$

(b)

$$\begin{aligned}\boldsymbol{S}^{*r} &= \boldsymbol{R}^{r}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{X}_{(k)}^{*r}-\bar{\boldsymbol{X}}^{*r}\right)\left(\boldsymbol{X}_{(k)}^{*r}-\bar{\boldsymbol{X}}^{*r}\right)^{\top}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\boldsymbol{X}_{(k)}^{*r}\left(\boldsymbol{X}_{(k)}^{*r}\right)^{\top}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)^{\top}\left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\\ &= \left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\left[\frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{r}\right)^{\top}\right]\left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\\ &= \left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}\boldsymbol{S}^{r}\left(\boldsymbol{D}^{r}\right)^{-\frac{1}{2}}.\end{aligned}$$

(c)

$$\begin{aligned}\bar{\boldsymbol{Y}}^{c} &= \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{Y}_{(k)}\\ &= \frac{1}{n}\sum_{k=1}^{n}\left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{c}\right)\\ &= \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\frac{1}{n}\sum_{k=1}^{n}\left(\boldsymbol{X}_{(k)}-\bar{\boldsymbol{X}}^{c}\right)\\ &= \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\bar{\boldsymbol{X}}^{c}-\bar{\boldsymbol{X}}^{c}\right)\\ &= \mathbf{0}.\end{aligned}$$

(d) By Theorem 3.1, we have $\tilde{\boldsymbol{S}}^{c} = \boldsymbol{R}^{c}$, thus

$$\begin{aligned}\tilde{\boldsymbol{D}}^{c} &= \operatorname{diag}\left(\operatorname{diag}\left(\tilde{\boldsymbol{S}}^{c}\right)\right)\\ &= \operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{R}^{c}\right)\right)\\ &= \operatorname{diag}\left((1,1,\cdots,1)\right)\\ &= \boldsymbol{E}.\end{aligned}$$

(e)

$$\begin{aligned}\boldsymbol{Y}^{*c}_{(k)} &= \left(\tilde{\boldsymbol{D}}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{c}\right)\\ &= \boldsymbol{E}^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \boldsymbol{0}\right)\\ &= \boldsymbol{Y}_{(k)}.\end{aligned}$$

(f)

$$\begin{aligned}\bar{\boldsymbol{Y}}^{r} &= \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{Y}_{(k)}\\ &= \frac{1}{M}\sum_{k=1}^{M}\left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^{c}\right)\\ &= \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\frac{1}{M}\sum_{k=1}^{M}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^{c}\right)\\ &= \left(\boldsymbol{D}^{c}\right)^{-\frac{1}{2}}\left(\bar{\boldsymbol{X}}^{r} - \bar{\boldsymbol{X}}^{c}\right).\end{aligned}$$

(g)

$$\begin{aligned}\bar{\boldsymbol{Y}}^{*r} &= \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{Y}^{*r}_{(k)}\\ &= \frac{1}{M}\sum_{k=1}^{M}\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)\\ &= \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\frac{1}{M}\sum_{k=1}^{M}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)\\ &= \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\left(\bar{\boldsymbol{Y}}^{r} - \bar{\boldsymbol{Y}}^{r}\right)\\ &= \boldsymbol{0}.\end{aligned}$$

(h)

$$\begin{aligned}\tilde{\boldsymbol{S}}^{*r} &= \tilde{\boldsymbol{R}}^{r}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{Y}^{*r}_{(k)} - \bar{\boldsymbol{Y}}^{*r}\right)\left(\boldsymbol{Y}^{*r}_{(k)} - \bar{\boldsymbol{Y}}^{*r}\right)^{\top}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\boldsymbol{Y}^{*r}_{(k)}\left(\boldsymbol{Y}^{*r}_{(k)}\right)^{\top}\\ &= \frac{1}{M-1}\sum_{k=1}^{M}\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)^{\top}\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\\ &= \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\left[\frac{1}{M-1}\sum_{k=1}^{M}\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^{r}\right)^{\top}\right]\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\\ &= \left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}\tilde{\boldsymbol{S}}^{r}\left(\tilde{\boldsymbol{D}}^{r}\right)^{-\frac{1}{2}}.\end{aligned}$$

Define the multiplication of two vectors by

$$\begin{aligned}\mathbf{x}\cdot\mathbf{y} &= (x_1, x_2, \cdots, x_p)\cdot(y_1, y_2, \cdots, y_p)\\ &= (x_1y_1, x_2y_2, \cdots, x_py_p)\,.\end{aligned}$$

To prove Theorem 3.1, we need the following lemma.

Lemma 3.1 *Let*

$$\mathbf{\Lambda}_1 = \begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p\end{pmatrix},\ \mathbf{\Lambda}_2 = \begin{pmatrix}\mu_1 & & & \\ & \mu_2 & & \\ & & \ddots & \\ & & & \mu_p\end{pmatrix},\ \boldsymbol{A} = (a_{ij})_{p\times p}\,.$$

Then

$$\begin{aligned}\operatorname{diag}(\mathbf{\Lambda}_1\boldsymbol{A}\mathbf{\Lambda}_2) &= \operatorname{diag}(\mathbf{\Lambda}_1)\cdot\operatorname{diag}(\boldsymbol{A})\cdot\operatorname{diag}(\mathbf{\Lambda}_2)\,,\\ \operatorname{diag}(\mathbf{\Lambda}_1\mathbf{\Lambda}_2) &= \operatorname{diag}(\mathbf{\Lambda}_1)\cdot\operatorname{diag}(\mathbf{\Lambda}_2)\,.\end{aligned}$$

Proof.

$$\begin{aligned}\mathbf{\Lambda}_1\boldsymbol{A}\mathbf{\Lambda}_2 &= \begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p\end{pmatrix}\begin{pmatrix}a_{11} & a_{12} & \cdots & a_{1p}\\ a_{21} & a_{22} & \cdots & a_{2p}\\ \vdots & \vdots & \ddots & \vdots\\ a_{p1} & a_{p2} & \cdots & a_{pp}\end{pmatrix}\begin{pmatrix}\mu_1 & & & \\ & \mu_2 & & \\ & & \ddots & \\ & & & \mu_p\end{pmatrix}\\ &= \begin{pmatrix}\lambda_1a_{11}\mu_1 & * & \cdots & *\\ * & \lambda_2a_{22}\mu_2 & \cdots & *\\ \vdots & \vdots & \ddots & \vdots\\ * & * & \cdots & \lambda_pa_{pp}\mu_p\end{pmatrix},\end{aligned}$$

$$\begin{aligned}\operatorname{diag}(\mathbf{\Lambda}_1\boldsymbol{A}\mathbf{\Lambda}_2) &= (\lambda_1a_{11}\mu_1, \lambda_2a_{22}\mu_2, \cdots, \lambda_pa_{pp}\mu_p)\\ &= (\lambda_1, \lambda_2, \cdots, \lambda_p)\cdot(a_{11}, a_{22}, \cdots, a_{pp})\cdot(\mu_1, \mu_2, \cdots, \mu_p)\\ &= \operatorname{diag}(\mathbf{\Lambda}_1)\cdot\operatorname{diag}(\boldsymbol{A})\cdot\operatorname{diag}(\mathbf{\Lambda}_2)\,.\end{aligned}$$

$$\mathbf{\Lambda}_1\mathbf{\Lambda}_2 = \begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p\end{pmatrix}\begin{pmatrix}\mu_1 & & & \\ & \mu_2 & & \\ & & \ddots & \\ & & & \mu_p\end{pmatrix} = \begin{pmatrix}\lambda_1\mu_1 & & & \\ & \lambda_2\mu_2 & & \\ & & \ddots & \\ & & & \lambda_p\mu_p\end{pmatrix},$$

$$\begin{aligned}\operatorname{diag}(\mathbf{\Lambda}_1\mathbf{\Lambda}_2) &= (\lambda_1\mu_1, \lambda_2\mu_2, \cdots, \lambda_p\mu_p)\\ &= (\lambda_1, \lambda_2, \cdots, \lambda_p)\cdot(\mu_1, \mu_2, \cdots, \mu_p)\\ &= \operatorname{diag}(\mathbf{\Lambda}_1)\cdot\operatorname{diag}(\mathbf{\Lambda}_2)\,.\end{aligned}$$

□

Theorem 3.1 *(a)* $\boldsymbol{R}^c = \tilde{\boldsymbol{S}}^c = \tilde{\boldsymbol{R}}^c$. *(b)* $\boldsymbol{R}^r = \tilde{\boldsymbol{R}}^r$.

Proof. (a) We prove $\boldsymbol{R}^c = \tilde{\boldsymbol{S}}^c$ first. Since $\boldsymbol{Y}_{(k)} = (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right) = \boldsymbol{X}^{*c}_{(k)}$,

$$\begin{aligned}\boldsymbol{R}^c &= \frac{1}{n-1}\sum_{k=1}^{n} \boldsymbol{X}^{*c}_{(k)} \left(\boldsymbol{X}^{*c}_{(k)}\right)^\top \\ &= \frac{1}{n-1}\sum_{k=1}^{n} \boldsymbol{Y}_{(k)} \boldsymbol{Y}^\top_{(k)} \\ &= \tilde{\boldsymbol{S}}^c.\end{aligned}$$

Next we prove $\tilde{\boldsymbol{R}}^c = \tilde{\boldsymbol{S}}^c$. From Table 3 (e), we have $\boldsymbol{Y}^{*c}_{(k)} = \boldsymbol{Y}_{(k)}$, thus

$$\begin{aligned}\tilde{\boldsymbol{R}}^c &= \frac{1}{n-1}\sum_{k=1}^{n} \boldsymbol{Y}^{*c}_{(k)} \left(\boldsymbol{Y}^{*c}_{(k)}\right)^\top \\ &= \frac{1}{n-1}\sum_{k=1}^{n} \boldsymbol{Y}_{(k)} \boldsymbol{Y}^\top_{(k)} \\ &= \tilde{\boldsymbol{S}}^c.\end{aligned}$$

(b) We have

$$\begin{aligned}\boldsymbol{R}^r = \boldsymbol{S}^{*r} &= \frac{1}{M-1}\sum_{k=1}^{M} \boldsymbol{X}^{*r}_{(k)} \left(\boldsymbol{X}^{*r}_{(k)}\right)^\top, \\ \tilde{\boldsymbol{R}}^r = \tilde{\boldsymbol{S}}^{*r} &= \frac{1}{M-1}\sum_{k=1}^{M} \boldsymbol{Y}^{*r}_{(k)} \left(\boldsymbol{Y}^{*r}_{(k)}\right)^\top.\end{aligned}$$

To prove that $\boldsymbol{R}^r = \tilde{\boldsymbol{R}}^r$, it suffices to prove that

$$\boldsymbol{X}^{*r}_{(k)} = \boldsymbol{Y}^{*r}_{(k)}. \tag{3.1}$$

Write

$$\begin{aligned}\boldsymbol{X}^{*r}_{(k)} &= (\boldsymbol{D}^r)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right), \\ \boldsymbol{Y}^{*r}_{(k)} &= \left(\tilde{\boldsymbol{D}}^r\right)^{-\frac{1}{2}} \left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^r\right), \\ \boldsymbol{Y}_{(k)} &= (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right),\end{aligned}$$

from Table 3 (f), we have

$$\bar{\boldsymbol{Y}}^r = (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\bar{\boldsymbol{X}}^r - \bar{\boldsymbol{X}}^c\right).$$

Thus

$$\begin{aligned}\boldsymbol{Y}^{*r}_{(k)} &= \left(\tilde{\boldsymbol{D}}^r\right)^{-\frac{1}{2}} \left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^r\right) \\ &= \left(\tilde{\boldsymbol{D}}^r\right)^{-\frac{1}{2}} \left[(\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^c\right) - (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\bar{\boldsymbol{X}}^r - \bar{\boldsymbol{X}}^c\right)\right] \\ &= \left(\tilde{\boldsymbol{D}}^r\right)^{-\frac{1}{2}} (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right).\end{aligned}$$

To prove (3.1), it suffices to prove

$$\left(\tilde{\boldsymbol{D}}^r\right)^{-\frac{1}{2}} (\boldsymbol{D}^c)^{-\frac{1}{2}} = (\boldsymbol{D}^r)^{-\frac{1}{2}},$$

which reduces to prove

$$\tilde{\boldsymbol{D}}^r \boldsymbol{D}^c = \boldsymbol{D}^r,$$

that is,

$$\operatorname{diag}\left(\operatorname{diag}\left(\tilde{\boldsymbol{S}}^r\right)\right) \cdot \operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}^c\right)\right) = \operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}^r\right)\right),$$

which is equivalent to

$$\operatorname{diag}\left(\tilde{\boldsymbol{S}}^r\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^c\right) = \operatorname{diag}\left(\boldsymbol{S}^r\right).$$

We have just proved

$$\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^r = (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right),$$

and thus

$$\begin{aligned}
\tilde{\boldsymbol{S}}^r &= \frac{1}{M-1}\sum_{k=1}^{M} \left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^r\right)\left(\boldsymbol{Y}_{(k)} - \bar{\boldsymbol{Y}}^r\right)^\top \\
&= \frac{1}{M-1}\sum_{k=1}^{M} (\boldsymbol{D}^c)^{-\frac{1}{2}} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)^\top (\boldsymbol{D}^c)^{-\frac{1}{2}} \\
&= (\boldsymbol{D}^c)^{-\frac{1}{2}} \left[\frac{1}{M-1}\sum_{k=1}^{M} \left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}^r\right)^\top\right] (\boldsymbol{D}^c)^{-\frac{1}{2}} \\
&= (\boldsymbol{D}^c)^{-\frac{1}{2}} \boldsymbol{S}^r (\boldsymbol{D}^c)^{-\frac{1}{2}}.
\end{aligned}$$

Consequently,

$$\begin{aligned}
\operatorname{diag}\left(\tilde{\boldsymbol{S}}^r\right) &= \operatorname{diag}\left((\boldsymbol{D}^c)^{-\frac{1}{2}} \boldsymbol{S}^r (\boldsymbol{D}^c)^{-\frac{1}{2}}\right) \\
&= \operatorname{diag}\left((\boldsymbol{D}^c)^{-\frac{1}{2}}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \cdot \operatorname{diag}\left((\boldsymbol{D}^c)^{-\frac{1}{2}}\right) \text{ (Lemma 3.1)} \\
&= \operatorname{diag}\left((\boldsymbol{D}^c)^{-\frac{1}{2}}\right) \cdot \operatorname{diag}\left((\boldsymbol{D}^c)^{-\frac{1}{2}}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \\
&= \operatorname{diag}\left((\boldsymbol{D}^c)^{-1}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \text{ (Lemma 3.1).}
\end{aligned}$$

Finally,

$$\begin{aligned}
\operatorname{diag}\left(\tilde{\boldsymbol{S}}^r\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^c\right) &= \operatorname{diag}\left((\boldsymbol{D}^c)^{-1}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^c\right) \\
&= \operatorname{diag}\left((\boldsymbol{D}^c)^{-1}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \cdot \operatorname{diag}\left(\boldsymbol{D}^c\right) \\
&= \operatorname{diag}\left((\boldsymbol{D}^c)^{-1}\right) \cdot \operatorname{diag}\left(\boldsymbol{D}^c\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \\
&= \operatorname{diag}\left(\boldsymbol{E}\right) \cdot \operatorname{diag}\left(\boldsymbol{S}^r\right) \text{ (Lemma 3.1)} \\
&= \operatorname{diag}\left(\boldsymbol{S}^r\right).
\end{aligned}$$

The proof is complete. □

Table 4: The score of the kth observation.

	regression	Bartlett
cor = FALSE	$\tilde{\boldsymbol{f}}_{(k)} = \hat{\boldsymbol{L}}^\top \boldsymbol{S}^{-1}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right)$	$\hat{\boldsymbol{f}}_{(k)} = \left(\hat{\boldsymbol{L}}^\top \hat{\boldsymbol{\Psi}}^{-1}\hat{\boldsymbol{L}}\right)^{-1}\hat{\boldsymbol{L}}^\top \hat{\boldsymbol{\Psi}}^{-1}\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right)$
cor = TRUE	$\tilde{\boldsymbol{f}}^*_{(k)} = \hat{\boldsymbol{L}}^{*\top} \boldsymbol{R}^{-1}\boldsymbol{X}^*_{(k)}$	$\hat{\boldsymbol{f}}^*_{(k)} = \left(\hat{\boldsymbol{L}}^{*\top} \hat{\boldsymbol{\Psi}}^{*-1}\hat{\boldsymbol{L}}^*\right)^{-1}\hat{\boldsymbol{L}}^{*\top} \hat{\boldsymbol{\Psi}}^{*-1}\boldsymbol{X}^*_{(k)}$

The score of the kth observation is summarized in Table 4. The scores method can be "regression" or "Bartlett". The running matrix can be the covariance matrix ($\boldsymbol{S}$) or the correlation matrix ($\boldsymbol{R}$).

If we define the scoring coefficient $\boldsymbol{S}_c$ by

$$\boldsymbol{S}_c = \begin{cases} \hat{\boldsymbol{L}}^\top \boldsymbol{S}^{-1}, & \text{cor = FALSE, scoresMethod = "regression",} \\ \left(\hat{\boldsymbol{L}}^\top \hat{\boldsymbol{\Psi}}^{-1}\hat{\boldsymbol{L}}\right)^{-1}\hat{\boldsymbol{L}}^\top \hat{\boldsymbol{\Psi}}^{-1}, & \text{cor = FALSE, scoresMethod = "Bartlett",} \\ \hat{\boldsymbol{L}}^{*\top} \boldsymbol{R}^{-1}, & \text{cor = TRUE, scoresMethod = "regression",} \\ \left(\hat{\boldsymbol{L}}^{*\top} \hat{\boldsymbol{\Psi}}^{*-1}\hat{\boldsymbol{L}}^*\right)^{-1}\hat{\boldsymbol{L}}^{*\top} \hat{\boldsymbol{\Psi}}^{*-1}, & \text{cor = TRUE, scoresMethod = "Bartlett",} \end{cases}$$

then the score of the kth observation simplifies to

$$\boldsymbol{f}_{(k)} = \begin{cases} \boldsymbol{S}_c\left(\boldsymbol{X}_{(k)} - \bar{\boldsymbol{X}}\right), & \text{cor = FALSE,} \\ \boldsymbol{S}_c\boldsymbol{X}^*_{(k)}, & \text{cor = TRUE.} \end{cases}$$

If cor = FALSE, then the score matrix is

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}_{(1)}^\top \\ \boldsymbol{f}_{(2)}^\top \\ \vdots \\ \boldsymbol{f}_{(n)}^\top \end{pmatrix} = \begin{pmatrix} \left(\boldsymbol{X}_{(1)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{S}_c^\top \\ \left(\boldsymbol{X}_{(2)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{S}_c^\top \\ \vdots \\ \left(\boldsymbol{X}_{(n)} - \bar{\boldsymbol{X}}\right)^\top \boldsymbol{S}_c^\top \end{pmatrix} = \begin{pmatrix} \left(\boldsymbol{X}_{(1)} - \bar{\boldsymbol{X}}\right)^\top \\ \left(\boldsymbol{X}_{(2)} - \bar{\boldsymbol{X}}\right)^\top \\ \vdots \\ \left(\boldsymbol{X}_{(n)} - \bar{\boldsymbol{X}}\right)^\top \end{pmatrix} \boldsymbol{S}_c^\top = \left(\boldsymbol{X} - \boldsymbol{1}\bar{\boldsymbol{X}}^\top\right)\boldsymbol{S}_c^\top.$$

If cor = TRUE, then the score matrix is

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}_{(1)}^\top \\ \boldsymbol{f}_{(2)}^\top \\ \vdots \\ \boldsymbol{f}_{(n)}^\top \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^{*\top}_{(1)}\boldsymbol{S}_c^\top \\ \boldsymbol{X}^{*\top}_{(2)}\boldsymbol{S}_c^\top \\ \vdots \\ \boldsymbol{X}^{*\top}_{(n)}\boldsymbol{S}_c^\top \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^{*\top}_{(1)} \\ \boldsymbol{X}^{*\top}_{(2)} \\ \vdots \\ \boldsymbol{X}^{*\top}_{(n)} \end{pmatrix} \boldsymbol{S}_c^\top = \boldsymbol{X}^*\boldsymbol{S}_c^\top,$$

where $\boldsymbol{X}^* = \left(\boldsymbol{X} - \boldsymbol{1}\bar{\boldsymbol{X}}^\top\right)\boldsymbol{D}^{-\frac{1}{2}}$. If we define

$$\text{scaledX} = \begin{cases} \boldsymbol{X} - \boldsymbol{1}\bar{\boldsymbol{X}}^\top, & \text{cor = FALSE,} \\ \boldsymbol{X}^*, & \text{cor = TRUE,} \end{cases}$$

then

$$\boldsymbol{F} = \text{scaledX} \cdot \boldsymbol{S}_c^\top. \tag{3.2}$$

Theorem 3.2 *The running matrices ($\boldsymbol{R}$), eigenvalues (λ), loadings ($\boldsymbol{L}$), uniquenesses ($\boldsymbol{\Psi}$), scoring coefficients ($\boldsymbol{S}_c$), scaled data matrices (scaledX), score matrices ($\boldsymbol{F}$) are the same for*

(a) combinations (2) and (4),

(b) combinations (6) and (8).

Proof. If the running matrices ($\boldsymbol{R}$) are the same, then the eigenvalues (λ), loadings ($\boldsymbol{L}$), uniquenesses ($\boldsymbol{\Psi}$) are the same. If $\boldsymbol{R}, \boldsymbol{L}, \boldsymbol{\Psi}$ are the same, then by the definition of scoring coefficient $\boldsymbol{S}_c$, we know that $\boldsymbol{S}_c$ are the same. If the scaled data matrices (scaledX) are the same, then by (3.2), the score matrices ($\boldsymbol{F}$) are the same. Thus it suffices to show that $\boldsymbol{R}$ and scaledX are the same.

(a) The running matrices $\boldsymbol{R}^c = \tilde{\boldsymbol{R}}^c$ by Theorem 3.1. From Table 3, we see that

$$\boldsymbol{Y}^{*c}_{(k)} = \boldsymbol{Y}_{(k)} = \boldsymbol{X}^{*c}_{(k)},$$

thus the scaledX are the same.

(b) The running matrices $\boldsymbol{R}^r = \tilde{\boldsymbol{R}}^r$ by Theorem 3.1. By (3.1), we have $\boldsymbol{X}^{*r}_{(k)} = \boldsymbol{Y}^{*r}_{(k)}$, thus the scaledX are the same. □

3.3. Example: hbk data

In this subsection, a data set `hbk` is used to show the base functionalities of the robust factor analysis solution. The Hawkins, Bradu and Kass data set `hbk` is from the package **robustbase** consists of 75 observations in 4 dimensions (one response and three explanatory variables). The first 10 observations are bad leverage points, and the next four points are good leverage points (i.e., their `x` are outlying, but the corresponding `y` fit the model quite well). We will consider only the X-part of the data.

Robust factor analysis is represented by the class `FaCov` which inherits from class `Fa` by class `FaRobust` of distance 2, and uses all slots and methods defined from `Fa`. The function `FaCov()` serves as a constructor (generating function) of the class. It can be called either by providing a data frame or matrix or a formula with no response variable, referring only to numeric variables. The usage of the default method of `FaCov` is:

```
## Loading required package:  rrcov
## Loading required package:  robustbase
## Scalable Robust Estimators with High Breakdown Point (version 1.7-2)
```

```
FaCov(x, factors = 2, cor = FALSE, cov.control = rrcov::CovControlMcd(),
      method = c("mle", "pca", "pfa"),
      scoresMethod = c("none", "regression", "Bartlett"), ...)
```

where `x` is a numeric matrix or an object that can be coerced to a numeric matrix. `factors` is the number of factors to be fitted. `cor` is a logical value indicating whether the calculation should use the correlation matrix (`cor = TRUE`) or the covariance matrix (`cor = FALSE`). `cov.control` specifies which covariance estimator to use by providing a `CovControl-class` object. The default is `CovControlMcd-class` which will indirectly call `CovMcd`. If `cov.control`

= NULL is specified, the classical estimates will be used by calling CovClassic. method is the method of factor analysis, one of "mle" (the default), "pca", and "pfa". scoresMethod specifies which type of scores to produce. The default is "none", "regression" gives Thompson's scores, and "Bartlett" gives Bartlett's weighted least-squares scores (Johnson and Wichern 2007; Xue and Chen 2007). The usage of the formula interface of FaCov is:

```
FaCov(formula, data = NULL, factors = 2, cor = FALSE,
      method = "mle", scoresMethod = "none", ...)
```

where formula is a formula with no response variable, referring only to numeric variables. data is an optional data frame containing the variables in the formula. Classical factor analysis is represented by the class FaClassic which inherits directly from Fa, and uses all slots and methods defined there. The function FaClassic() serves as a constructor (generating function) of the class. It also has its default method and formula interface as FaCov().

The code lines

```
  ##
  ## faCovPcaRegMcd is obtained from FaCov.default
  ##
  faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
                         scoresMethod = "regression", cov.control = rrcov::CovControlMcd())
faCovPcaRegMcd
```

generate an object faCovPcaRegMcd of class FaCov. In fact, it is equivalent to use the formula interface

```
  faCovForPcaRegMcd = FaCov(~., data = as.data.frame(hbk.x),
                            factors = 2, method = "pca", scoresMethod = "regression",
                            cov.control = rrcov::CovControlMcd())
```

That is faCovForPcaRegMcd = faCovPcaRegMcd. Type

```
  class(faCovPcaRegMcd)

## [1] "FaCov"
## attr(,"package")
## [1] "robustfa"
```

We see that the class FaCov is defined in the package **robustfa**. For an object obj of class Fa, we have

obj = print(obj) = myFaPrint(obj).

Here print() is a S4 generic function, while myFaPrint() is an S3 function acting as a function definition for setMethod of the generic function print.

```
  print(faCovPcaRegMcd)

## [1] "Call:\n FaCov(x = hbk.x, factors = 2, cov.control = rrcov::CovControlMcd(),  \n"
## [2] "Call:\n     method = \"pca\", scoresMethod = \"regression\") \n"
## [1] "Standard deviations:\n 1.39107203014516"
## [2] "Standard deviations:\n 1.26173192309765"
## [3] "Standard deviations:\n 1.17062612110883"
## [1] "Loadings:\n -0.00415024202396914"
## [2] "Loadings:\n 1.18079786613697"
## [3] "Loadings:\n 0.64616966781657"
## [4] "Loadings:\n 1.2079697328235"
## [5] "Loadings:\n -0.124696507700608"
## [6] "Loadings:\n 0.490380137828612"
```

From Figure 2 we see that `summary()` generates an object of class `SummaryFa` which has its own `print()` method.

```
  summaryFaCovPcaRegMcd = summary(faCovPcaRegMcd);
summaryFaCovPcaRegMcd

## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaCov"
## Slot "call":
## FaCov(x = hbk.x, factors = 2, cov.control = rrcov::CovControlMcd(),
##     method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##         Factor1    Factor2
## X1 -0.004150242  1.2079697
## X2  1.180797866 -0.1246965
## X3  0.646169668  0.4903801
##
## Slot "communality":
##        X1        X2        X3
## 1.4592081 1.4098328 0.6580079
##
## Slot "uniquenesses":
##        X1        X2        X3
## 0.1935459 0.2724395 0.9043802
##
## Slot "cor":
## [1] FALSE
```

```
##
## Slot "covariance":
##            X1         X2        X3
## X1 1.65275399 0.07409893 0.1713059
## X2 0.07409893 1.68227229 0.2054714
## X3 0.17130591 0.20547136 1.5623881
##
## Slot "correlation":
##            X1         X2        X3
## X1 1.00000000 0.04443853 0.1066041
## X2 0.04443853 1.00000000 0.1267385
## X3 0.10660407 0.12673854 1.0000000
##
## Slot "usedMatrix":
##            X1         X2        X3
## X1 1.65275399 0.07409893 0.1713059
## X2 0.07409893 1.68227229 0.2054714
## X3 0.17130591 0.20547136 1.5623881
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##        Factor1     Factor2
## 1  20.13910700 10.41054843
## 2  20.97706402 10.01367633
## 3  21.39809904 11.44059122
## 4  22.54842888 10.87239607
## 5  22.05589849 11.05891277
## 6  20.92380674 11.02737636
## 7  21.24256223 10.72056381
## 8  20.31926457 10.39535479
## 9  21.79701349 10.66168792
## 10 20.92614819 10.33613015
```

```
## 11 25.23594154 12.15088328
## 12 25.17211484 13.50577832
## 13 26.16372927 12.33480781
## 14 31.54346390 10.53262812
## 15  0.75687531  1.27562475
## 16 -0.28837150  0.70092910
## 17 -0.51292895 -1.44572060
## 18 -0.06627823  0.64585062
## 19  0.76392511 -0.69909928
## 20  1.14234504  1.01958952
## 21  0.27877233  0.75180530
## 22  1.08997319 -0.94813829
## 23  0.01837079  0.03247537
## 24 -0.03489873 -0.54117005
## 25 -1.57564945 -0.46595942
## 26  1.32309697 -0.45410754
## 27  0.76477451  1.46224032
## 28 -0.56413583  0.39965765
## 29 -0.85789449 -0.34506095
## 30 -0.12343120  0.34233173
## 31 -0.58615914  0.98779924
## 32  0.01828578 -1.17413897
## 33  0.81658802 -0.25429589
## 34 -1.43546222 -0.81620435
## 35  0.74509265  1.35913570
## 36  0.64591896 -0.11111270
## 37  1.21070910 -0.95633572
## 38 -0.87886235 -0.32511853
## 39 -1.38233822  0.51890327
## 40  0.05285613 -0.89106624
## 41  0.16039011  1.65610198
## 42 -0.09752445 -0.51593638
## 43  0.84287084 -1.42887409
## 44 -0.36308318  0.74525270
## 45 -1.50269806  0.21079062
## 46 -0.42979742  0.69447536
## 47 -1.50952957  1.04332132
## 48  0.21380517  1.46828518
## 49  0.44457525  1.06621669
## 50  1.04423905  0.56848741
## 51 -0.66640311  0.25327565
## 52 -1.06410434  1.28986302
## 53 -0.29584733 -0.28174228
## 54  0.37663553 -0.82959015
## 55  0.21842851 -1.02180698
## 56  0.66279829 -0.20706611
```

```
## 57 -1.06418935  0.08324869
## 58  0.95564703  0.34438945
## 59  0.64614951  0.33403943
## 60  0.43467296 -0.43182281
## 61  1.58972340 -0.69615034
## 62 -1.53590966 -0.57606026
## 63  0.14404560 -0.81404776
## 64  1.13025908  1.03867946
## 65 -0.41041887  0.73315897
## 66 -0.19333630 -1.17853900
## 67 -0.02068962 -0.10931287
## 68 -1.28197580 -0.92785516
## 69 -1.27740073  0.13747469
## 70  0.71931918 -0.13140385
## 71  0.63817753  0.52815496
## 72  0.14624152 -0.74383175
## 73  0.20056649 -0.73838550
## 74  0.35253637 -1.17214139
## 75 -0.52934719 -0.45946294
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                  X1         X2        X3
## Factor1 -0.06685977  0.6641094 0.3335712
## Factor2  0.71068488 -0.1364369 0.2538867
##
## Slot "meanF":
##  Factor1  Factor2
## 4.272569 2.072818
##
## Slot "corF":
##           Factor1   Factor2
## Factor1 1.0000000 0.9730208
## Factor2 0.9730208 1.0000000
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
```

```
## Slot "center":
##       X1       X2       X3
## 1.537705 1.780328 1.686885
##
## Slot "eigenvalues":
## [1] 1.935081 1.591967 1.370366
##
## Slot "cov.control":
## An object of class "CovControlMcd"
## Slot "alpha":
## [1] 0.5
##
## Slot "nsamp":
## [1] 500
##
## Slot "scalefn":
## NULL
##
## Slot "maxcsteps":
## [1] 200
##
## Slot "seed":
## NULL
##
## Slot "use.correction":
## [1] TRUE
##
## Slot "trace":
## [1] FALSE
##
## Slot "tolSolve":
## [1] 1e-14
##
##
##
## Slot "importance":
##                Factor1 Factor2
## SS loadings      1.812   1.715
## Proportion Var   0.370   0.350
## Cumulative Var   0.370   0.720
```

From the `summary` result of `faCovPcaRegMcd`, we see that the first two factors accout for about 72.0% of its total variance.

Next we calculate prediction/scores using `predict()`. The usage is `predict(object, ...)`, where `object` is an object of class `Fa`. If missing `...`(`newdata`), the `scores` slot of `object` is used. To save space, only the first five and last five rows of the scores are displayed.

```
  predict(faCovPcaRegMcd)
```

```
##       Factor1    Factor2
## 1  20.1391070 10.4105484
## 2  20.9770640 10.0136763
## 3  21.3980990 11.4405912
## 4  22.5484289 10.8723961
## 5  22.0558985 11.0589128
## 71  0.6381775  0.5281550
## 72  0.1462415 -0.7438318
## 73  0.2005665 -0.7383855
## 74  0.3525364 -1.1721414
## 75 -0.5293472 -0.4594629
```

If not missing `...`, then `...` must have the name `newdata`. Moreover, `newdata` should be scaled first. For example, the original data is `x = hbk.x`, and `newdata` is a one row `data.frame`.

```
  newdata = hbk.x[1, ]
cor = FALSE # the default
newdata = { if (cor == TRUE)
  # standardized transformation
  scale(newdata, center = faCovPcaRegMcd@center,
        scale = sqrt(diag(faCovPcaRegMcd@covariance)))
  else # cor == FALSE
    # centralized transformation
    scale(newdata, center = faCovPcaRegMcd@center, scale = FALSE)
}
```

Finally we get

```
  prediction = predict(faCovPcaRegMcd, newdata = newdata)
prediction

##    Factor1  Factor2
## 1 20.13911 10.41055
```

One can easily check that

```
prediction = predict(faCovPcaRegMcd)[1,] = faCovPcaRegMcd@scores[1,]
```

To visualize the factor analysis results, the `plot` method can be used. We have `setMethod plot` with signature

```
x = "Fa", y = "missing".
```

The usage is

`plot(x, which = c("factorScore", "screeplot"), choices = 1:2).`

Where `x` is an object of class `Fa`. The argument `which` indicates what kind of plot. If `which = "factorScore"`, then a scatterplot of the factor scores is produced; if `which = "screeplot"`, then the eigenvalues plot is created and is helpful to select the number of factors. The argument `choices` is an integer vector of length two indicating which columns of the factor scores to plot. To see how `plot` is functioning, we first generate an `Fa` object.

```
  faClassicPcaReg = FaClassic(x = hbk.x, factors = 2, method = "pca",
                              scoresMethod = "regression"); faClassicPcaReg

## An object of class "FaClassic"
## Slot "call":
## FaClassic(x = hbk.x, factors = 2, method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##      Factor1  Factor2
## X1  3.411036 0.936662
## X2  7.278529 3.860723
## X3 11.270812 3.273734
##
## Slot "communality":
##        X1        X2        X3
##  12.51250  67.88217 137.74853
##
## Slot "uniquenesses":
##           X1           X2           X3
## 0.8292095832 0.0007959085 0.0863235416
##
## Slot "cor":
## [1] FALSE
##
## Slot "covariance":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "correlation":
##           X1        X2        X3
## X1 1.0000000 0.9459958 0.9617790
## X2 0.9459958 1.0000000 0.9786612
```

```
## X3 0.9617790 0.9786612 1.0000000
##
## Slot "usedMatrix":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##         Factor1      Factor2
## 1   1.836216178  0.161337548
## 2   1.756800683  0.549735006
## 3   2.250318962 -0.462115084
## 4   2.110379324  0.145591275
## 5   2.095218367  0.066345803
## 6   1.906582805  0.232737572
## 7   1.788199906  0.587263155
## 8   1.911901278  0.021274118
## 9   2.106202931 -0.053757894
## 10  2.122518267 -0.342046036
## 11  2.348800415  0.342830312
## 12  2.927387983 -1.009336488
## 13  1.905818664  1.685956006
## 14  0.528829255  6.359573459
## 15 -0.448347445  0.133780216
## 16 -0.668908585  0.366404244
## 17 -0.779858578  0.442576537
## 18 -0.320380311 -0.436151460
## 19 -0.697421067  0.621061863
## 20 -0.531502346  0.422359906
## 21 -0.418046839 -0.099159644
```

```
## 22 -0.718663795  0.742356037
## 23 -0.665393822  0.394816805
## 24 -0.761582949  0.580540198
## 25 -0.419693120 -0.657298051
## 26 -0.598455877  0.539385243
## 27 -0.247874654 -0.345083251
## 28 -0.219836054 -0.829109302
## 29 -0.484797036 -0.302159906
## 30  0.006043560 -1.273196784
## 31 -0.414104099 -0.319367475
## 32 -0.862788836  0.802561432
## 33 -0.704564559  0.680970529
## 34 -0.404005807 -0.681812987
## 35 -0.226207238 -0.410125033
## 36 -0.370021556 -0.176933456
## 37 -0.581066520  0.435007090
## 38 -0.621218623  0.029086231
## 39 -0.210664210 -1.057488778
## 40 -0.638509599  0.278324368
## 41 -0.093737402 -0.869315133
## 42 -0.222301393 -0.760169894
## 43 -0.929819437  1.166157588
## 44  0.040846868 -1.395395068
## 45 -0.345171409 -0.777979662
## 46  0.005618871 -1.329449176
## 47 -0.241826492 -0.980332711
## 48 -0.094700744 -0.865086128
## 49 -0.436399538  0.009447513
## 50 -0.371498956 -0.025665034
## 51 -0.585765982  0.034545544
## 52 -0.234984713 -0.865075764
## 53 -0.017987529 -1.299228393
## 54 -0.924264023  1.069378350
## 55 -0.757104843  0.604723754
## 56 -0.808109523  0.898476392
## 57 -0.432379728 -0.457331137
## 58 -0.698147469  0.739975084
## 59 -0.269779011 -0.394725685
## 60 -0.198808958 -0.673656887
## 61 -0.557786430  0.493393901
## 62 -0.459937878 -0.555054318
## 63 -0.892563087  0.931740886
## 64 -0.361028449  0.001270686
## 65 -0.067714855 -1.141591364
## 66 -0.694666313  0.333677831
## 67 -0.570392680  0.141917212
```

```
## 68 -0.352127153 -0.776502820
## 69 -0.474012517 -0.407110790
## 70 -0.340986005 -0.230478698
## 71 -0.421675567 -0.010838417
## 72 -0.579475155  0.167004879
## 73 -0.425439569 -0.197214204
## 74 -0.631892463  0.322176109
## 75 -0.141285522 -1.068417769
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                   X1         X2         X3
## Factor1  0.06192447 -0.1643831  0.1761400
## Factor2 -0.12400651  0.5687014 -0.3297295
##
## Slot "meanF":
##       Factor1       Factor2
## -1.368424e-16  2.902366e-16
##
## Slot "corF":
##             Factor1     Factor2
## Factor1 1.00000e+00 1.62519e-15
## Factor2 1.62519e-15 1.00000e+00
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
## Slot "center":
##       X1       X2       X3
## 3.206667 5.597333 7.230667
##
## Slot "eigenvalues":
## [1] 216.162129   1.981077   0.916329
##
## Slot "cov.control":
## NULL


summary(faClassicPcaReg)
```

```
## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaClassic"
## Slot "call":
## FaClassic(x = hbk.x, factors = 2, method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##      Factor1  Factor2
## X1  3.411036 0.936662
## X2  7.278529 3.860723
## X3 11.270812 3.273734
##
## Slot "communality":
##        X1        X2        X3
##  12.51250  67.88217 137.74853
##
## Slot "uniquenesses":
##           X1           X2           X3
## 0.8292095832 0.0007959085 0.0863235416
##
## Slot "cor":
## [1] FALSE
##
## Slot "covariance":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "correlation":
##           X1        X2        X3
## X1 1.0000000 0.9459958 0.9617790
## X2 0.9459958 1.0000000 0.9786612
## X3 0.9617790 0.9786612 1.0000000
##
## Slot "usedMatrix":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "reducedCorrelation":
## NULL
```

```
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##         Factor1      Factor2
## 1   1.836216178  0.161337548
## 2   1.756800683  0.549735006
## 3   2.250318962 -0.462115084
## 4   2.110379324  0.145591275
## 5   2.095218367  0.066345803
## 6   1.906582805  0.232737572
## 7   1.788199906  0.587263155
## 8   1.911901278  0.021274118
## 9   2.106202931 -0.053757894
## 10  2.122518267 -0.342046036
## 11  2.348800415  0.342830312
## 12  2.927387983 -1.009336488
## 13  1.905818664  1.685956006
## 14  0.528829255  6.359573459
## 15 -0.448347445  0.133780216
## 16 -0.668908585  0.366404244
## 17 -0.779858578  0.442576537
## 18 -0.320380311 -0.436151460
## 19 -0.697421067  0.621061863
## 20 -0.531502346  0.422359906
## 21 -0.418046839 -0.099159644
## 22 -0.718663795  0.742356037
## 23 -0.665393822  0.394816805
## 24 -0.761582949  0.580540198
## 25 -0.419693120 -0.657298051
## 26 -0.598455877  0.539385243
## 27 -0.247874654 -0.345083251
## 28 -0.219836054 -0.829109302
## 29 -0.484797036 -0.302159906
## 30  0.006043560 -1.273196784
## 31 -0.414104099 -0.319367475
```

```
## 32 -0.862788836  0.802561432
## 33 -0.704564559  0.680970529
## 34 -0.404005807 -0.681812987
## 35 -0.226207238 -0.410125033
## 36 -0.370021556 -0.176933456
## 37 -0.581066520  0.435007090
## 38 -0.621218623  0.029086231
## 39 -0.210664210 -1.057488778
## 40 -0.638509599  0.278324368
## 41 -0.093737402 -0.869315133
## 42 -0.222301393 -0.760169894
## 43 -0.929819437  1.166157588
## 44  0.040846868 -1.395395068
## 45 -0.345171409 -0.777979662
## 46  0.005618871 -1.329449176
## 47 -0.241826492 -0.980332711
## 48 -0.094700744 -0.865086128
## 49 -0.436399538  0.009447513
## 50 -0.371498956 -0.025665034
## 51 -0.585765982  0.034545544
## 52 -0.234984713 -0.865075764
## 53 -0.017987529 -1.299228393
## 54 -0.924264023  1.069378350
## 55 -0.757104843  0.604723754
## 56 -0.808109523  0.898476392
## 57 -0.432379728 -0.457331137
## 58 -0.698147469  0.739975084
## 59 -0.269779011 -0.394725685
## 60 -0.198808958 -0.673656887
## 61 -0.557786430  0.493393901
## 62 -0.459937878 -0.555054318
## 63 -0.892563087  0.931740886
## 64 -0.361028449  0.001270686
## 65 -0.067714855 -1.141591364
## 66 -0.694666313  0.333677831
## 67 -0.570392680  0.141917212
## 68 -0.352127153 -0.776502820
## 69 -0.474012517 -0.407110790
## 70 -0.340986005 -0.230478698
## 71 -0.421675567 -0.010838417
## 72 -0.579475155  0.167004879
## 73 -0.425439569 -0.197214204
## 74 -0.631892463  0.322176109
## 75 -0.141285522 -1.068417769
##
## Slot "scoresMethod":
```

```
## [1] "regression"
##
## Slot "scoringCoef":
##                   X1         X2         X3
## Factor1  0.06192447 -0.1643831  0.1761400
## Factor2 -0.12400651  0.5687014 -0.3297295
##
## Slot "meanF":
##       Factor1       Factor2
## -1.368424e-16  2.902366e-16
##
## Slot "corF":
##             Factor1     Factor2
## Factor1 1.00000e+00 1.62519e-15
## Factor2 1.62519e-15 1.00000e+00
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
## Slot "center":
##       X1       X2       X3
## 3.206667 5.597333 7.230667
##
## Slot "eigenvalues":
## [1] 216.162129   1.981077   0.916329
##
## Slot "cov.control":
## NULL
##
##
## Slot "importance":
##                Factor1 Factor2
## SS loadings    191.643  26.500
## Proportion Var   0.875   0.121
## Cumulative Var   0.875   0.996
```

`faClassicPcaReg` is an object of class `FaClassic`. From the `summary` result of `faClassicPcaReg`, we see that the first two factors accout for about 99.6% of its total variance. Next we generate an object `faCovPcaRegMcd` of class `FaCov` using the same data set. The `print` and `summary` results have already been shown before.

The following code lines generate classical and robust scatterplots of the first two factor scores. See Figure 8.

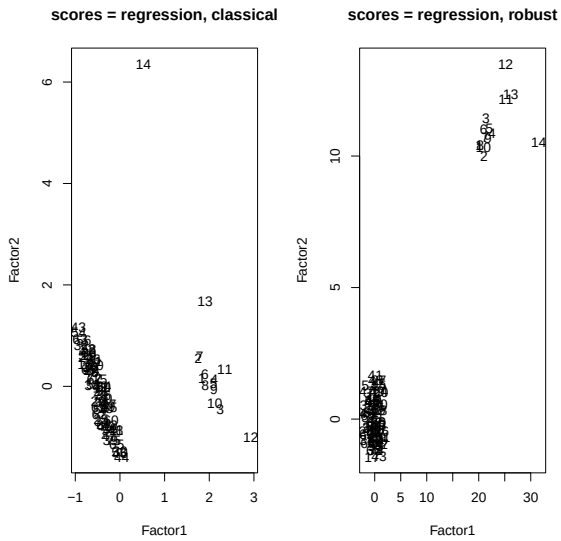


Figure 3: Classical and robust scatterplots of the first two factor scores of the hbk data.

The following code lines generate classical and robust scree plots. See Figure 9.

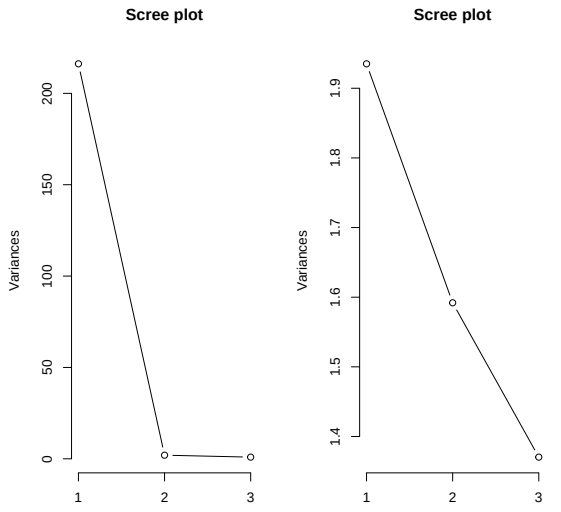


Figure 4: Classical and robust scree plots of the hbk data.

Next we impose a 97.5% tolerance ellipse on the scatterplot of the first two factor scores of the `hbk` data by using a function `rrcov:::.myellipse`. See Figure 5. The left panel shows the classical 97.5% tolerance ellipse, which is very large and contains the outliers 1-13. The regular points are not well seperated from the outliers. The right panel shows the robust 97.5% tolerance ellipse which clearly separates the regular points and outliers. We see that

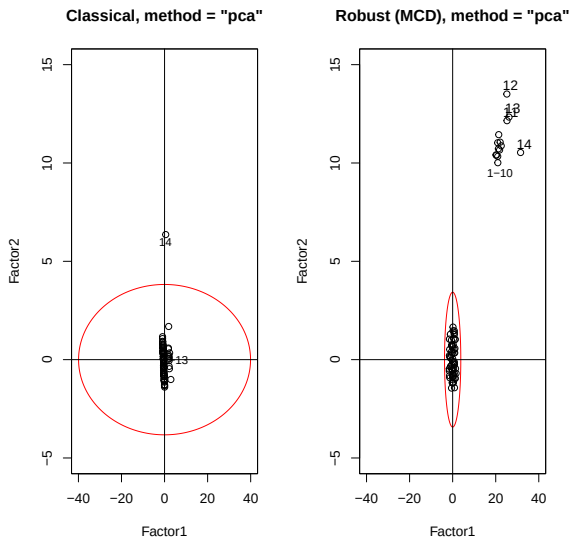


Figure 5: Classical and robust scatterplot of the first two factor scores of the hbk data with a 97.5% tolerance ellipse.

the estimate of the center of the regular points is located at the origin (where the mean of the scores should be located). The following code lines compute the means of the classical and robust scores. We see that the mean of all observations of the classical scores equals 0, while the mean of good observations (regular points, excluding the outliers) of the robust scores equals 0.

```
colMeans(faClassicPcaReg@scores)

##       Factor1       Factor2
## -1.369275e-16  2.901383e-16

colMeans(faCovPcaRegMcd@scores)

##  Factor1  Factor2
## 4.272569 2.072818

colMeans(faClassicPcaReg@scores[15:75,])

##    Factor1    Factor2
## -0.4523799 -0.1358260

colMeans(faCovPcaRegMcd@scores[15:75,])

##       Factor1       Factor2
## -6.415633e-17  1.433280e-17
```

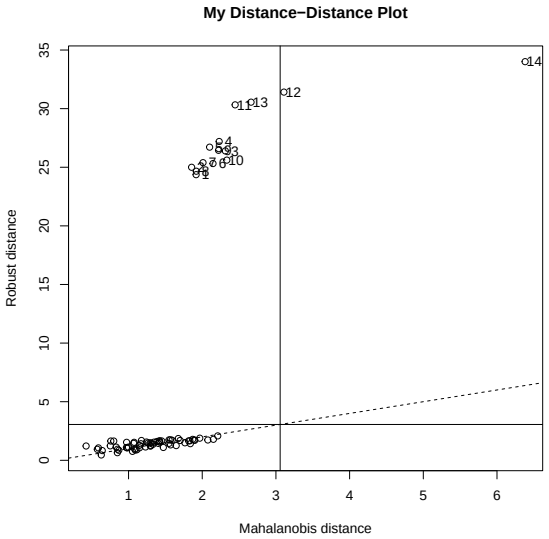
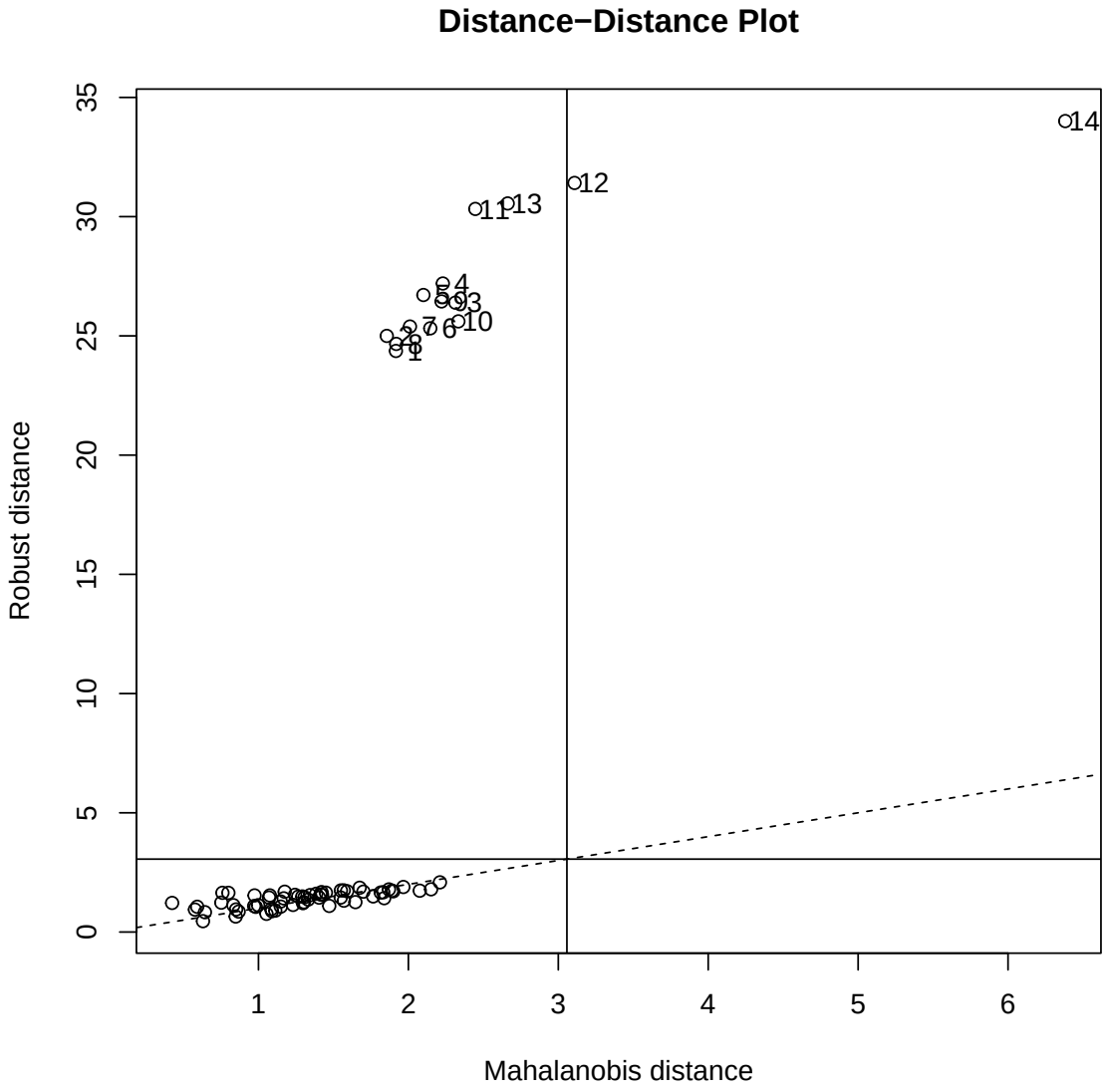
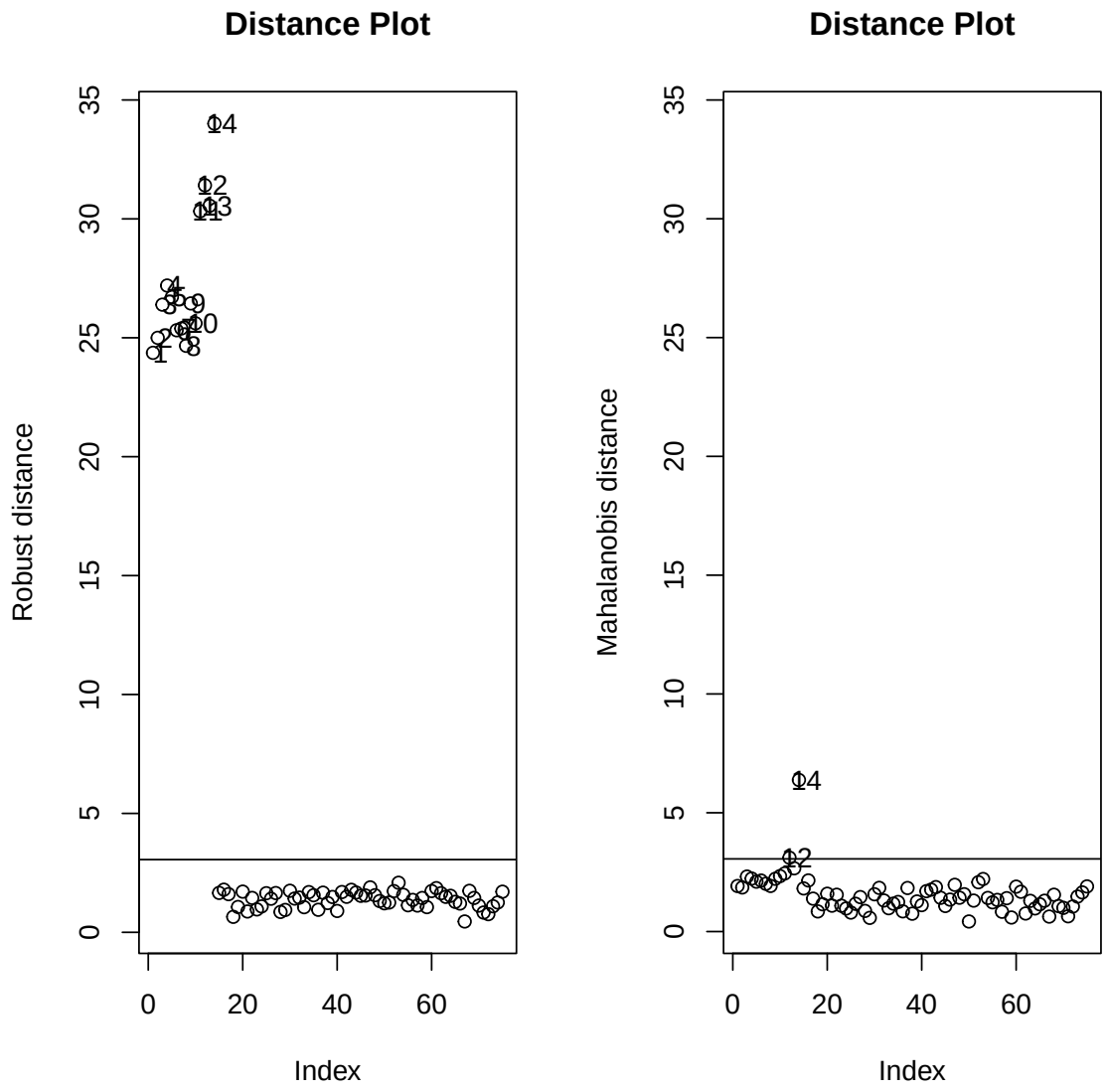
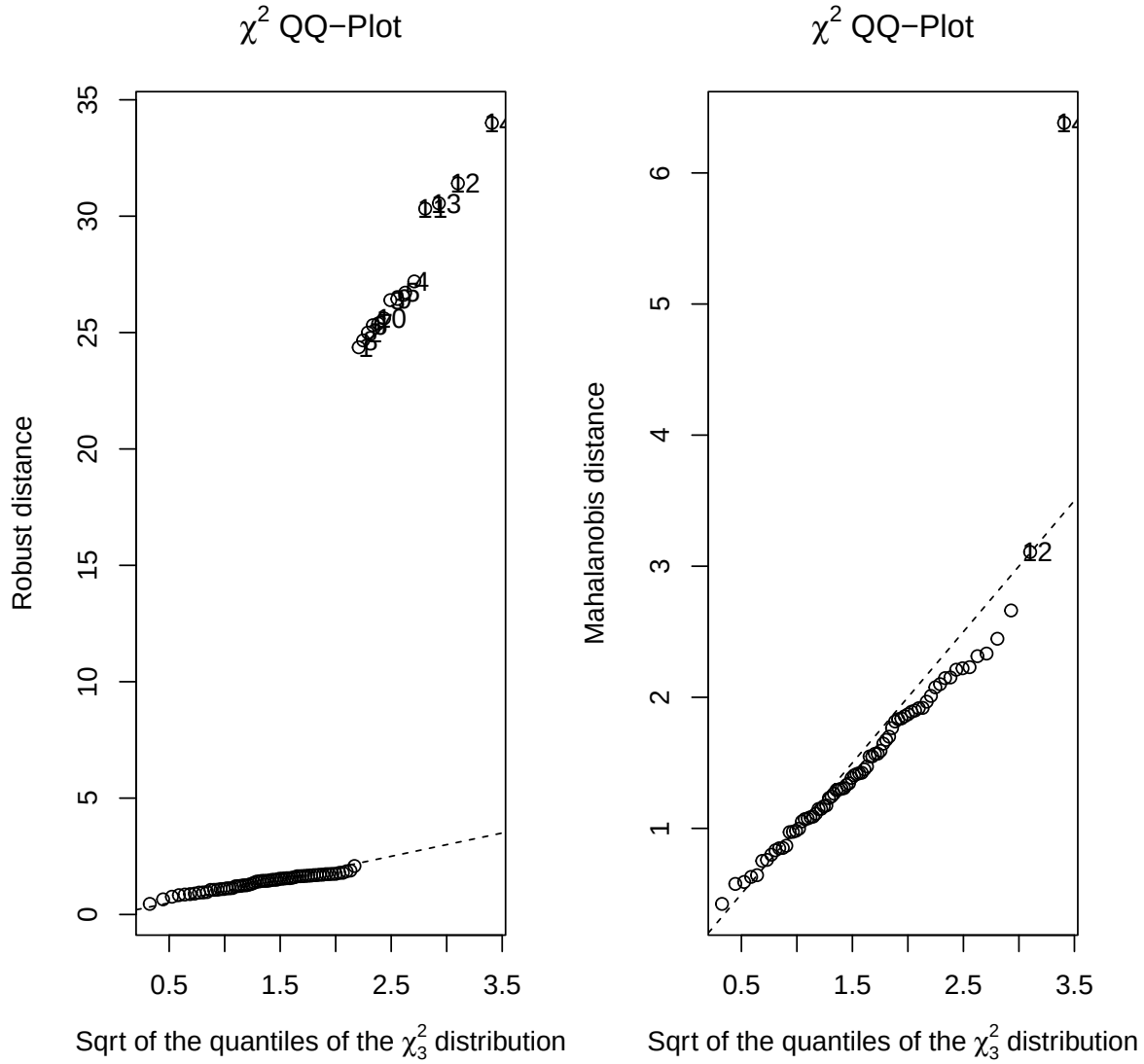


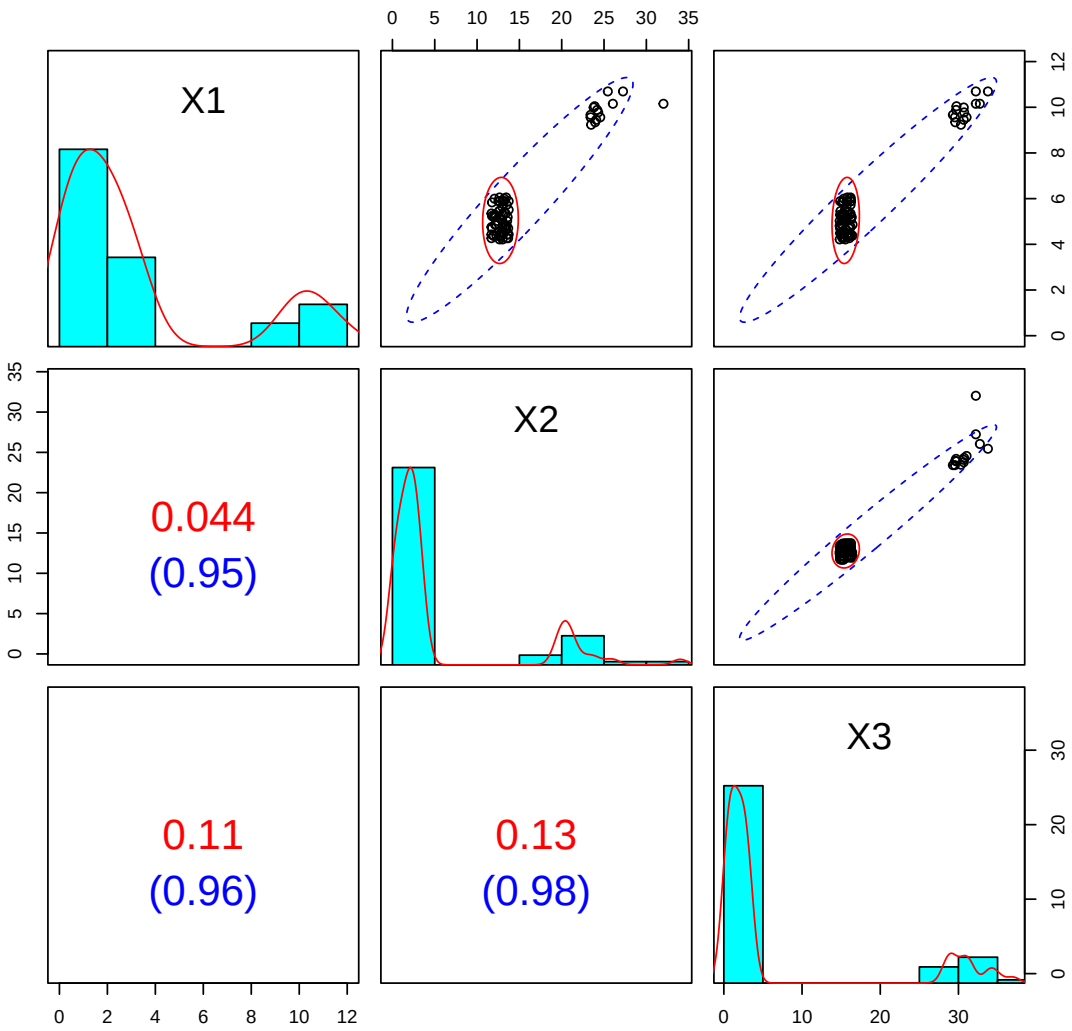
Figure 6: A distance-distance plot for hbk data.

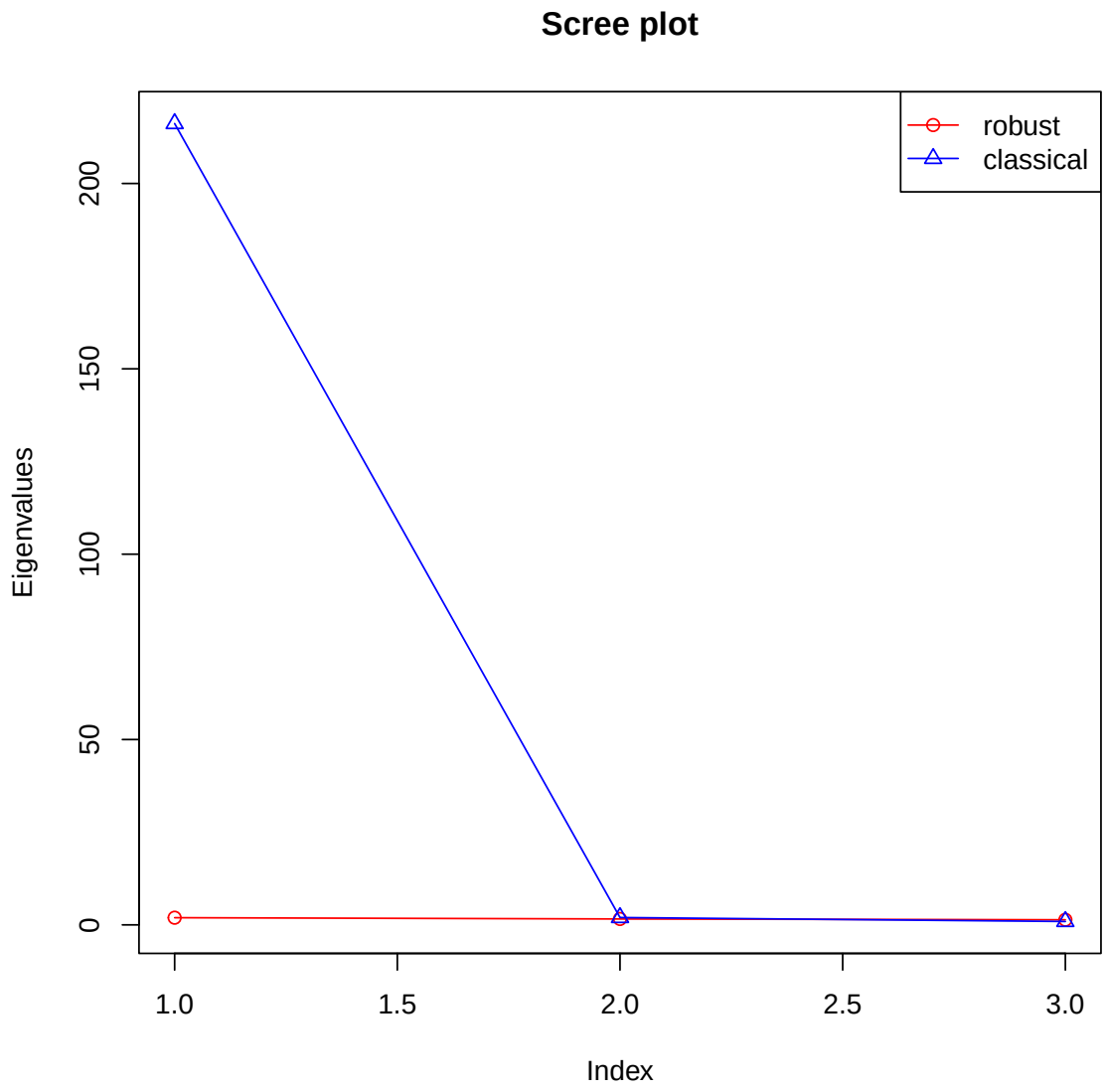



Distance Plot
Robust distance
Index
0
5
10
15
20
25
30
35
0
20
40
60
Distance Plot
Mahalanobis distance
Index




```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```


X1
X2
X3
0.044
(0.95)
0.11
(0.96)
0.13
(0.98)



Next we plot a `Cov-class`. See Figure 11. The figure shows a distance-distance plot. We see that the robust (mahalanobis) distances are far larger than the (classical) mahalanobis distances. The outliers have large robust distances. The figure is generated by `myplotDD(x = covMcd)`. `myplotDD()` is a revised version of `.myddplot()` in `plot-utils.R` in the package **rrcov**. In `myplotDD()`, `id.n` and `ind` are printed out. Here `id.n` is the number of observations to identify by a label. By default, the number of observations with robust distances larger than `cutoff` is used. By default `cutoff = sqrt(qchisq(0.975, p))`. `ind` is the index of robust distances whose values are larger than `cutoff`.

```
## cutoff = 3.057516
## id.n <- length(which(rd>cutoff))
## id.n = 14
## Here y is the robust distance (rd).
```

```
## sort.y = (To save space, only the smallest five and largest five
## elements of sort.y$x and sort.y$ix are shown.)
## $x
##         67         18         72         71         28          4
##  0.4553524  0.6505517  0.7631246  0.8306446  0.8570290 27.1972578
##         11         13         12         14
## 30.3225470 30.5536038 31.4123574 34.0079508
##
## $ix
##  [1] 67 18 72 71 28  4 11 13 12 14
## ind =
##  [1]  1  8  2  6  7 10  3  9  5  4 11 13 12 14
```

From the above results we see that the `cutoff` is computed as 3.057516. There are `id.n = 14` observations with robust distance larger than `cutoff`. `sort.y` is a list containing the sorted values of `y` (the robust distance). `sort.y$x` is arranged in increasing order. To save space, only the smallest five and largest five robust distances with their indices are shown. `sort.y$ix` contains the indices. `ind` shows the indices of the largest `id.n = 14` observations whose robust distances are larger than `cutoff`.

The accessor functions `getCenter()`, `getEigenvalues()`, `getLoadings()`, `getQuan()`, `getScores()`, and `getSdev()` are used to access the corresponding slots. For instance

```
robustfa::getEigenvalues(faCovPcaRegMcd)

## [1] 1.935081 1.591967 1.370366
```

The function `getFa()`, which is used in `predict()` and `screeplot()`, returns a list of class `"fa"`.

Note that our previous comparisons in this subsection are just (1) vs (5), i.e., classical and robust factor analysis comparisons using `x = hbk.x` as the data input and the covariance matrix (`cor = FALSE`) as the running matrix.

For `x = hbk.x`, we checked that $\boldsymbol{S}^r \neq \tilde{\boldsymbol{S}}^r$, and $\boldsymbol{R}^r = \tilde{\boldsymbol{R}}^r$ for `control = "mcd"`, `"ogk"`, `"m"`, `"mve"`, `"sfast"`, `"bisquare"`, `"rocke"`, small differences between $\boldsymbol{R}^r$ and $\tilde{\boldsymbol{R}}^r$ for `control = "sde"`, `"surreal"`. The results illustrate Theorem 3.1.

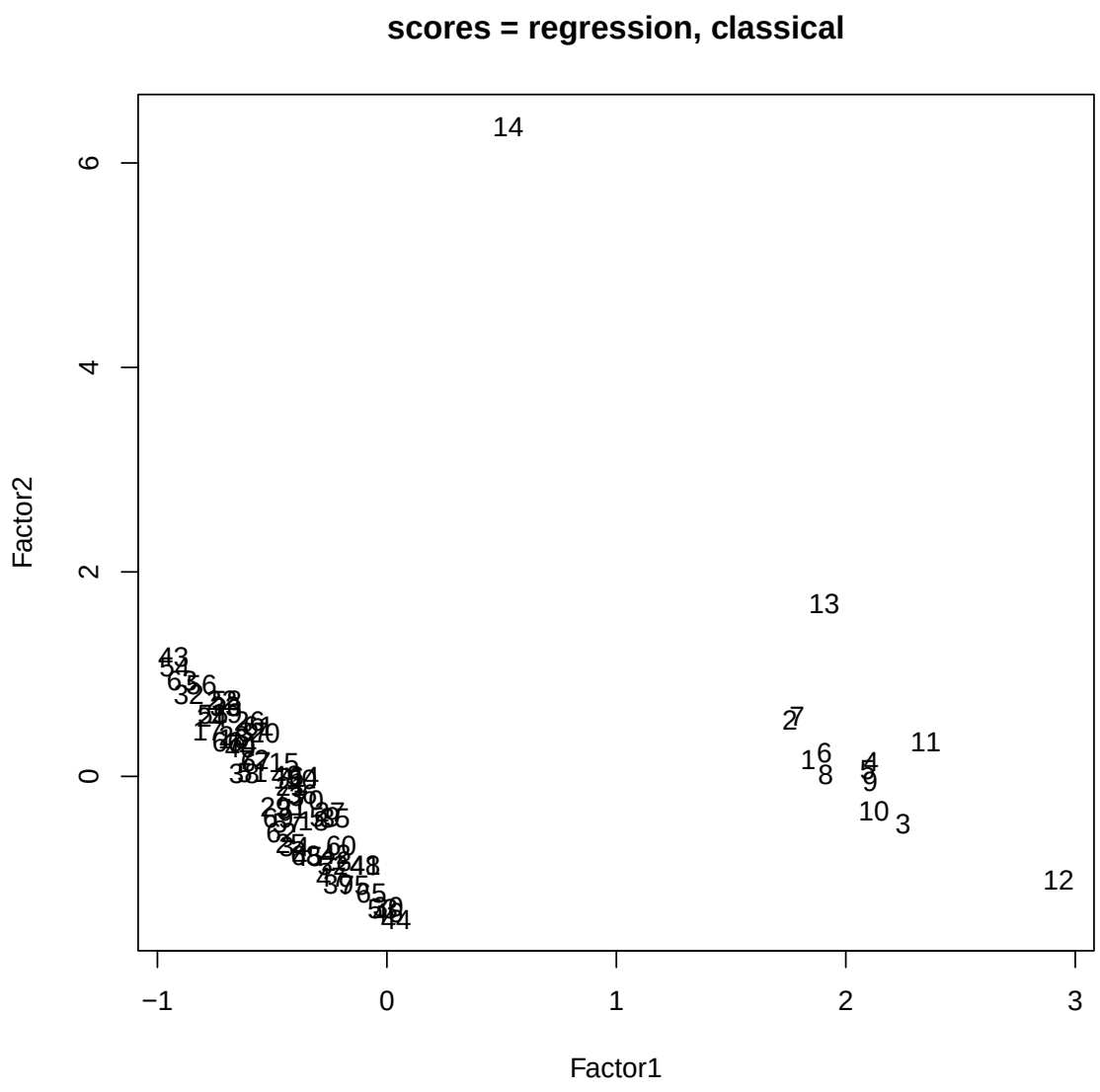
The eigenvalues of the running matrices of the `hbk` data of the 8 combinations are given in Table 6. From Table 6 we see that the eigenvalues of (2), (3), and (4) are the same, the eigenvalues of (6) and (8) are the same. The results illustrate Theorems 3.1 and 3.2.

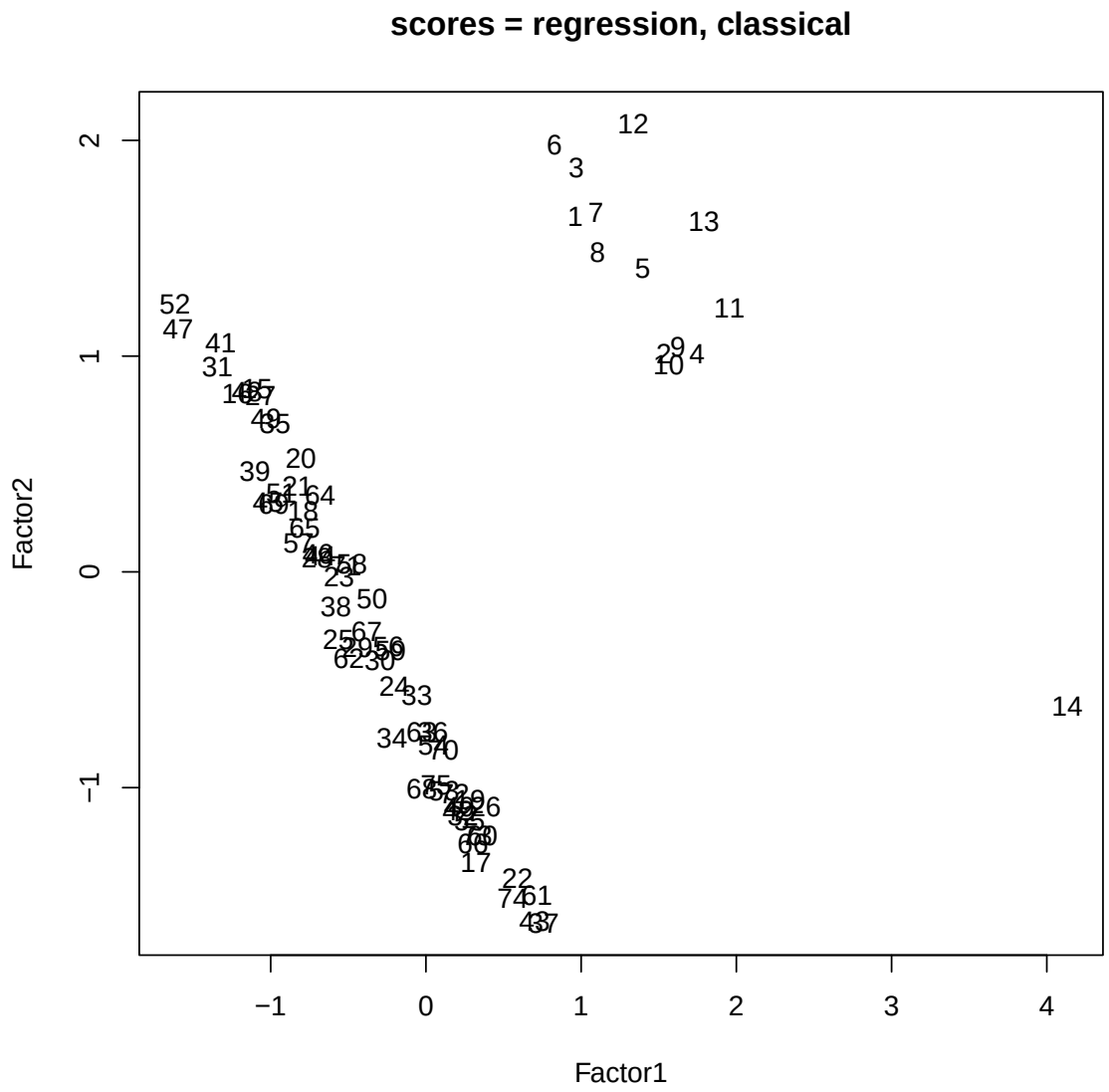
Classical and robust (MCD) scatterplots of the first two factor scores of the `hbk` data with 97.5% tolerance ellipses are ploted in Figure 12. We see that the scores of (2), (3), and (4) are the same, the scores of (6) and (8) are the same, in agree with Theorem 3.2. Note that the tolerance ellipse is very large for (1), since the outliers severely affected the eigenvalues of the running matrix $\boldsymbol{S}^c$. While the tolerance ellipses are very small for (2), (3), and (4), also due to the outliers severely affected the eigenvalues of the running matrices $\boldsymbol{R}^c = \tilde{\boldsymbol{S}}^c = \tilde{\boldsymbol{R}}^c$. The tolerance ellipse is very small for (7) and it does not cover the regular points, due to the first two eigenvalues of (7) are very small. It exemplifies that the results from robust

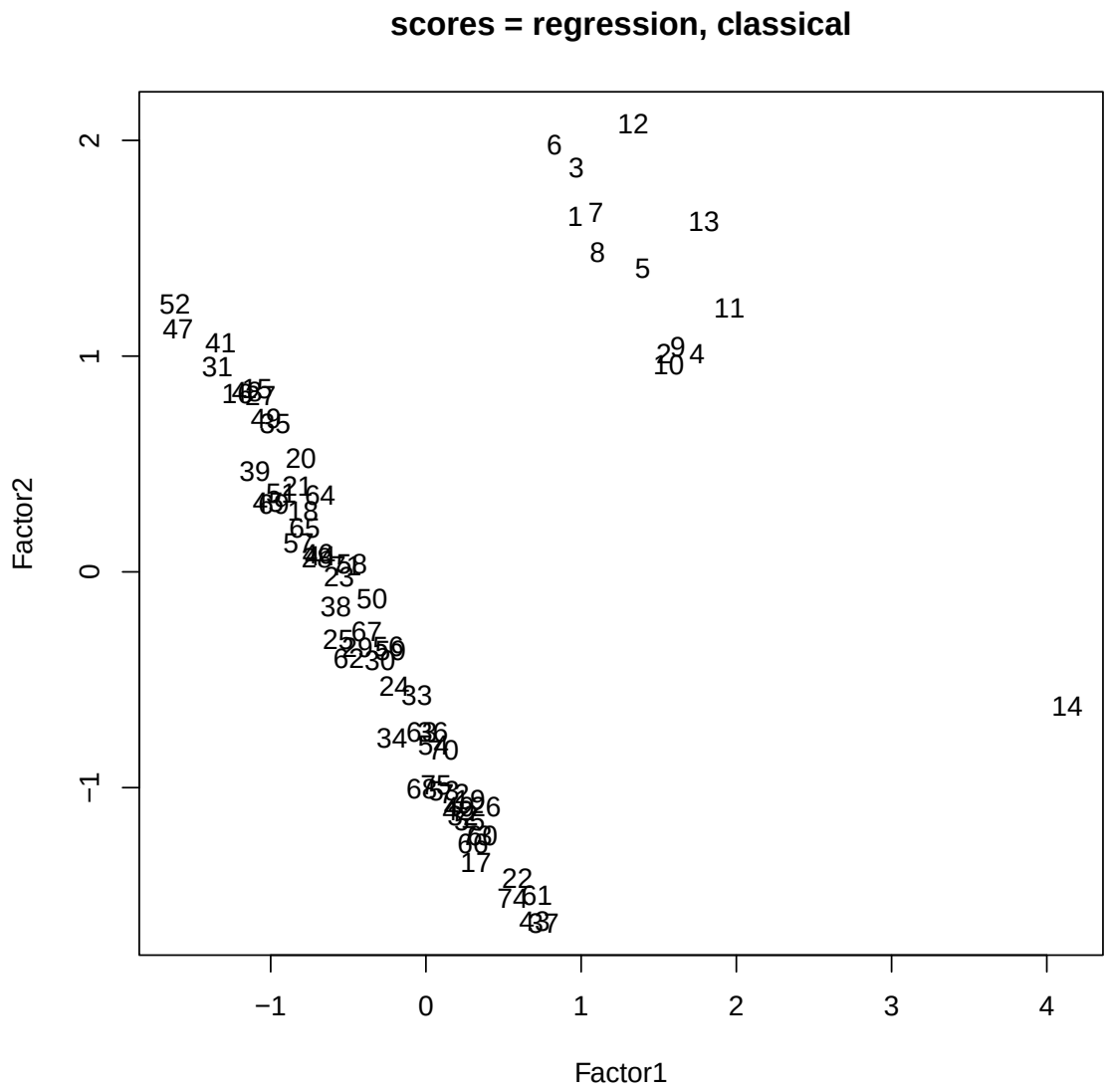
Table 5: The eigenvalues of the running matrices for hbk data.

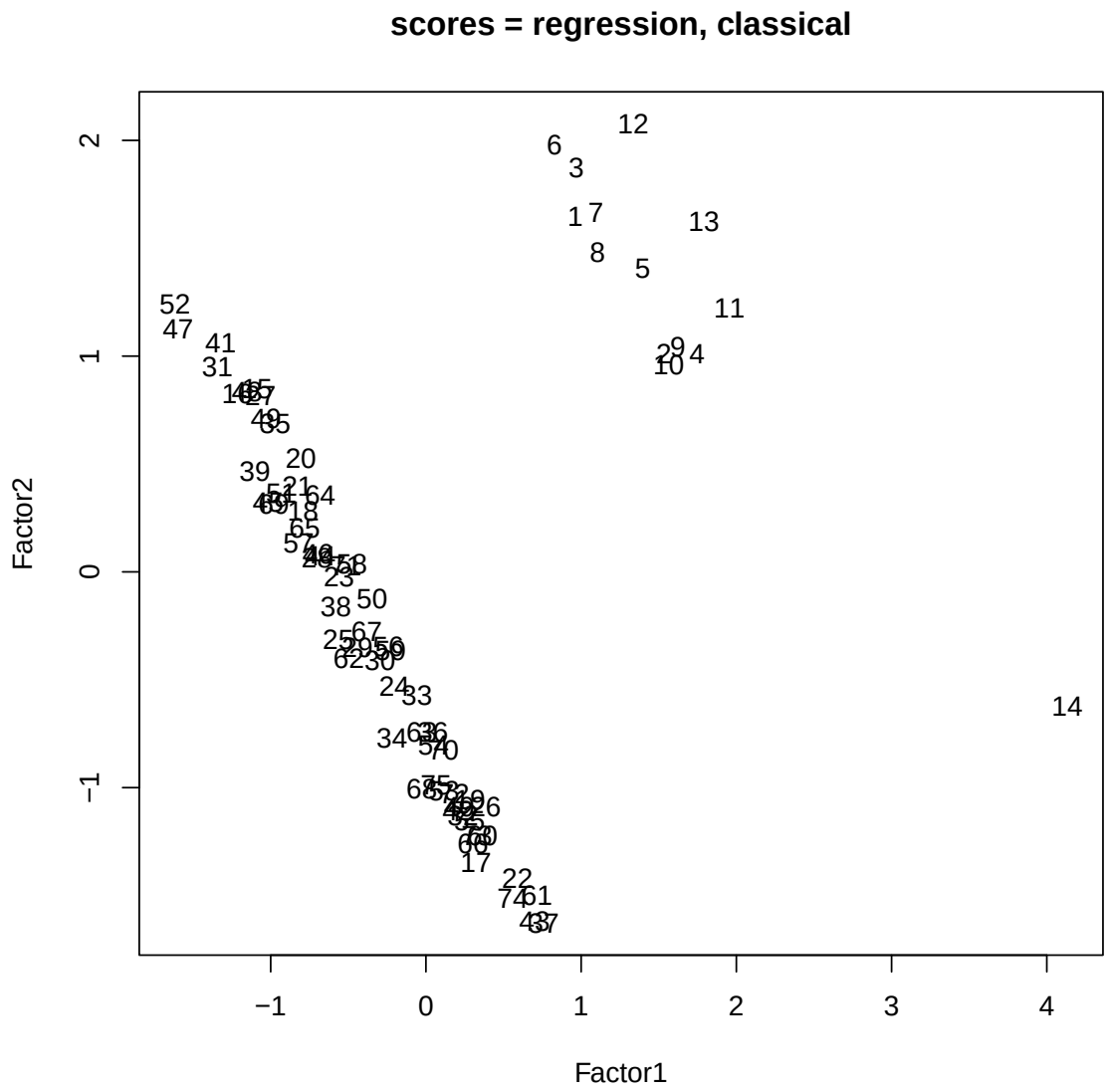
		Classical	Robust (MCD)
hbk.x	covariance	(1) 216.162129 1.981077 0.916329	(5) 1.935081 1.591967 1.370366
	correlation	(2) 2.92435228 0.05668483 0.01896288	(6) 1.1890834 0.9563255 0.8545911
scale(hbk.x)	covariance	(3) 2.92435228 0.05668483 0.01896288	(7) 0.12408511 0.02501676 0.01089401
	correlation	(4) 2.92435228 0.05668483 0.01896288	(8) 1.1890834 0.9563255 0.8545911

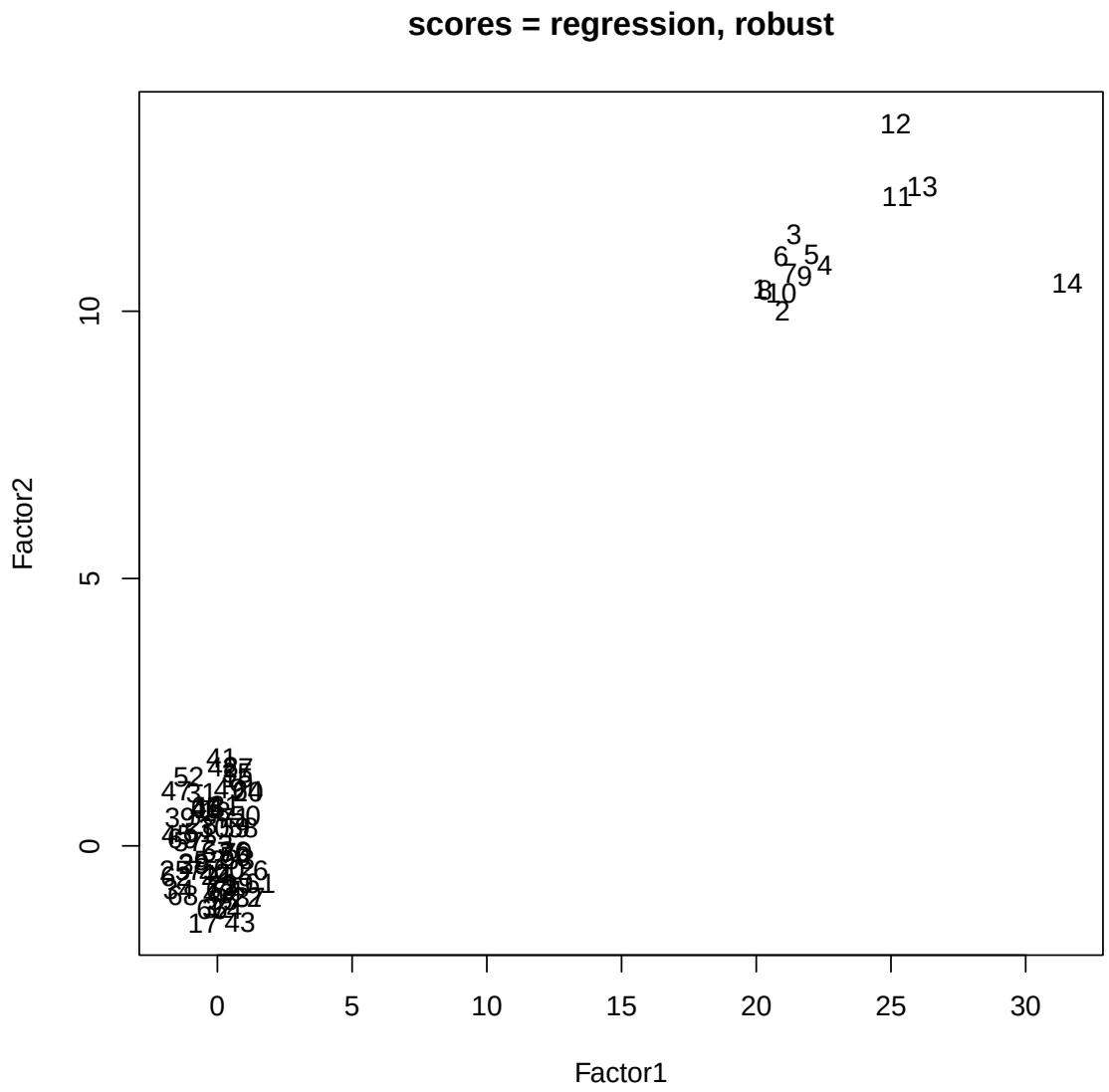
covariance matrix of the scaled data is not very reliable. The tolerance ellipses of (6) and (8) well seperate the regular points and the outliers.

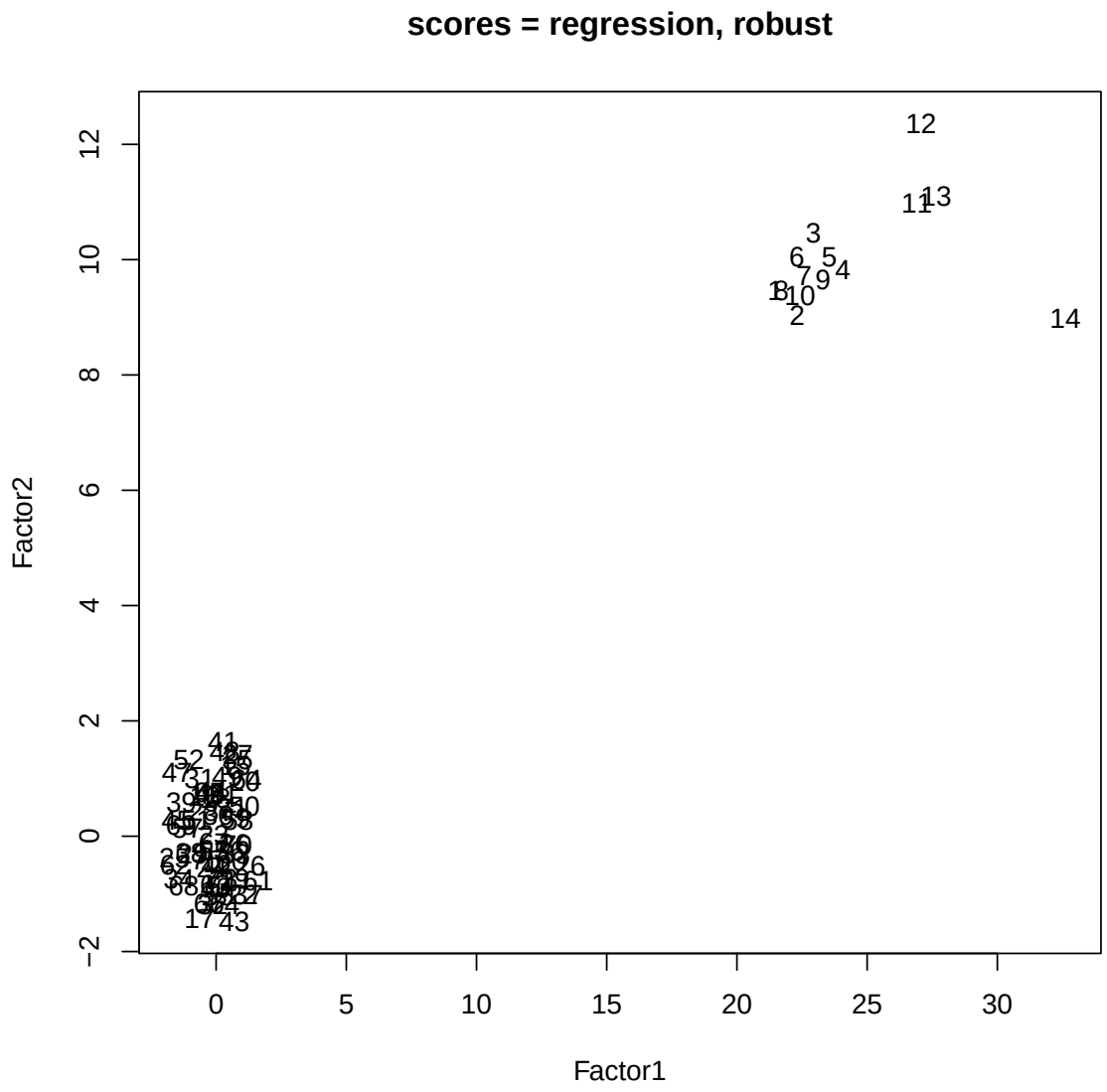


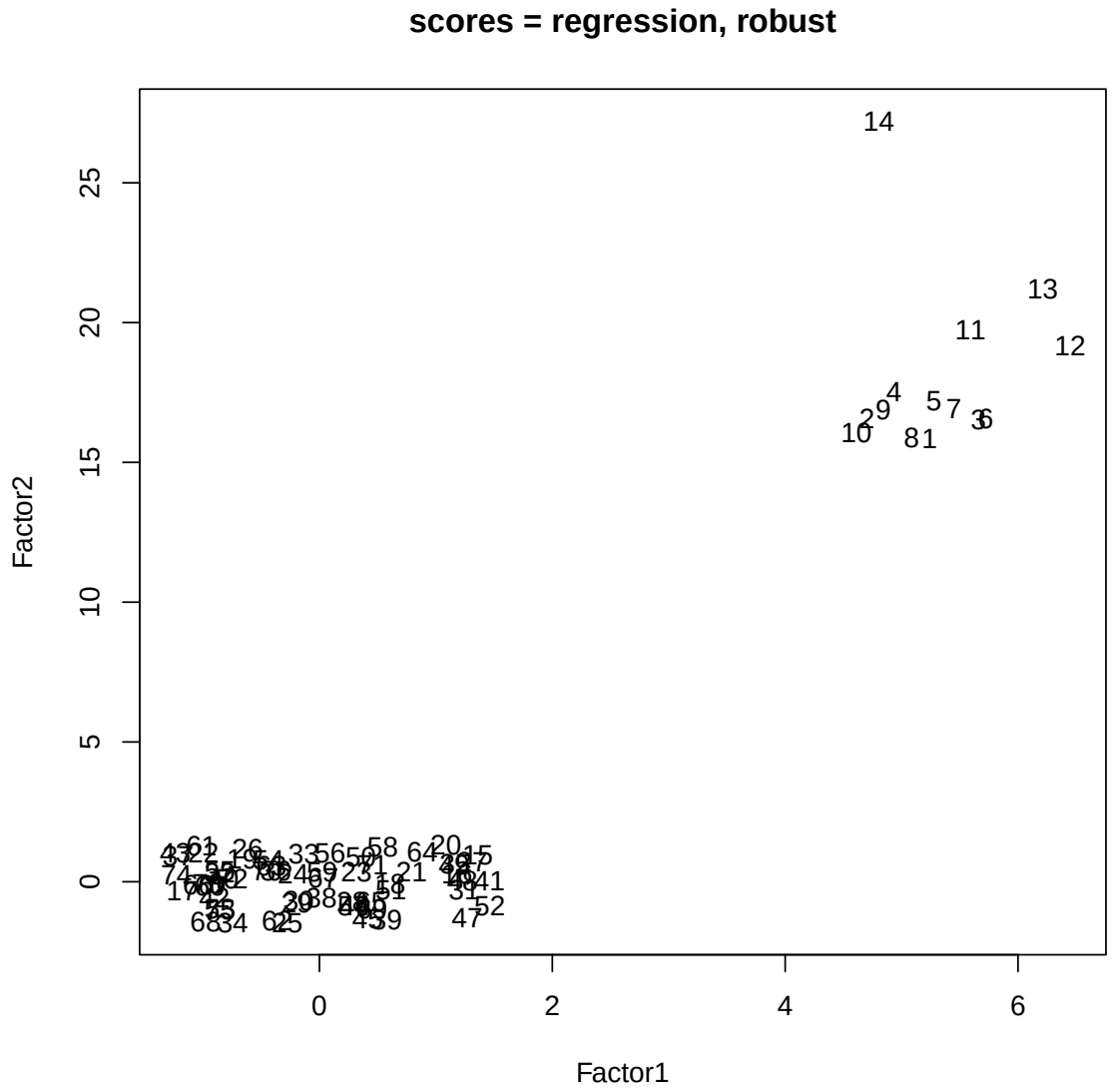

scores = regression, classical
Factor2
Factor1
2
1
0
−1
−1
0
1
2
3
4

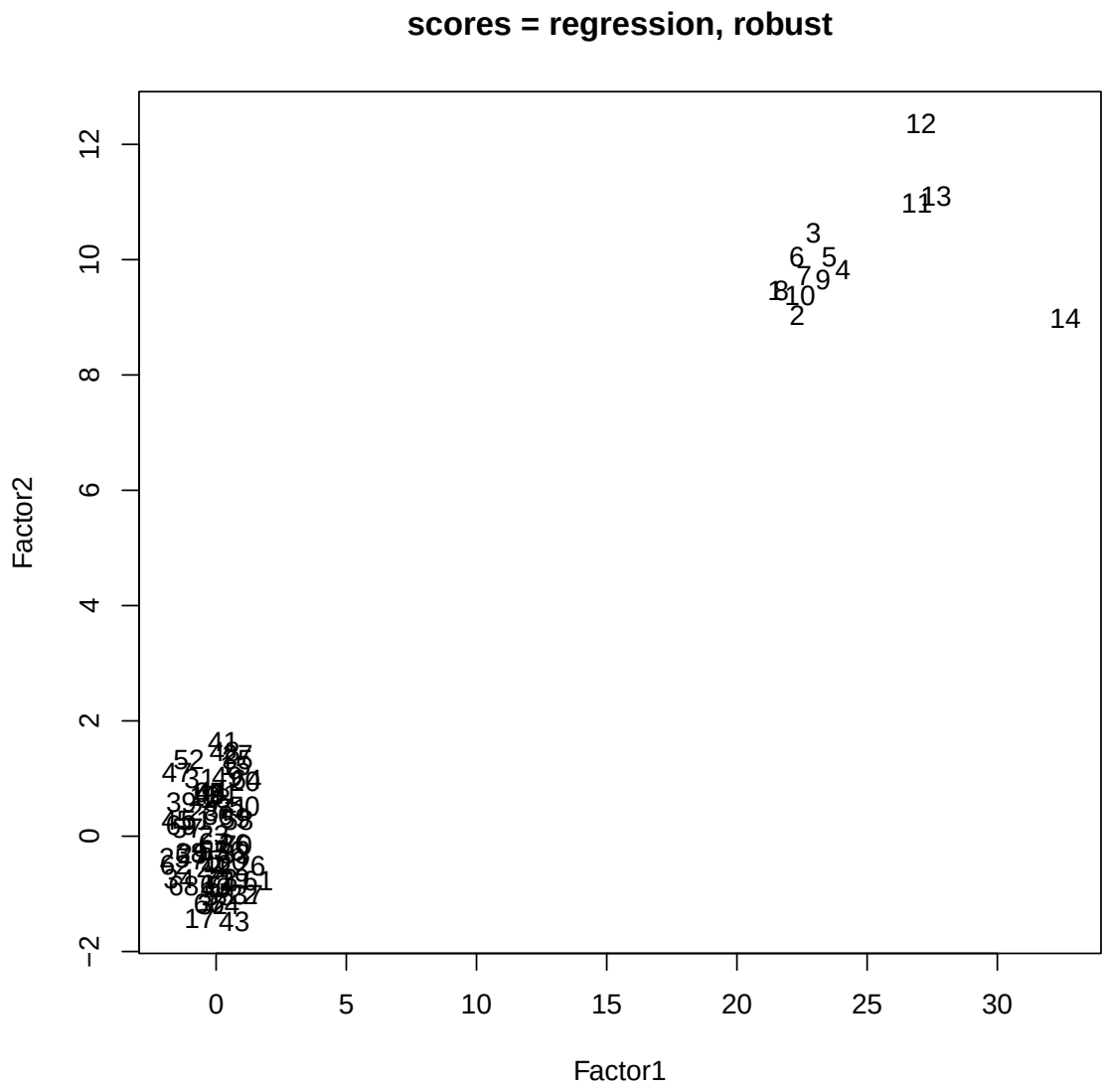

scores = regression, classical
Factor2
Factor1
2
1
0
−1
−1
0
1
2
3
4


scores = regression, classical
Factor2
Factor1
2
1
0
−1
−1
0
1
2
3
4


scores = regression, robust
Factor2
Factor1
0
5
10
15
20
25
30
12
11
13
3
6
5
4
7
9
14
10
2


scores = regression, robust
Factor2
Factor1
12
10
8
6
4
2
0
−2
0
5
10
15
20
25
30
12
13
11
3
6
5
4
7
9
1
8
10
2
14
41
52
47
17
43


scores = regression, robust
Factor2
Factor1
0
5
10
15
20
25
0
2
4
6



3.4. Example: hbk data

In this subsection, a data set `hbk` is used to show the base functionalities of the robust factor analysis solution. The Hawkins, Bradu and Kass data set `hbk` is from the package **robustbase** consists of 75 observations in 4 dimensions (one response and three explanatory variables). The first 10 observations are bad leverage points, and the next four points are good leverage points (i.e., their `x` are outlying, but the corresponding `y` fit the model quite well). We will consider only the X-part of the data.

Robust factor analysis is represented by the class `FaCov` which inherits from class `Fa` by class `FaRobust` of distance 2, and uses all slots and methods defined from `Fa`. The function `FaCov()` serves as a constructor (generating function) of the class. It can be called either by providing a data frame or matrix or a formula with no response variable, referring only to numeric

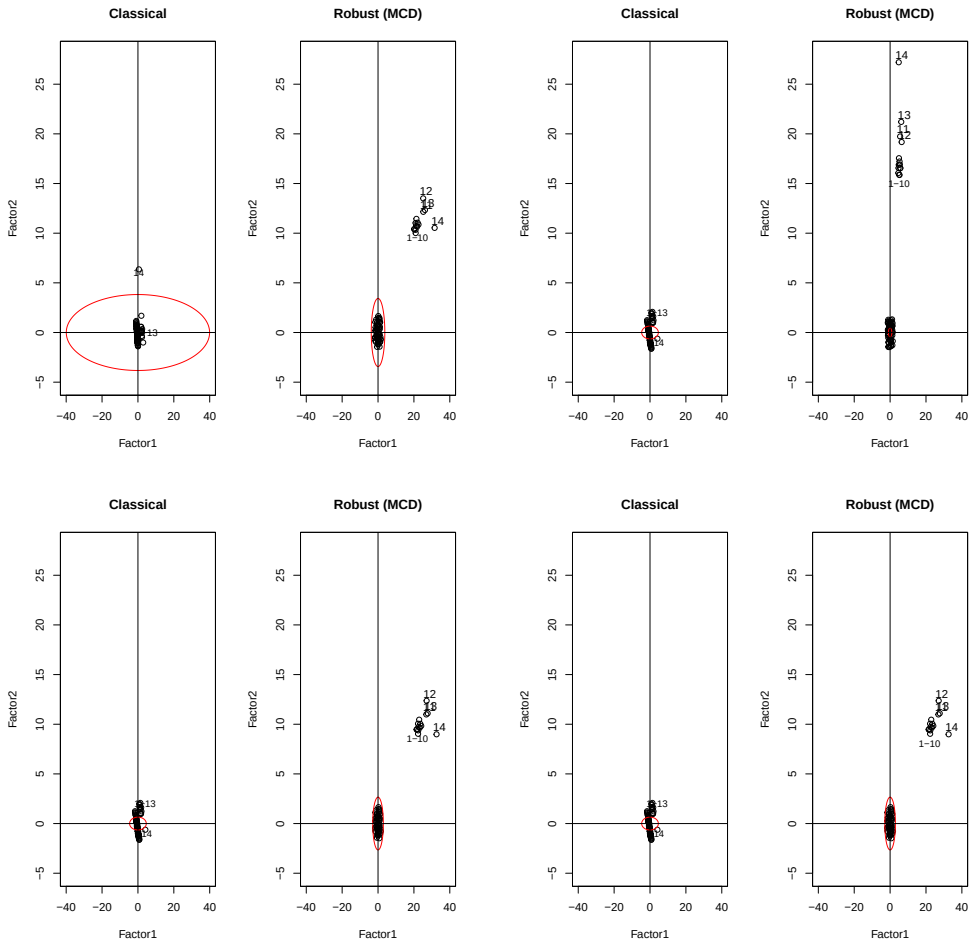


Figure 7: Classical and robust (MCD) scatterplots of the first two factor scores of the hbk data with 97.5% tolerance ellipses. First row: (1) vs (5); (3) vs (7). Second row: (2) vs (6); (4) vs (8).

variables. The usage of the default method of `FaCov` is:

```
FaCov(x, factors = 2, cor = FALSE, cov.control = rrcov::CovControlMcd(),
method = c("mle", "pca", "pfa"),
scoresMethod = c("none", "regression", "Bartlett"), ...)
```

where `x` is a numeric matrix or an object that can be coerced to a numeric matrix. `factors` is the number of factors to be fitted. `cor` is a logical value indicating whether the calculation should use the correlation matrix (`cor = TRUE`) or the covariance matrix (`cor = FALSE`). `cov.control` specifies which covariance estimator to use by providing a `CovControl-class` object. The default is `CovControlMcd-class` which will indirectly call `CovMcd`. If `cov.control = NULL` is specified, the classical estimates will be used by calling `CovClassic`. `method` is the method of factor analysis, one of `"mle"` (the default), `"pca"`, and `"pfa"`. `scoresMethod` specifies which type of scores to produce. The default is `"none"`, `"regression"` gives Thompson's scores, and `"Bartlett"` gives Bartlett's weighted least-squares scores (Johnson and Wichern 2007; Xue and Chen 2007). The usage of the formula interface of `FaCov` is:

```
FaCov(formula, data = NULL, factors = 2, cor = FALSE,
method = "mle", scoresMethod = "none", ...)
```

where `formula` is a formula with no response variable, referring only to numeric variables. `data` is an optional data frame containing the variables in the `formula`. Classical factor analysis is represented by the class `FaClassic` which inherits directly from `Fa`, and uses all slots and methods defined there. The function `FaClassic()` serves as a constructor (generating function) of the class. It also has its default method and formula interface as `FaCov()`.

The code lines

```
##
## faCovPcaRegMcd is obtained from FaCov.default
##
faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
                    scoresMethod = "regression", cov.control = rrcov::CovControlMcd());
faCovPcaRegMcd
```

generate an object `faCovPcaRegMcd` of class `FaCov`. In fact, it is equivalent to use the formula interface

```
faCovForPcaRegMcd = FaCov(~., data = as.data.frame(hbk.x),
                       factors = 2, method = "pca", scoresMethod = "regression",
                       cov.control = rrcov::CovControlMcd())
```

That is `faCovForPcaRegMcd = faCovPcaRegMcd`. Type

```
class(faCovPcaRegMcd)

## [1] "FaCov"
## attr(,"package")
## [1] "robustfa"
```

We see that the class `FaCov` is defined in the package **robustfa**. For an object `obj` of class `Fa`, we have

`obj = print(obj) = myFaPrint(obj).`

Here `print()` is a S4 generic function, while `myFaPrint()` is an S3 function acting as a function definition for `setMethod` of the generic function `print`.

```
print(faCovPcaRegMcd)

## [1] "Call:\n FaCov(x = hbk.x, factors = 2, cov.control = rrcov::CovControlMcd(),  \n"
## [2] "Call:\n     method = \"pca\", scoresMethod = \"regression\") \n"
## [1] "Standard deviations:\n 1.39107203014516"
## [2] "Standard deviations:\n 1.26173192309765"
## [3] "Standard deviations:\n 1.17062612110883"
## [1] "Loadings:\n -0.00415024202396914"
## [2] "Loadings:\n 1.18079786613697"
## [3] "Loadings:\n 0.64616966781657"
## [4] "Loadings:\n 1.2079697328235"
## [5] "Loadings:\n -0.124696507700608"
## [6] "Loadings:\n 0.490380137828612"
```

From Figure 2 we see that `summary()` generates an object of class `SummaryFa` which has its own `print()` method.

```
summaryFaCovPcaRegMcd = summary(faCovPcaRegMcd);
summaryFaCovPcaRegMcd

## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaCov"
## Slot "call":
## FaCov(x = hbk.x, factors = 2, cov.control = rrcov::CovControlMcd(),
##     method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##         Factor1    Factor2
## X1 -0.004150242  1.2079697
## X2  1.180797866 -0.1246965
## X3  0.646169668  0.4903801
##
## Slot "communality":
##        X1        X2        X3
## 1.4592081 1.4098328 0.6580079
```

```
##
## Slot "uniquenesses":
##        X1        X2        X3
## 0.1935459 0.2724395 0.9043802
##
## Slot "cor":
## [1] FALSE
##
## Slot "covariance":
##            X1         X2        X3
## X1 1.65275399 0.07409893 0.1713059
## X2 0.07409893 1.68227229 0.2054714
## X3 0.17130591 0.20547136 1.5623881
##
## Slot "correlation":
##            X1         X2        X3
## X1 1.00000000 0.04443853 0.1066041
## X2 0.04443853 1.00000000 0.1267385
## X3 0.10660407 0.12673854 1.0000000
##
## Slot "usedMatrix":
##            X1         X2        X3
## X1 1.65275399 0.07409893 0.1713059
## X2 0.07409893 1.68227229 0.2054714
## X3 0.17130591 0.20547136 1.5623881
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##        Factor1     Factor2
## 1  20.13910700 10.41054843
## 2  20.97706402 10.01367633
## 3  21.39809904 11.44059122
```

```
## 4  22.54842888 10.87239607
## 5  22.05589849 11.05891277
## 6  20.92380674 11.02737636
## 7  21.24256223 10.72056381
## 8  20.31926457 10.39535479
## 9  21.79701349 10.66168792
## 10 20.92614819 10.33613015
## 11 25.23594154 12.15088328
## 12 25.17211484 13.50577832
## 13 26.16372927 12.33480781
## 14 31.54346390 10.53262812
## 15  0.75687531  1.27562475
## 16 -0.28837150  0.70092910
## 17 -0.51292895 -1.44572060
## 18 -0.06627823  0.64585062
## 19  0.76392511 -0.69909928
## 20  1.14234504  1.01958952
## 21  0.27877233  0.75180530
## 22  1.08997319 -0.94813829
## 23  0.01837079  0.03247537
## 24 -0.03489873 -0.54117005
## 25 -1.57564945 -0.46595942
## 26  1.32309697 -0.45410754
## 27  0.76477451  1.46224032
## 28 -0.56413583  0.39965765
## 29 -0.85789449 -0.34506095
## 30 -0.12343120  0.34233173
## 31 -0.58615914  0.98779924
## 32  0.01828578 -1.17413897
## 33  0.81658802 -0.25429589
## 34 -1.43546222 -0.81620435
## 35  0.74509265  1.35913570
## 36  0.64591896 -0.11111270
## 37  1.21070910 -0.95633572
## 38 -0.87886235 -0.32511853
## 39 -1.38233822  0.51890327
## 40  0.05285613 -0.89106624
## 41  0.16039011  1.65610198
## 42 -0.09752445 -0.51593638
## 43  0.84287084 -1.42887409
## 44 -0.36308318  0.74525270
## 45 -1.50269806  0.21079062
## 46 -0.42979742  0.69447536
## 47 -1.50952957  1.04332132
## 48  0.21380517  1.46828518
## 49  0.44457525  1.06621669
```

```
## 50  1.04423905  0.56848741
## 51 -0.66640311  0.25327565
## 52 -1.06410434  1.28986302
## 53 -0.29584733 -0.28174228
## 54  0.37663553 -0.82959015
## 55  0.21842851 -1.02180698
## 56  0.66279829 -0.20706611
## 57 -1.06418935  0.08324869
## 58  0.95564703  0.34438945
## 59  0.64614951  0.33403943
## 60  0.43467296 -0.43182281
## 61  1.58972340 -0.69615034
## 62 -1.53590966 -0.57606026
## 63  0.14404560 -0.81404776
## 64  1.13025908  1.03867946
## 65 -0.41041887  0.73315897
## 66 -0.19333630 -1.17853900
## 67 -0.02068962 -0.10931287
## 68 -1.28197580 -0.92785516
## 69 -1.27740073  0.13747469
## 70  0.71931918 -0.13140385
## 71  0.63817753  0.52815496
## 72  0.14624152 -0.74383175
## 73  0.20056649 -0.73838550
## 74  0.35253637 -1.17214139
## 75 -0.52934719 -0.45946294
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                  X1         X2        X3
## Factor1 -0.06685977  0.6641094 0.3335712
## Factor2  0.71068488 -0.1364369 0.2538867
##
## Slot "meanF":
##  Factor1  Factor2
## 4.272569 2.072818
##
## Slot "corF":
##           Factor1   Factor2
## Factor1 1.0000000 0.9730208
## Factor2 0.9730208 1.0000000
##
## Slot "STATISTIC":
## NULL
```

```
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
## Slot "center":
##       X1       X2       X3
## 1.537705 1.780328 1.686885
##
## Slot "eigenvalues":
## [1] 1.935081 1.591967 1.370366
##
## Slot "cov.control":
## An object of class "CovControlMcd"
## Slot "alpha":
## [1] 0.5
##
## Slot "nsamp":
## [1] 500
##
## Slot "scalefn":
## NULL
##
## Slot "maxcsteps":
## [1] 200
##
## Slot "seed":
## NULL
##
## Slot "use.correction":
## [1] TRUE
##
## Slot "trace":
## [1] FALSE
##
## Slot "tolSolve":
## [1] 1e-14
##
##
##
## Slot "importance":
##                Factor1 Factor2
## SS loadings      1.812   1.715
## Proportion Var   0.370   0.350
```

```
## Cumulative Var   0.370   0.720
```

From the `summary` result of `faCovPcaRegMcd`, we see that the first two factors accout for about 72.0% of its total variance.

Next we calculate prediction/scores using `predict()`. The usage is `predict(object, ...)`, where `object` is an object of class `Fa`. If missing `...`(`newdata`), the `scores` slot of `object` is used. To save space, only the first five and last five rows of the scores are displayed.

```
predict(faCovPcaRegMcd)
```

```
##       Factor1    Factor2
## 1  20.1391070 10.4105484
## 2  20.9770640 10.0136763
## 3  21.3980990 11.4405912
## 4  22.5484289 10.8723961
## 5  22.0558985 11.0589128
## 71  0.6381775  0.5281550
## 72  0.1462415 -0.7438318
## 73  0.2005665 -0.7383855
## 74  0.3525364 -1.1721414
## 75 -0.5293472 -0.4594629
```

If not missing `...`, then `...` must have the name `newdata`. Moreover, `newdata` should be scaled first. For example, the original data is `x = hbk.x`, and `newdata` is a one row `data.frame`.

```
newdata = hbk.x[1, ]
cor = FALSE # the default
newdata = { if (cor == TRUE)
# standardized transformation
scale(newdata, center = faCovPcaRegMcd@center,
  scale = sqrt(diag(faCovPcaRegMcd@covariance)))
else # cor == FALSE
# centralized transformation
scale(newdata, center = faCovPcaRegMcd@center, scale = FALSE)
}
```

Finally we get

```
prediction = predict(faCovPcaRegMcd, newdata = newdata)
prediction

##    Factor1  Factor2
## 1 20.13911 10.41055
```

One can easily check that

`prediction = predict(faCovPcaRegMcd)[1,] = faCovPcaRegMcd@scores[1,]`

To visualize the factor analysis results, the `plot` method can be used. We have `setMethod plot` with signature

`x = "Fa", y = "missing".`

The usage is

`plot(x, which = c("factorScore", "screeplot"), choices = 1:2).`

Where `x` is an object of class `Fa`. The argument `which` indicates what kind of plot. If `which = "factorScore"`, then a scatterplot of the factor scores is produced; if `which = "screeplot"`, then the eigenvalues plot is created and is helpful to select the number of factors. The argument `choices` is an integer vector of length two indicating which columns of the factor scores to plot. To see how `plot` is functioning, we first generate an `Fa` object.

```
faClassicPcaReg = FaClassic(x = hbk.x, factors = 2, method = "pca",
                        scoresMethod = "regression"); faClassicPcaReg

## An object of class "FaClassic"
## Slot "call":
## FaClassic(x = hbk.x, factors = 2, method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##      Factor1  Factor2
## X1  3.411036 0.936662
## X2  7.278529 3.860723
## X3 11.270812 3.273734
##
## Slot "communality":
##        X1        X2        X3
##  12.51250  67.88217 137.74853
##
## Slot "uniquenesses":
##           X1           X2           X3
## 0.8292095832 0.0007959085 0.0863235416
##
## Slot "cor":
## [1] FALSE
##
## Slot "covariance":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
```

```
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "correlation":
##           X1        X2        X3
## X1 1.0000000 0.9459958 0.9617790
## X2 0.9459958 1.0000000 0.9786612
## X3 0.9617790 0.9786612 1.0000000
##
## Slot "usedMatrix":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##         Factor1      Factor2
## 1   1.836216178  0.161337548
## 2   1.756800683  0.549735006
## 3   2.250318962 -0.462115084
## 4   2.110379324  0.145591275
## 5   2.095218367  0.066345803
## 6   1.906582805  0.232737572
## 7   1.788199906  0.587263155
## 8   1.911901278  0.021274118
## 9   2.106202931 -0.053757894
## 10  2.122518267 -0.342046036
## 11  2.348800415  0.342830312
## 12  2.927387983 -1.009336488
## 13  1.905818664  1.685956006
## 14  0.528829255  6.359573459
```

```
## 15 -0.448347445  0.133780216
## 16 -0.668908585  0.366404244
## 17 -0.779858578  0.442576537
## 18 -0.320380311 -0.436151460
## 19 -0.697421067  0.621061863
## 20 -0.531502346  0.422359906
## 21 -0.418046839 -0.099159644
## 22 -0.718663795  0.742356037
## 23 -0.665393822  0.394816805
## 24 -0.761582949  0.580540198
## 25 -0.419693120 -0.657298051
## 26 -0.598455877  0.539385243
## 27 -0.247874654 -0.345083251
## 28 -0.219836054 -0.829109302
## 29 -0.484797036 -0.302159906
## 30  0.006043560 -1.273196784
## 31 -0.414104099 -0.319367475
## 32 -0.862788836  0.802561432
## 33 -0.704564559  0.680970529
## 34 -0.404005807 -0.681812987
## 35 -0.226207238 -0.410125033
## 36 -0.370021556 -0.176933456
## 37 -0.581066520  0.435007090
## 38 -0.621218623  0.029086231
## 39 -0.210664210 -1.057488778
## 40 -0.638509599  0.278324368
## 41 -0.093737402 -0.869315133
## 42 -0.222301393 -0.760169894
## 43 -0.929819437  1.166157588
## 44  0.040846868 -1.395395068
## 45 -0.345171409 -0.777979662
## 46  0.005618871 -1.329449176
## 47 -0.241826492 -0.980332711
## 48 -0.094700744 -0.865086128
## 49 -0.436399538  0.009447513
## 50 -0.371498956 -0.025665034
## 51 -0.585765982  0.034545544
## 52 -0.234984713 -0.865075764
## 53 -0.017987529 -1.299228393
## 54 -0.924264023  1.069378350
## 55 -0.757104843  0.604723754
## 56 -0.808109523  0.898476392
## 57 -0.432379728 -0.457331137
## 58 -0.698147469  0.739975084
## 59 -0.269779011 -0.394725685
## 60 -0.198808958 -0.673656887
```

```
## 61 -0.557786430  0.493393901
## 62 -0.459937878 -0.555054318
## 63 -0.892563087  0.931740886
## 64 -0.361028449  0.001270686
## 65 -0.067714855 -1.141591364
## 66 -0.694666313  0.333677831
## 67 -0.570392680  0.141917212
## 68 -0.352127153 -0.776502820
## 69 -0.474012517 -0.407110790
## 70 -0.340986005 -0.230478698
## 71 -0.421675567 -0.010838417
## 72 -0.579475155  0.167004879
## 73 -0.425439569 -0.197214204
## 74 -0.631892463  0.322176109
## 75 -0.141285522 -1.068417769
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                   X1         X2         X3
## Factor1  0.06192447 -0.1643831  0.1761400
## Factor2 -0.12400651  0.5687014 -0.3297295
##
## Slot "meanF":
##       Factor1       Factor2
## -1.368424e-16  2.902366e-16
##
## Slot "corF":
##             Factor1     Factor2
## Factor1 1.00000e+00 1.62519e-15
## Factor2 1.62519e-15 1.00000e+00
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
## Slot "center":
##       X1       X2       X3
## 3.206667 5.597333 7.230667
##
```

```
## Slot "eigenvalues":
## [1] 216.162129   1.981077   0.916329
##
## Slot "cov.control":
## NULL

summary(faClassicPcaReg)

## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaClassic"
## Slot "call":
## FaClassic(x = hbk.x, factors = 2, method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##      Factor1  Factor2
## X1  3.411036 0.936662
## X2  7.278529 3.860723
## X3 11.270812 3.273734
##
## Slot "communality":
##        X1        X2        X3
##  12.51250  67.88217 137.74853
##
## Slot "uniquenesses":
##           X1           X2           X3
## 0.8292095832 0.0007959085 0.0863235416
##
## Slot "cor":
## [1] FALSE
##
## Slot "covariance":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "correlation":
##           X1        X2        X3
## X1 1.0000000 0.9459958 0.9617790
## X2 0.9459958 1.0000000 0.9786612
## X3 0.9617790 0.9786612 1.0000000
##
```

```
## Slot "usedMatrix":
##          X1       X2        X3
## X1 13.34171 28.46921  41.24398
## X2 28.46921 67.88297  94.66562
## X3 41.24398 94.66562 137.83486
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##         Factor1      Factor2
## 1   1.836216178  0.161337548
## 2   1.756800683  0.549735006
## 3   2.250318962 -0.462115084
## 4   2.110379324  0.145591275
## 5   2.095218367  0.066345803
## 6   1.906582805  0.232737572
## 7   1.788199906  0.587263155
## 8   1.911901278  0.021274118
## 9   2.106202931 -0.053757894
## 10  2.122518267 -0.342046036
## 11  2.348800415  0.342830312
## 12  2.927387983 -1.009336488
## 13  1.905818664  1.685956006
## 14  0.528829255  6.359573459
## 15 -0.448347445  0.133780216
## 16 -0.668908585  0.366404244
## 17 -0.779858578  0.442576537
## 18 -0.320380311 -0.436151460
## 19 -0.697421067  0.621061863
## 20 -0.531502346  0.422359906
## 21 -0.418046839 -0.099159644
## 22 -0.718663795  0.742356037
## 23 -0.665393822  0.394816805
```

```
## 24 -0.761582949  0.580540198
## 25 -0.419693120 -0.657298051
## 26 -0.598455877  0.539385243
## 27 -0.247874654 -0.345083251
## 28 -0.219836054 -0.829109302
## 29 -0.484797036 -0.302159906
## 30  0.006043560 -1.273196784
## 31 -0.414104099 -0.319367475
## 32 -0.862788836  0.802561432
## 33 -0.704564559  0.680970529
## 34 -0.404005807 -0.681812987
## 35 -0.226207238 -0.410125033
## 36 -0.370021556 -0.176933456
## 37 -0.581066520  0.435007090
## 38 -0.621218623  0.029086231
## 39 -0.210664210 -1.057488778
## 40 -0.638509599  0.278324368
## 41 -0.093737402 -0.869315133
## 42 -0.222301393 -0.760169894
## 43 -0.929819437  1.166157588
## 44  0.040846868 -1.395395068
## 45 -0.345171409 -0.777979662
## 46  0.005618871 -1.329449176
## 47 -0.241826492 -0.980332711
## 48 -0.094700744 -0.865086128
## 49 -0.436399538  0.009447513
## 50 -0.371498956 -0.025665034
## 51 -0.585765982  0.034545544
## 52 -0.234984713 -0.865075764
## 53 -0.017987529 -1.299228393
## 54 -0.924264023  1.069378350
## 55 -0.757104843  0.604723754
## 56 -0.808109523  0.898476392
## 57 -0.432379728 -0.457331137
## 58 -0.698147469  0.739975084
## 59 -0.269779011 -0.394725685
## 60 -0.198808958 -0.673656887
## 61 -0.557786430  0.493393901
## 62 -0.459937878 -0.555054318
## 63 -0.892563087  0.931740886
## 64 -0.361028449  0.001270686
## 65 -0.067714855 -1.141591364
## 66 -0.694666313  0.333677831
## 67 -0.570392680  0.141917212
## 68 -0.352127153 -0.776502820
## 69 -0.474012517 -0.407110790
```

```
## 70 -0.340986005 -0.230478698
## 71 -0.421675567 -0.010838417
## 72 -0.579475155  0.167004879
## 73 -0.425439569 -0.197214204
## 74 -0.631892463  0.322176109
## 75 -0.141285522 -1.068417769
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                   X1         X2         X3
## Factor1  0.06192447 -0.1643831  0.1761400
## Factor2 -0.12400651  0.5687014 -0.3297295
##
## Slot "meanF":
##       Factor1       Factor2
## -1.368424e-16  2.902366e-16
##
## Slot "corF":
##             Factor1     Factor2
## Factor1 1.00000e+00 1.62519e-15
## Factor2 1.62519e-15 1.00000e+00
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 75
##
## Slot "center":
##       X1       X2       X3
## 3.206667 5.597333 7.230667
##
## Slot "eigenvalues":
## [1] 216.162129   1.981077   0.916329
##
## Slot "cov.control":
## NULL
##
##
## Slot "importance":
##                Factor1 Factor2
```

```
## SS loadings    191.643  26.500
## Proportion Var   0.875   0.121
## Cumulative Var   0.875   0.996
```

`faClassicPcaReg` is an object of class `FaClassic`. From the `summary` result of `faClassicPcaReg`, we see that the first two factors accout for about 99.6% of its total variance. Next we generate an object `faCovPcaRegMcd` of class `FaCov` using the same data set. The `print` and `summary` results have already been shown before.

The following code lines generate classical and robust scatterplots of the first two factor scores. See Figure 8.

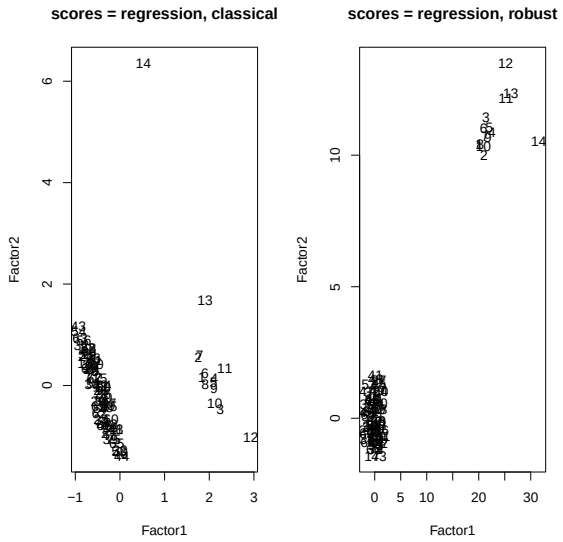


Figure 8: Classical and robust scatterplots of the first two factor scores of the hbk data.

The following code lines generate classical and robust scree plots. See Figure 9.

Next we impose a 97.5% tolerance ellipse on the scatterplot of the first two factor scores of the `hbk` data by using a function `rrcov:::.myellipse`. See Figure 10. The left panel shows the classical 97.5% tolerance ellipse, which is very large and contains the outliers 1-13. The regular points are not well seperated from the outliers. The right panel shows the robust 97.5% tolerance ellipse which clearly separates the regular points and outliers. We see that the estimate of the center of the regular points is located at the origin (where the mean of the scores should be located). The following code lines compute the means of the classical and robust scores. We see that the mean of all observations of the classical scores equals 0, while the mean of good observations (regular points, excluding the outliers) of the robust scores equals 0.

```
colMeans(faClassicPcaReg@scores)

##       Factor1       Factor2
## -1.369275e-16  2.901383e-16
```

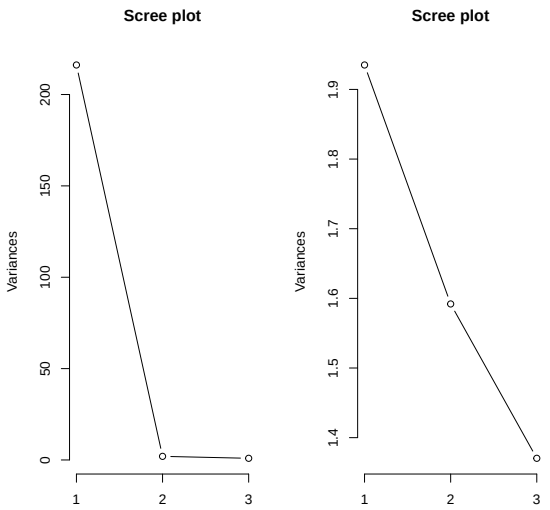


Figure 9: Classical and robust scree plots of the hbk data.

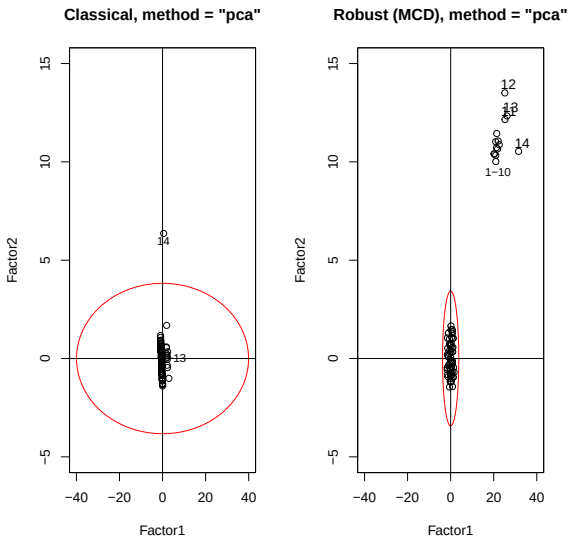


Figure 10: Classical and robust scatterplot of the first two factor scores of the hbk data with a 97.5% tolerance ellipse.

```
colMeans(faCovPcaRegMcd@scores)

##  Factor1  Factor2
## 4.272569 2.072818

colMeans(faClassicPcaReg@scores[15:75,])

##    Factor1    Factor2
## -0.4523799 -0.1358260

colMeans(faCovPcaRegMcd@scores[15:75,])

##       Factor1       Factor2
## -6.415633e-17  1.433280e-17
```

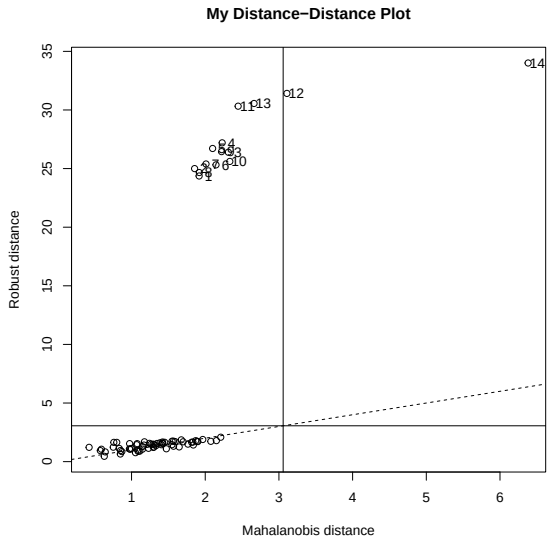
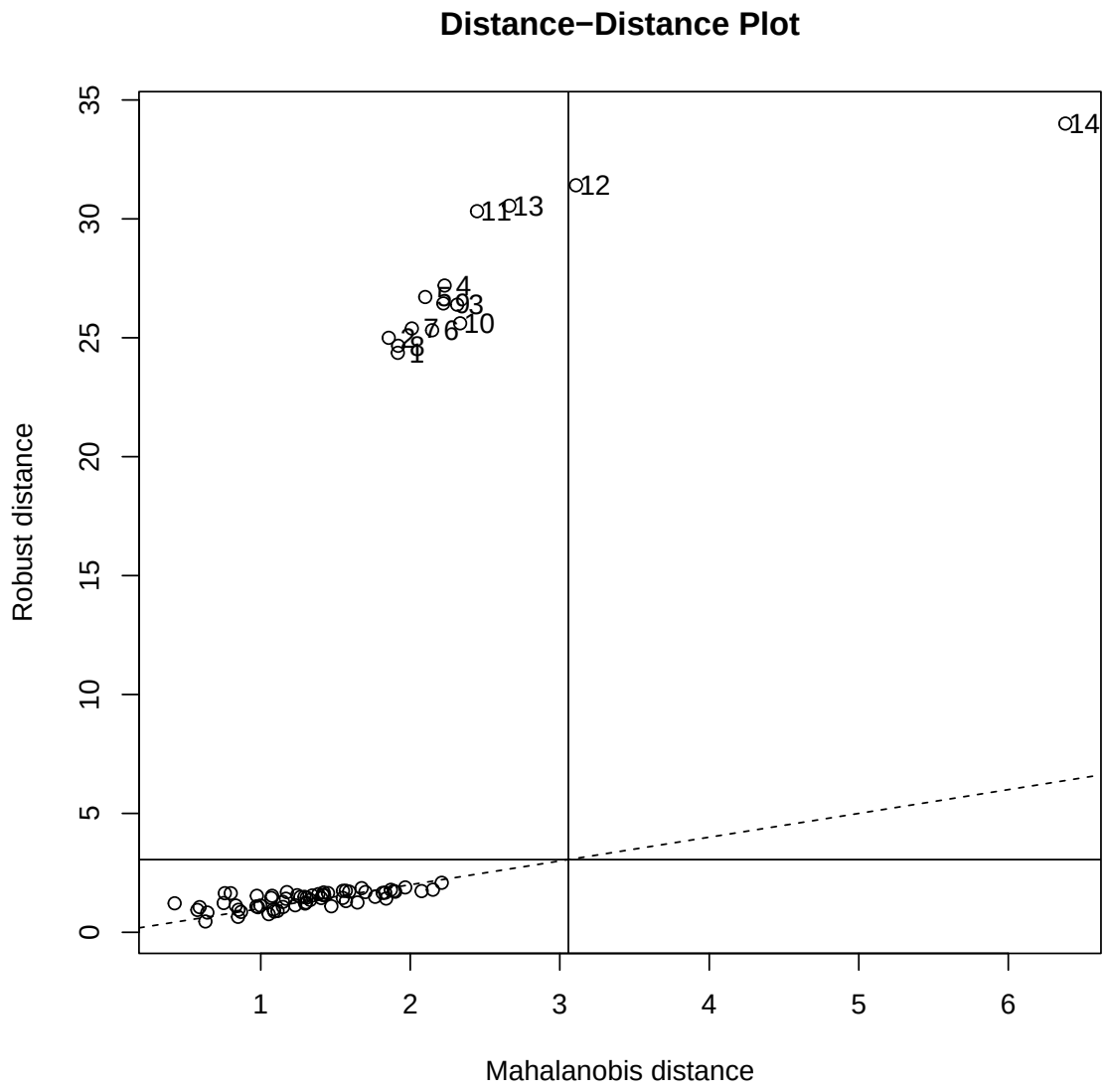
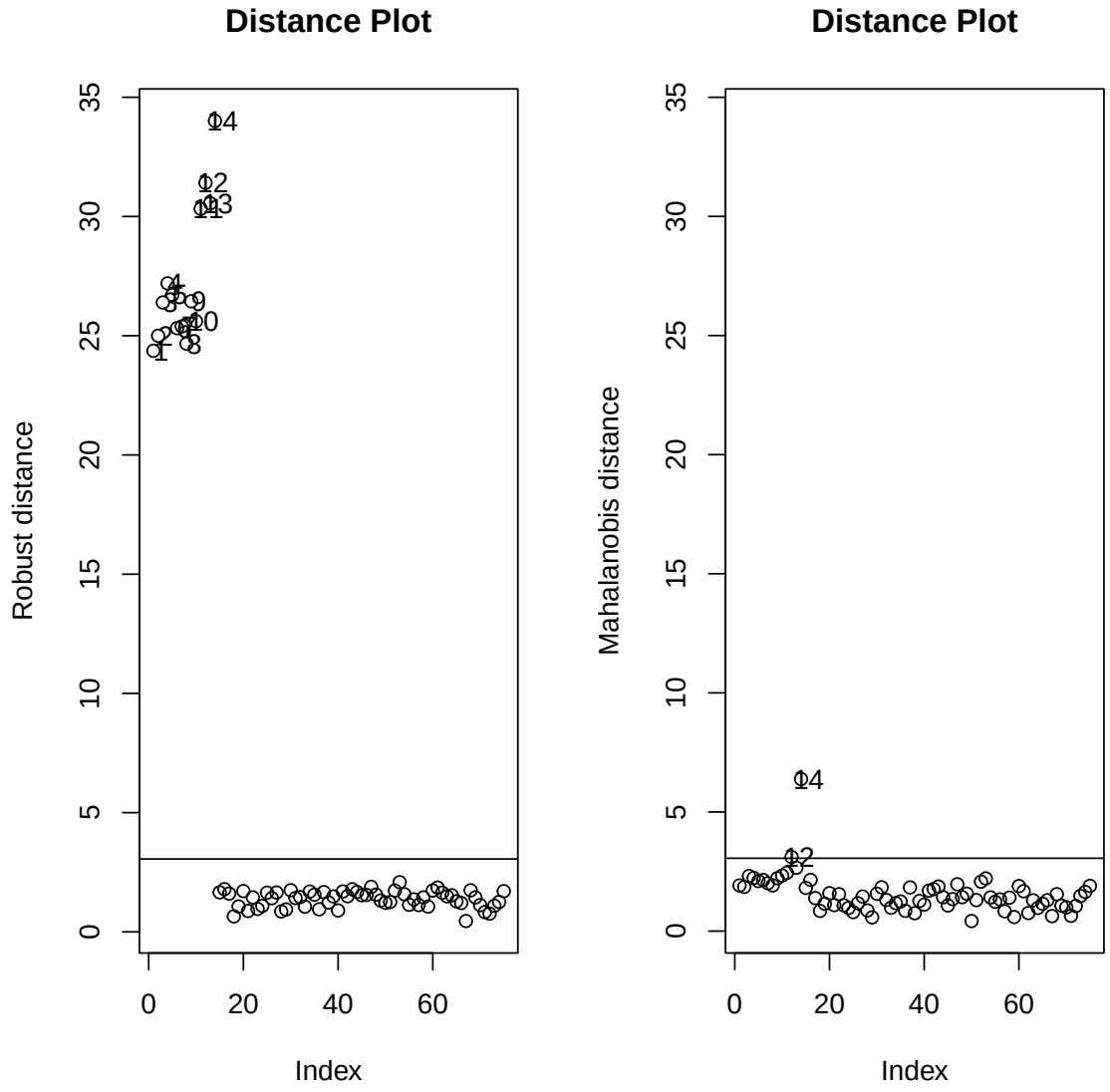
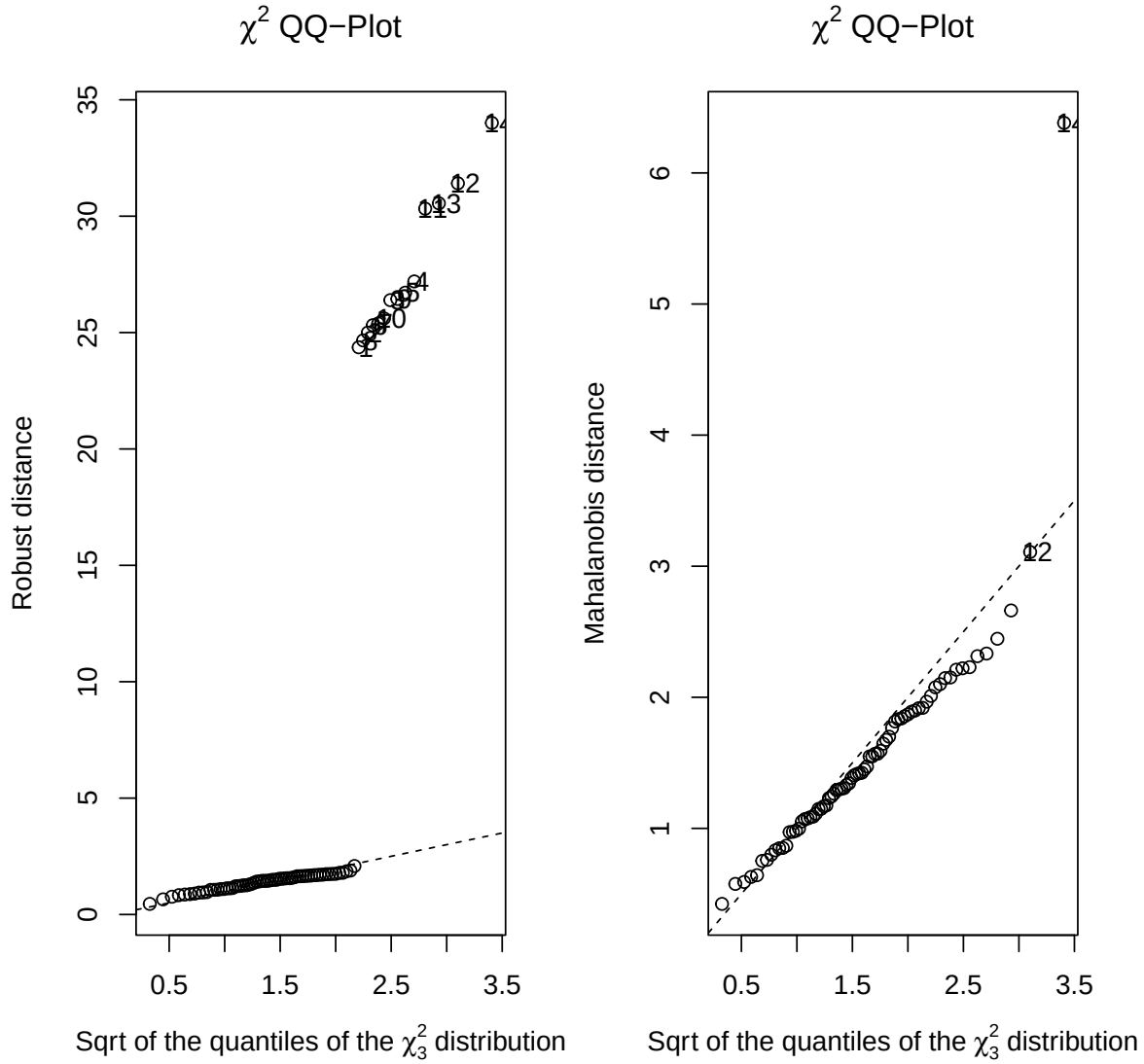


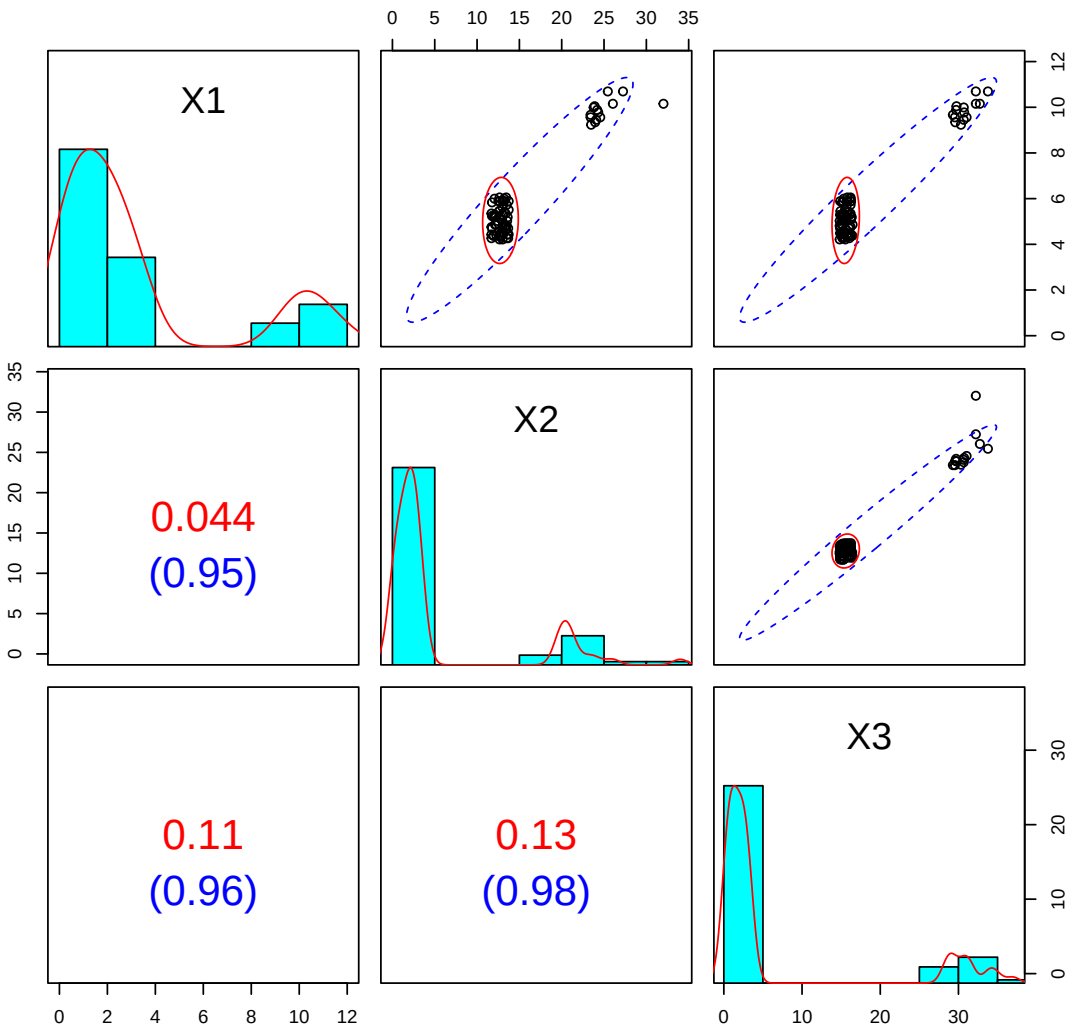
Figure 11: A distance-distance plot for hbk data.

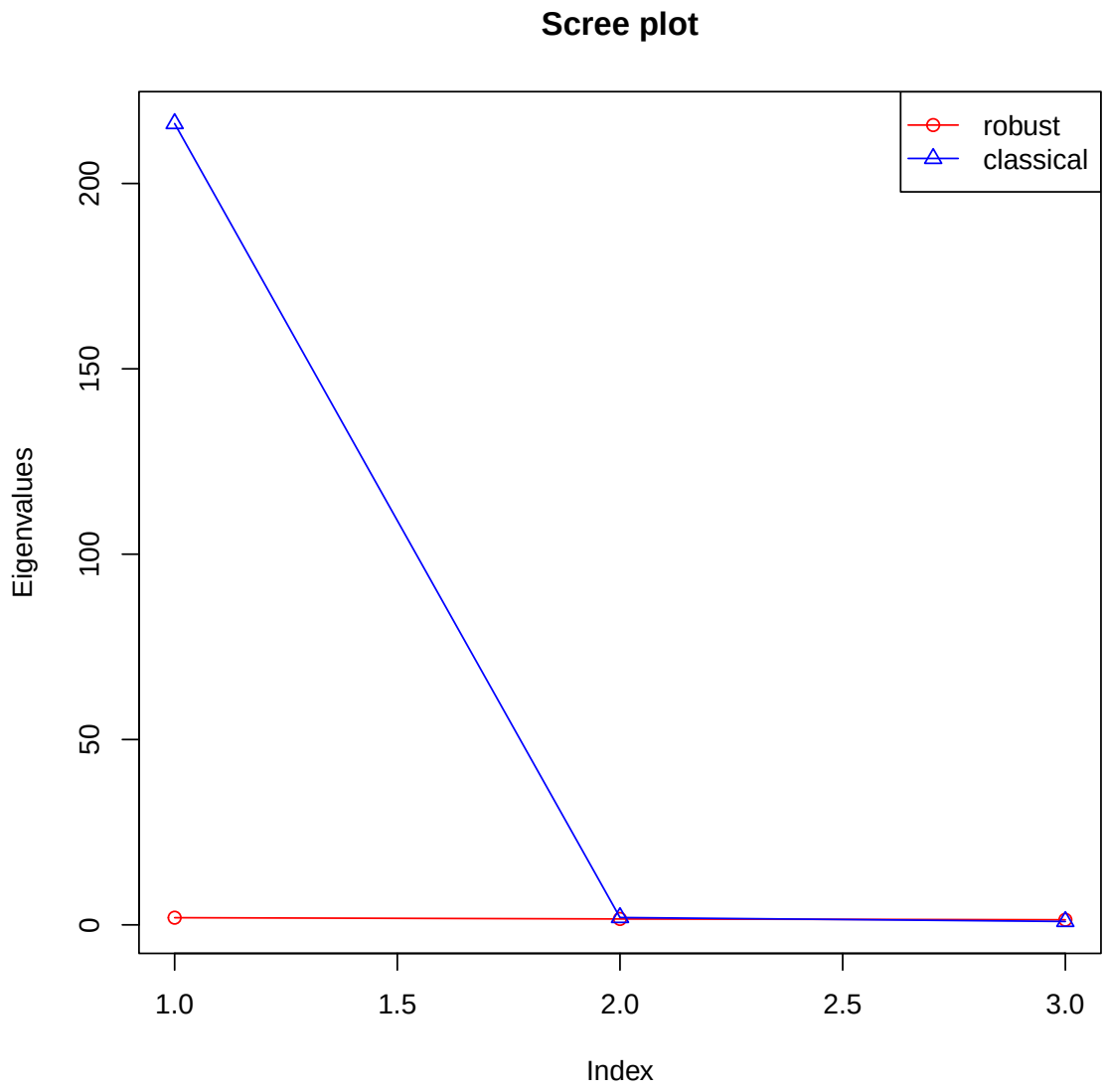

Distance−Distance Plot
Robust distance
Mahalanobis distance
0
5
10
15
20
25
30
35
1
2
3
4
5
6
14
12
13
11
10


Distance Plot
Robust distance
Index
0
5
10
15
20
25
30
35
0
20
40
60
Distance Plot
Mahalanobis distance
Index
0
5
10
15
20
25
30
35
0
20
40
60




```
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
## Warning in par(usr):  argument 1 does not name a graphical parameter
```


X1
X2
X3
0.044
(0.95)
0.11
(0.96)
0.13
(0.98)
0 5 10 15 20 25 30 35
0 2 4 6 8 10 12
0 10 20 30



Next we plot a `Cov-class`. See Figure **??**. The figure shows a distance-distance plot. We see that the robust (mahalanobis) distances are far larger than the (classical) mahalanobis distances. The outliers have large robust distances. The figure is generated by `myplotDD(x = covMcd)`. `myplotDD()` is a revised version of `.myddplot()` in `plot-utils.R` in the package **rrcov**. In `myplotDD()`, `id.n` and `ind` are printed out. Here `id.n` is the number of observations to identify by a label. By default, the number of observations with robust distances larger than `cutoff` is used. By default `cutoff = sqrt(qchisq(0.975, p))`. `ind` is the index of robust distances whose values are larger than `cutoff`.

```
## cutoff = 3.057516
## id.n <- length(which(rd>cutoff))
## id.n = 14
## Here y is the robust distance (rd).
```

```
## sort.y = (To save space, only the smallest five and largest five
## elements of sort.y$x and sort.y$ix are shown.)
## $x
##         67         18         72         71         28          4
##  0.4553524  0.6505517  0.7631246  0.8306446  0.8570290 27.1972578
##         11         13         12         14
## 30.3225470 30.5536038 31.4123574 34.0079508
##
## $ix
##  [1] 67 18 72 71 28  4 11 13 12 14
## ind =
##  [1]  1  8  2  6  7 10  3  9  5  4 11 13 12 14
```

From the above results we see that the `cutoff` is computed as 3.057516. There are `id.n = 14` observations with robust distance larger than `cutoff`. `sort.y` is a list containing the sorted values of `y` (the robust distance). `sort.y$x` is arranged in increasing order. To save space, only the smallest five and largest five robust distances with their indices are shown. `sort.y$ix` contains the indices. `ind` shows the indices of the largest `id.n = 14` observations whose robust distances are larger than `cutoff`.

The accessor functions `getCenter()`, `getEigenvalues()`, `getLoadings()`, `getQuan()`, `getScores()`, and `getSdev()` are used to access the corresponding slots. For instance

```
robustfa::getEigenvalues(faCovPcaRegMcd)

## [1] 1.935081 1.591967 1.370366
```

The function `getFa()`, which is used in `predict()` and `screeplot()`, returns a list of class `"fa"`.

Note that our previous comparisons in this subsection are just (1) vs (5), i.e., classical and robust factor analysis comparisons using `x = hbk.x` as the data input and the covariance matrix (`cor = FALSE`) as the running matrix.

For `x = hbk.x`, we checked that $\boldsymbol{S}^r \neq \tilde{\boldsymbol{S}}^r$, and $\boldsymbol{R}^r = \tilde{\boldsymbol{R}}^r$ for `control = "mcd"`, `"ogk"`, `"m"`, `"mve"`, `"sfast"`, `"bisquare"`, `"rocke"`, small differences between $\boldsymbol{R}^r$ and $\tilde{\boldsymbol{R}}^r$ for `control = "sde"`, `"surreal"`. The results illustrate Theorem 3.1.

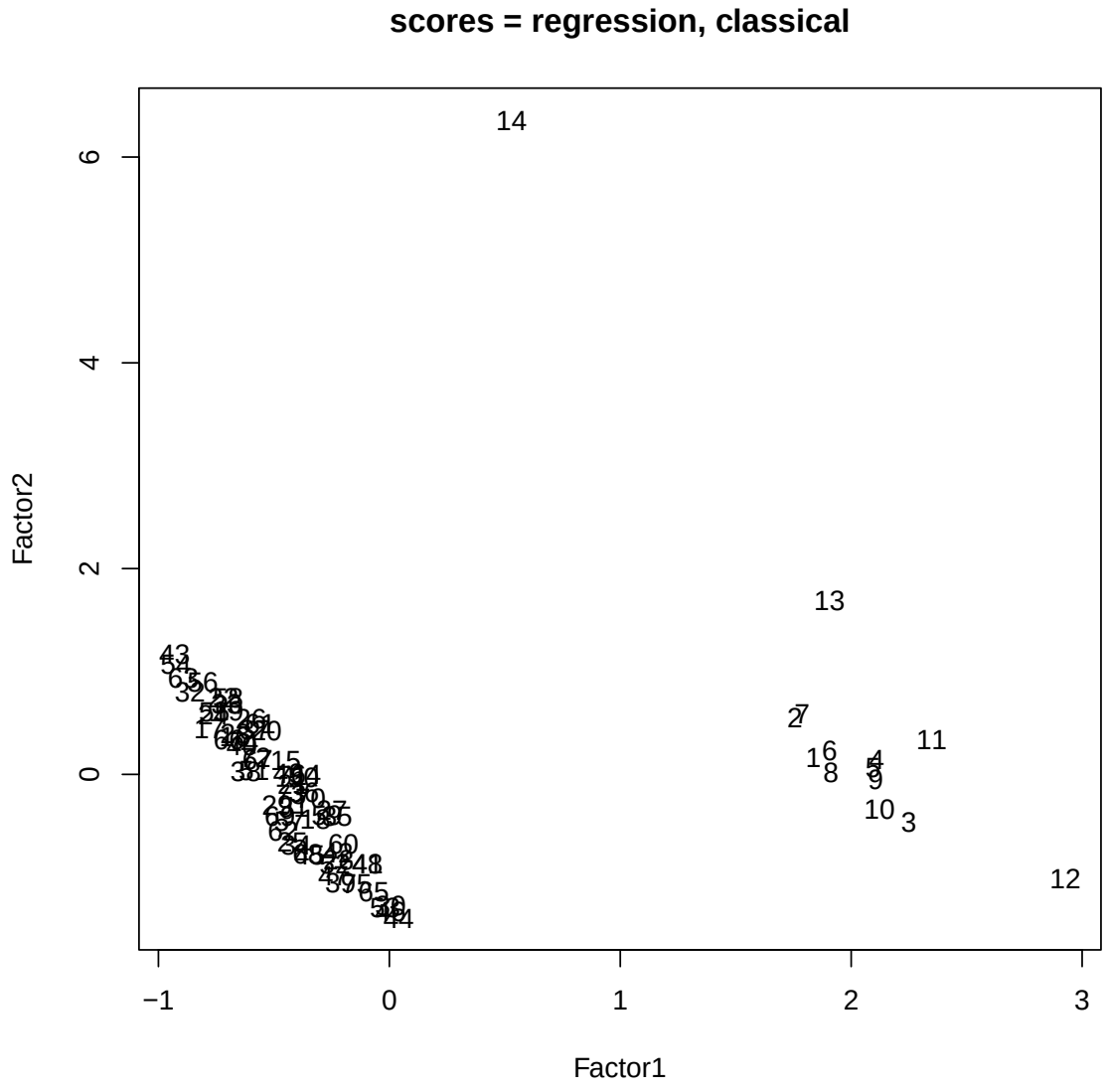
The eigenvalues of the running matrices of the `hbk` data of the 8 combinations are given in Table 6. From Table 6 we see that the eigenvalues of (2), (3), and (4) are the same, the eigenvalues of (6) and (8) are the same. The results illustrate Theorems 3.1 and 3.2.

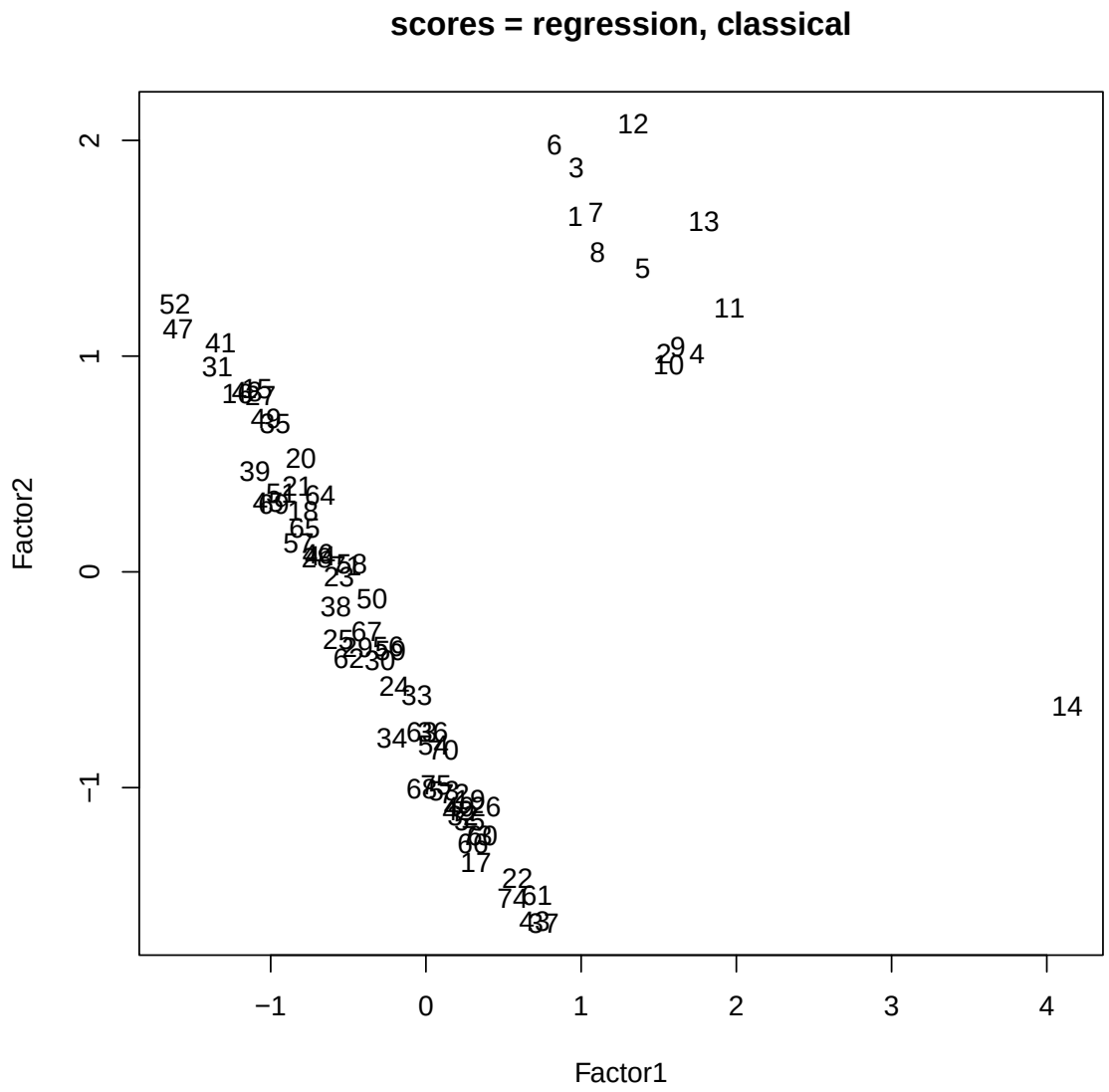
Classical and robust (MCD) scatterplots of the first two factor scores of the `hbk` data with 97.5% tolerance ellipses are ploted in Figure 12. We see that the scores of (2), (3), and (4) are the same, the scores of (6) and (8) are the same, in agree with Theorem 3.2. Note that the tolerance ellipse is very large for (1), since the outliers severely affected the eigenvalues of the running matrix $\boldsymbol{S}^c$. While the tolerance ellipses are very small for (2), (3), and (4), also due to the outliers severely affected the eigenvalues of the running matrices $\boldsymbol{R}^c = \tilde{\boldsymbol{S}}^c = \tilde{\boldsymbol{R}}^c$. The tolerance ellipse is very small for (7) and it does not cover the regular points, due to the first two eigenvalues of (7) are very small. It exemplifies that the results from robust

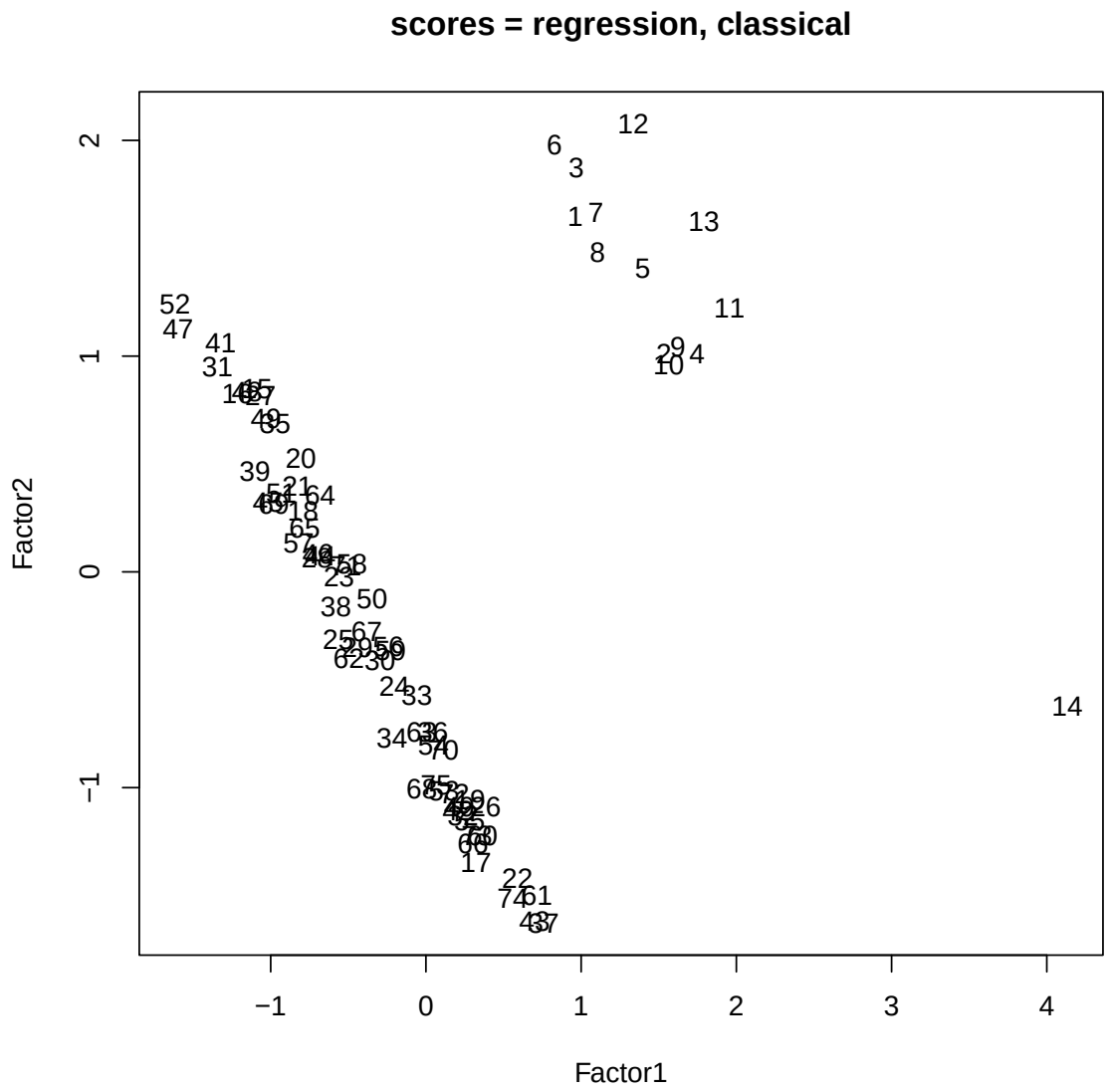
Table 6: The eigenvalues of the running matrices for hbk data.

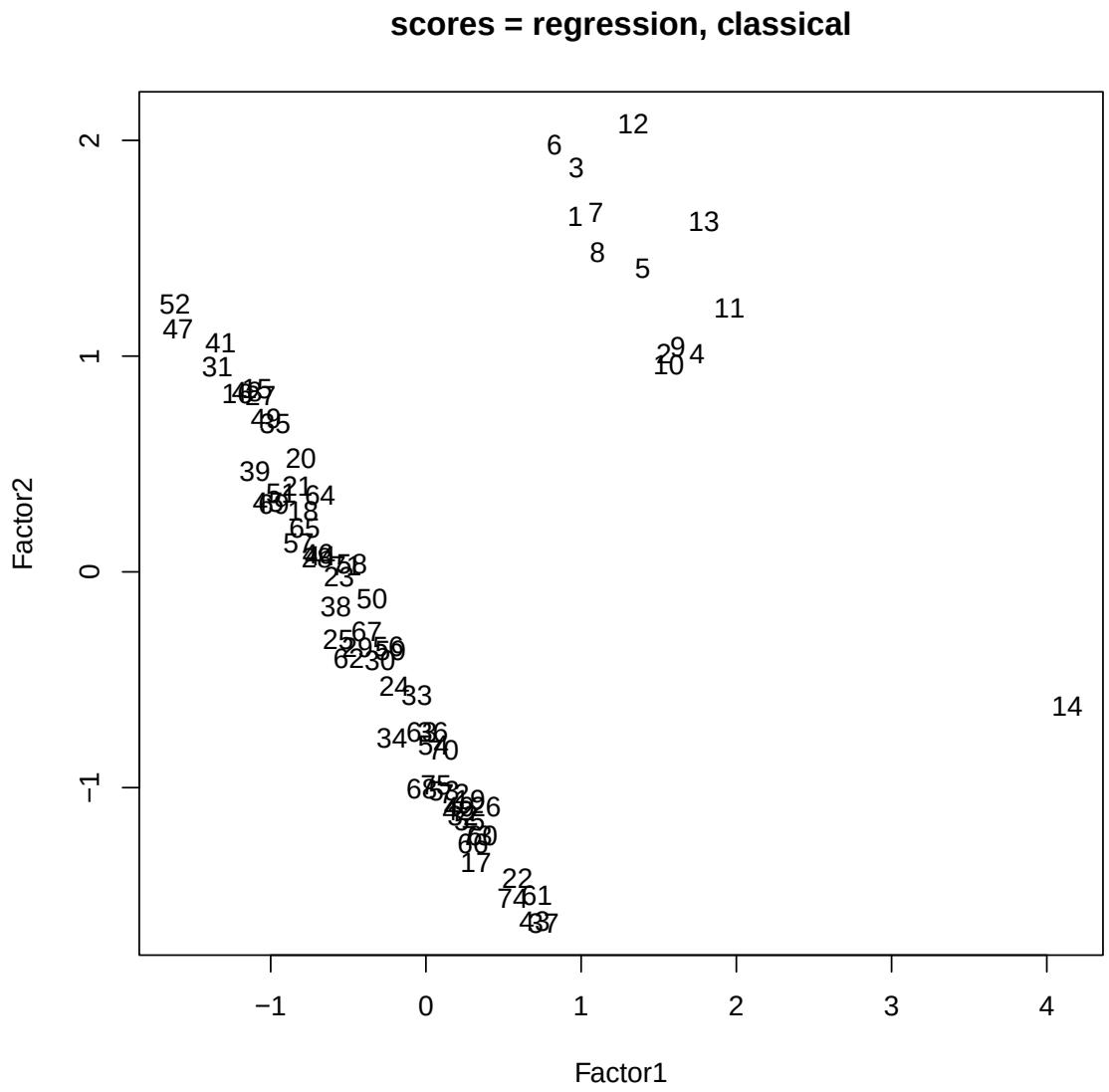
		Classical	Robust (MCD)
hbk.x	covariance	(1) 216.162129 1.981077 0.916329	(5) 1.935081 1.591967 1.370366
	correlation	(2) 2.92435228 0.05668483 0.01896288	(6) 1.1890834 0.9563255 0.8545911
scale(hbk.x)	covariance	(3) 2.92435228 0.05668483 0.01896288	(7) 0.12408511 0.02501676 0.01089401
	correlation	(4) 2.92435228 0.05668483 0.01896288	(8) 1.1890834 0.9563255 0.8545911

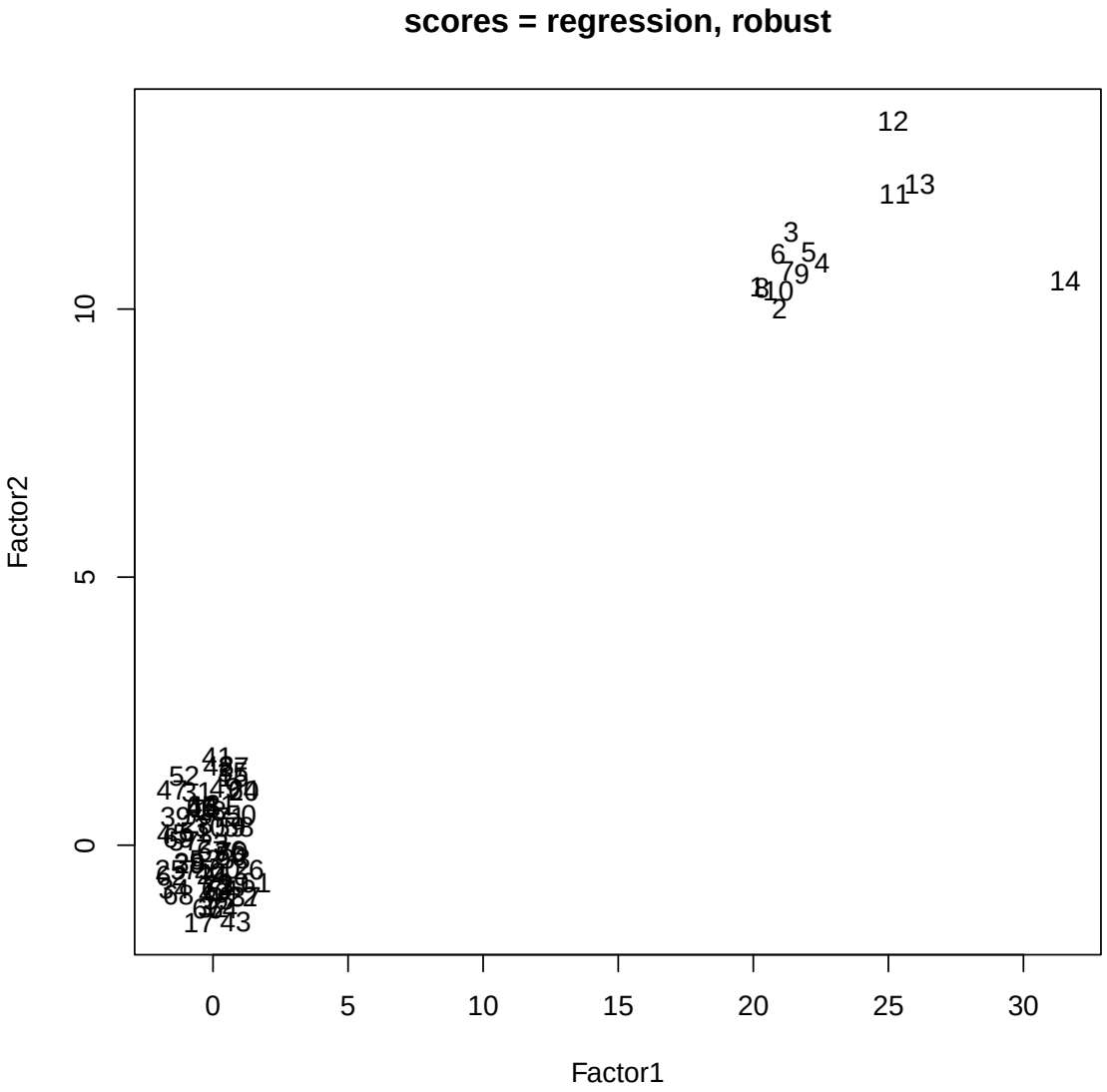
covariance matrix of the scaled data is not very reliable. The tolerance ellipses of (6) and (8) well seperate the regular points and the outliers.

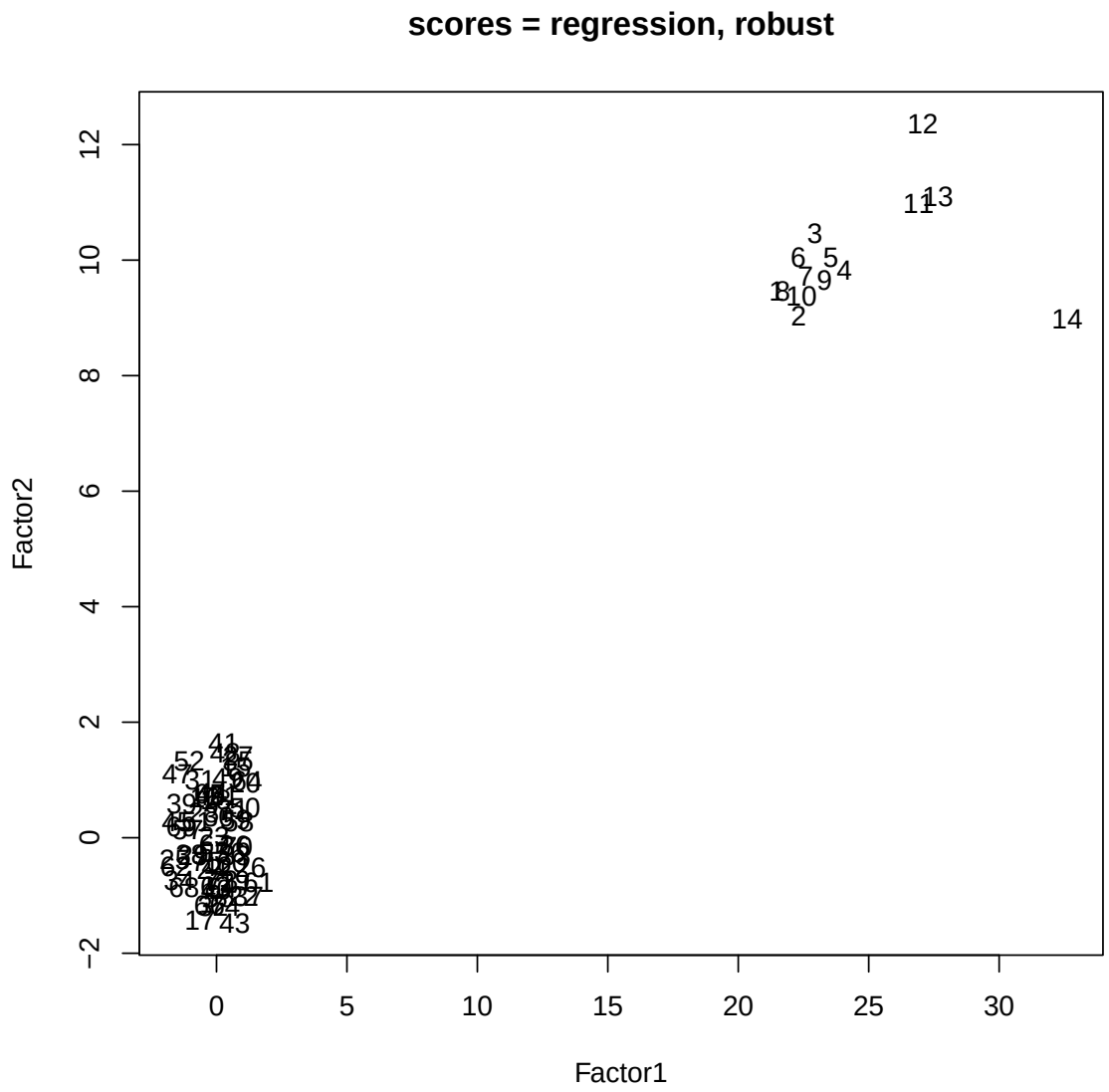


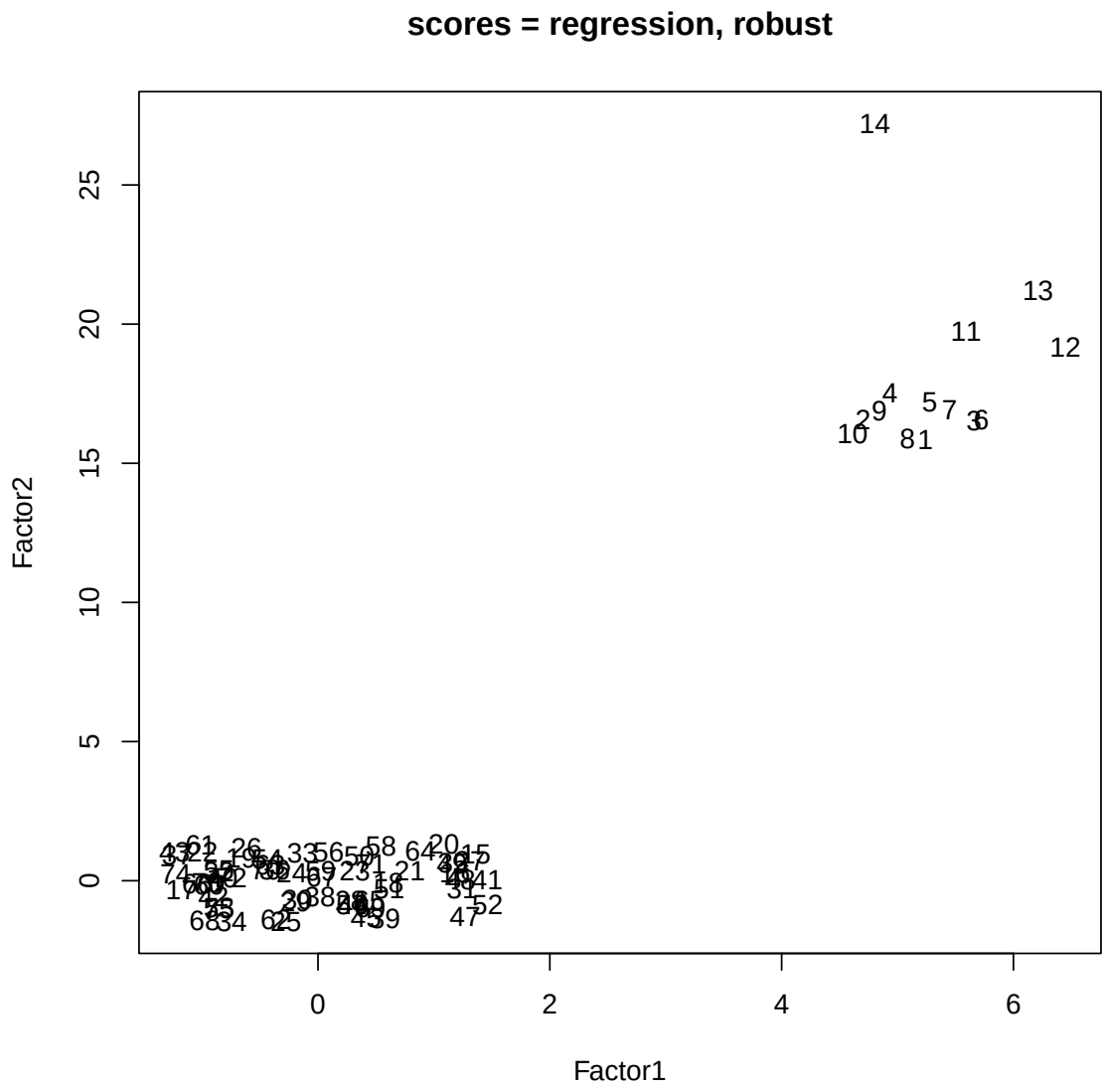

scores = regression, classical
Factor2
Factor1
2
1
0
−1
−1
0
1
2
3
4

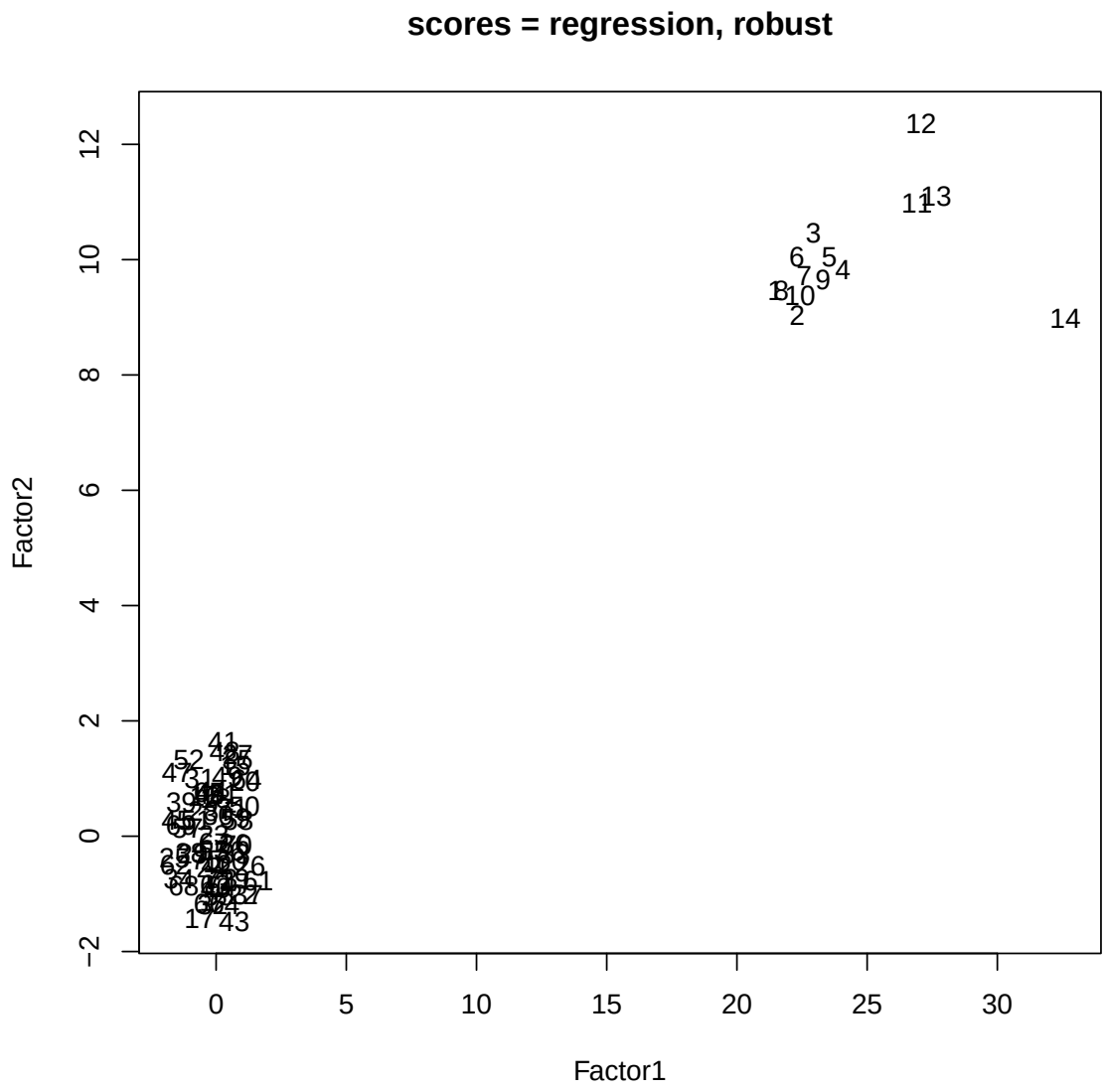

scores = regression, classical
Factor2
Factor1
−1
0
1
2
3
4


scores = regression, classical
Factor2
Factor1
−1
0
1
2
3
4
−1
0
1
2


scores = regression, robust
Factor2
Factor1
0
5
10
15
20
25
30
12
11
13
3
6
5
4
7
9
14
10
2


scores = regression, robust
Factor2
Factor1
12
13
11
3
6
5
4
7
9
10
2
14
0
5
10
15
20
25
30
−2
0
2
4
6
8
10
12


scores = regression, robust
Factor2
Factor1
0
5
10
15
20
25
0
2
4
6



3.5. Example: Stocks data

In this subsection, we apply the robust factor analysis solution to a real data set `stock611`. This data set consists of 611 observations with 12 variables. The data set is from Chinese stock market in the year 2001. It is used in (Wang 2009) to illustrate factor analysis methods. For `x = stock611[,3:12]`,

```
cov_x = rrcov::CovRobust(x = x, control = control)
```

gets error message for `control = "mcd", "m", "mve", "sde", "sfast"`, thus we can not compute `cov_x`, $\boldsymbol{S}^r$, and $\boldsymbol{R}^r$ for these robust estimators. That is, we can not get results for combinations (5) and (6) for these robust estimators. However, for `x = scale(stock611[,3:12])`, we can compute `cov_x`, $\tilde{\boldsymbol{S}}^r$, and $\tilde{\boldsymbol{R}}^r$ for these robust estimators, and we can get results for

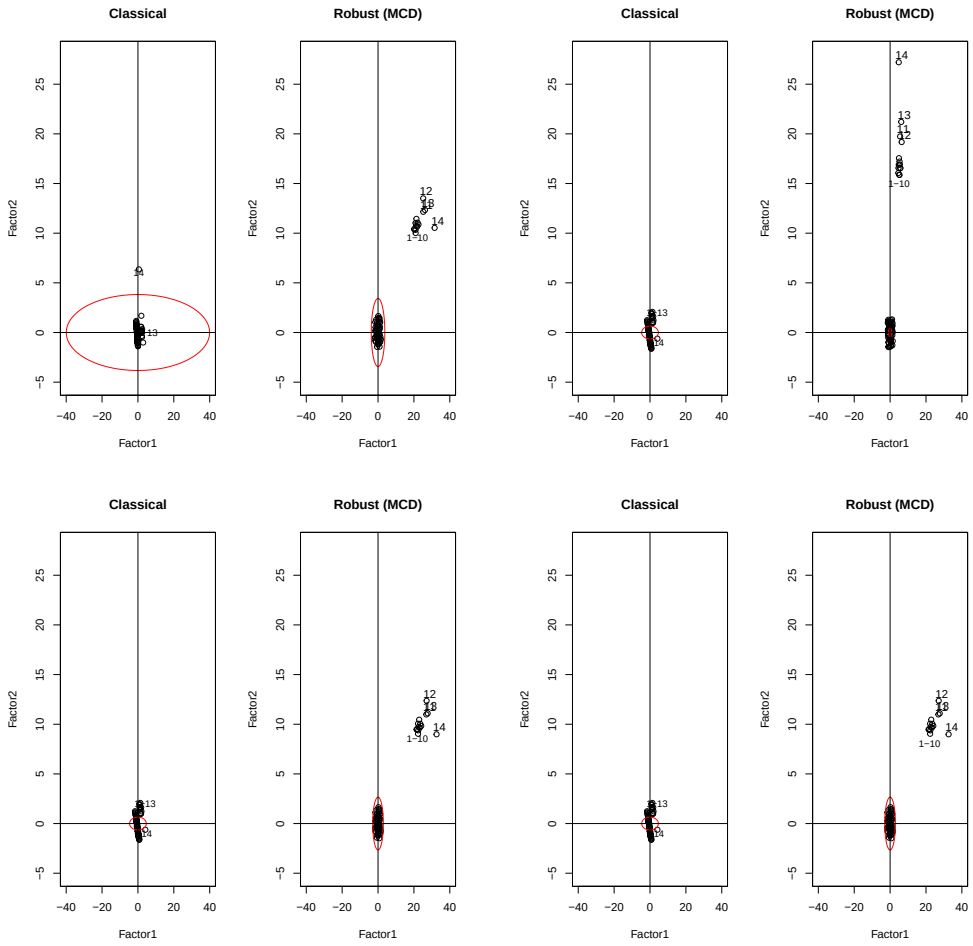


Figure 12: Classical and robust (MCD) scatterplots of the first two factor scores of the hbk data with 97.5% tolerance ellipses. First row: (1) vs (5); (3) vs (7). Second row: (2) vs (6); (4) vs (8).

combinations (7) and (8). Although (6) and (8) have the same running matrices ($\boldsymbol{R}$), eigenvalues (λ), loadings ($\boldsymbol{L}$), uniquenesses ($\boldsymbol{\Psi}$), scoring coefficients ($\boldsymbol{S}_c$), scaled data matrices (scaledX), and score matrices ($\boldsymbol{F}$), as were proved in Theorem 3.2, we may not be able to get results for (6) due to computational error for `cov_x`, while for (8) the computational error does not occur. That is why we recommend (4) vs (8) for classical and robust factor analysis.

The first two eigenvalues of the running matrices of the `stock611` data of the 8 combinations are given in Table 7. From Table 7 we see that the eigenvalues of (2), (3), and (4) are the same, the eigenvalues of (6) and (8) are the same. The results also illustrate Theorems 3.1 and 3.2.

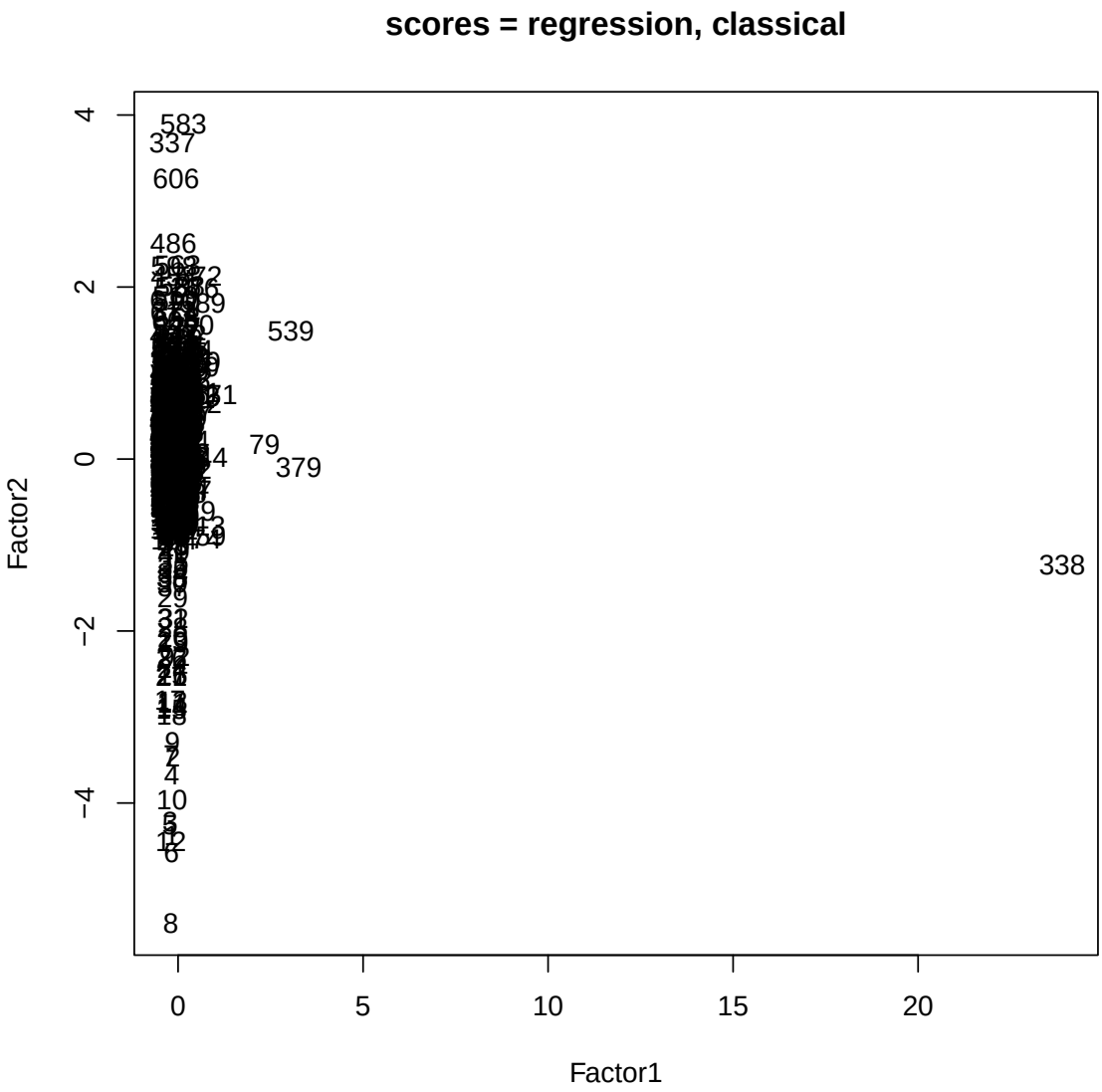
Table 7: The first two eigenvalues of the running matrices of the stock611 data.

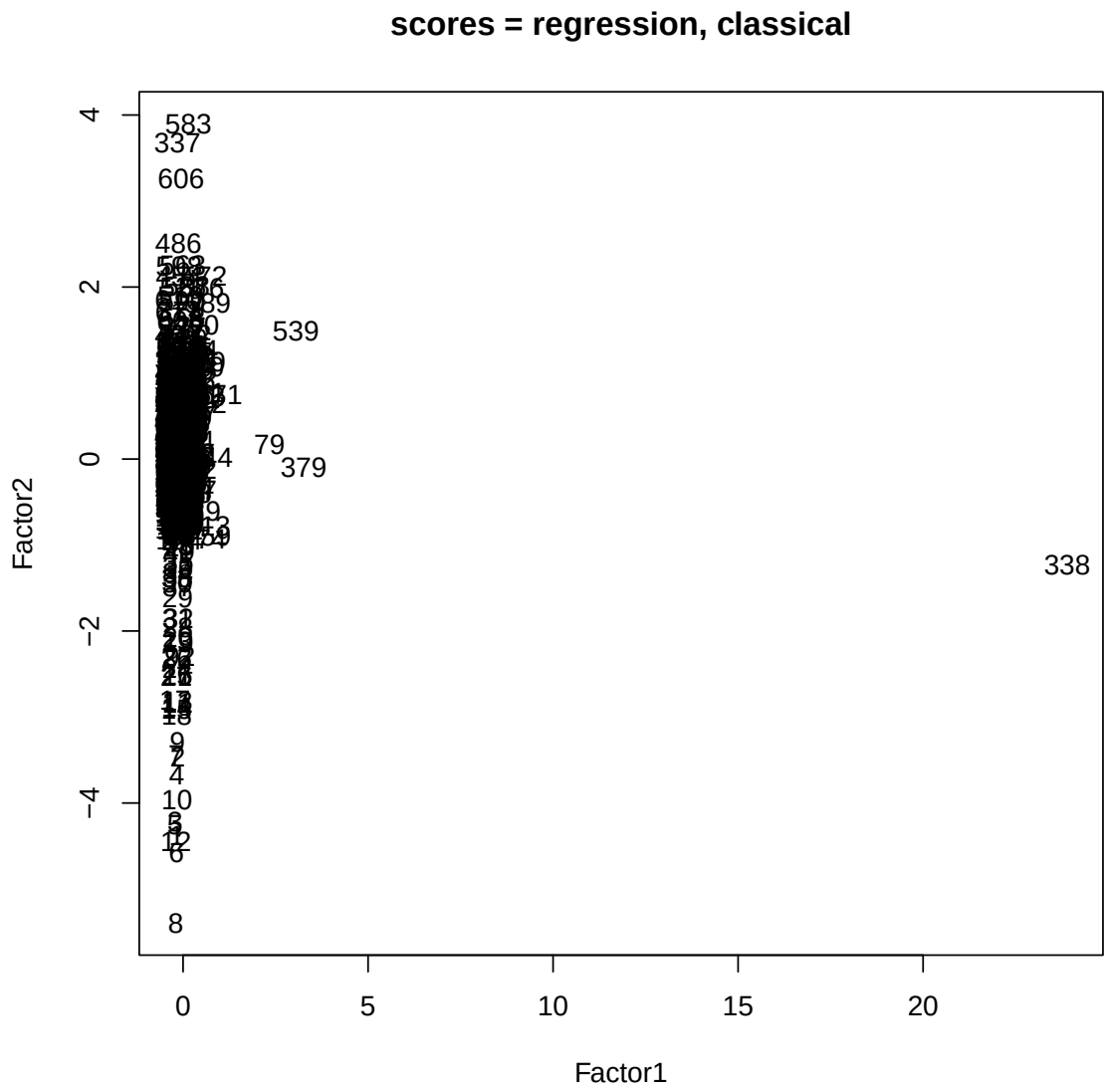
		Classical	Robust (MCD)
stock611[,3:12])	covariance	(1) 4.272384e+20 1.985165e+19	(5) 3.990230e+17 7.358056e+16
	correlation	(2) 5.7900498488 2.3189552681	(6) 5.15840672 2.40502329
scale(stock611[,3:12]))	covariance	(3) 5.7900498488 2.3189552681	(7) 7.527778e-01 2.967432e-01
	correlation	(4) 5.7900498488 2.3189552681	(8) 5.15840672 2.40502329

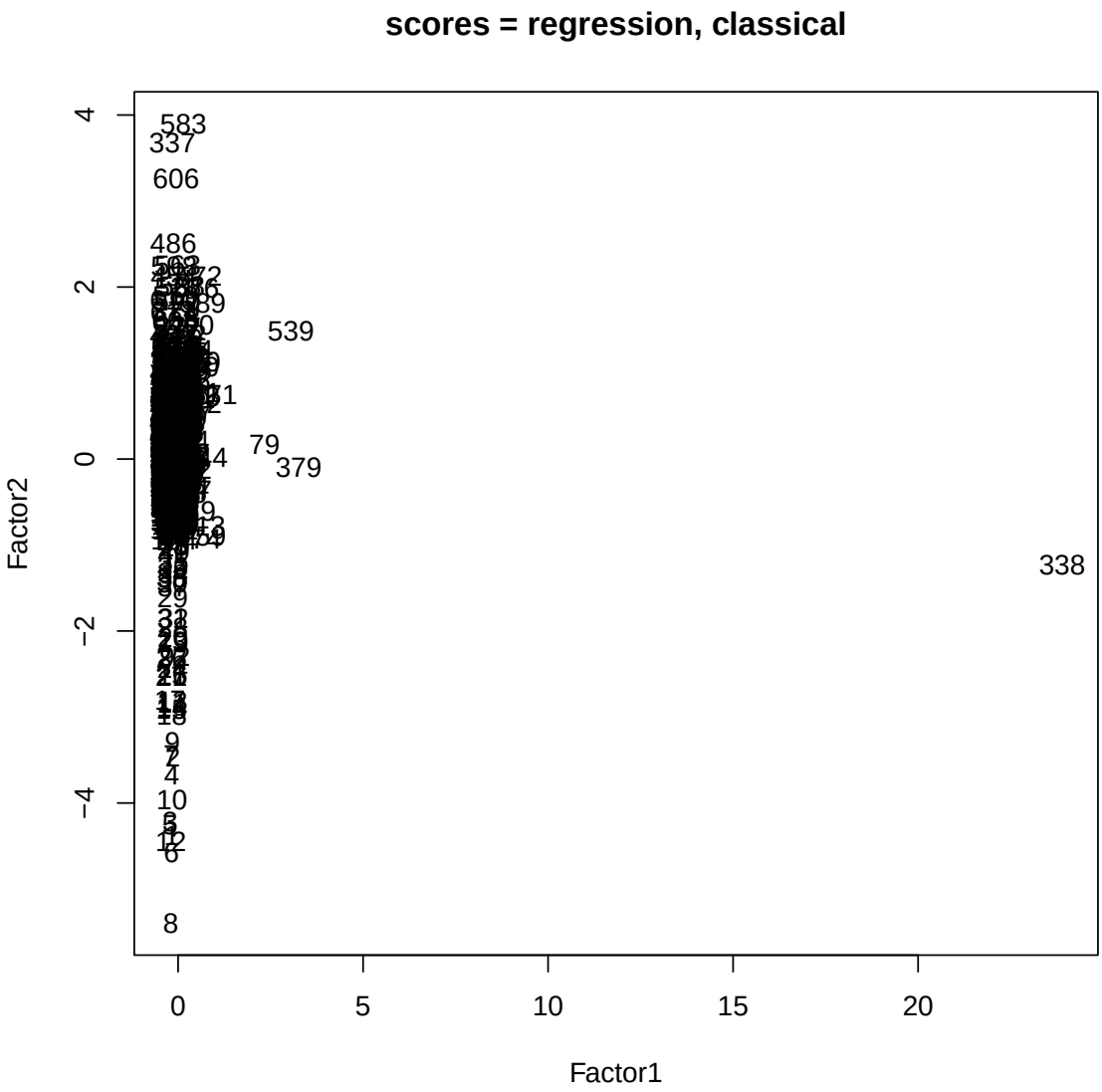
Classical and robust (OGK) scatterplots of the first two factor scores of the `stock611` data with 97.5% tolerance ellipses are ploted in Figure 13. The scatterplots of the first two factor scores of combinations (1) and (5) are not shown, because errors occur in

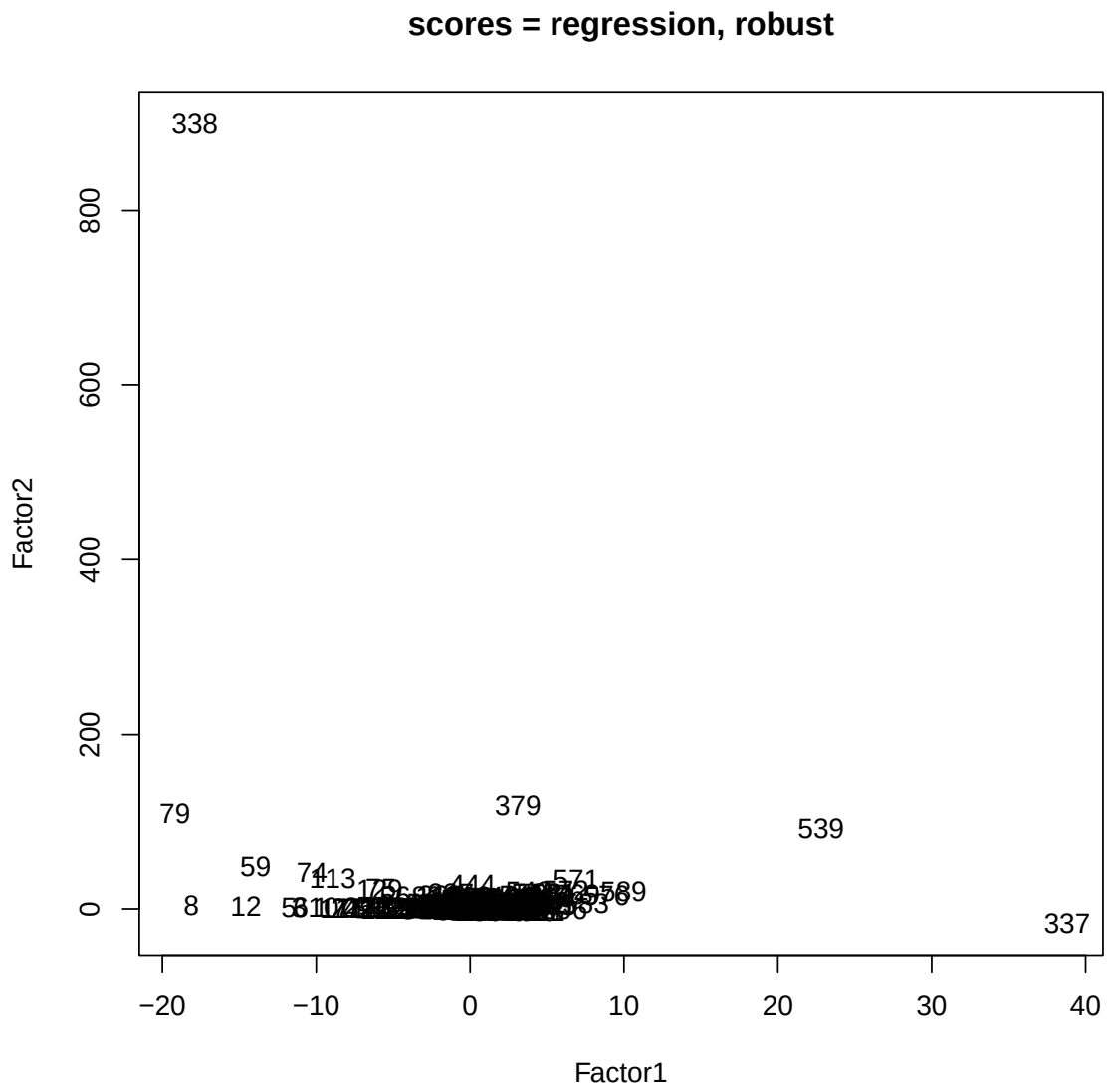
```
solve.default(S): system is computationally singular.
```

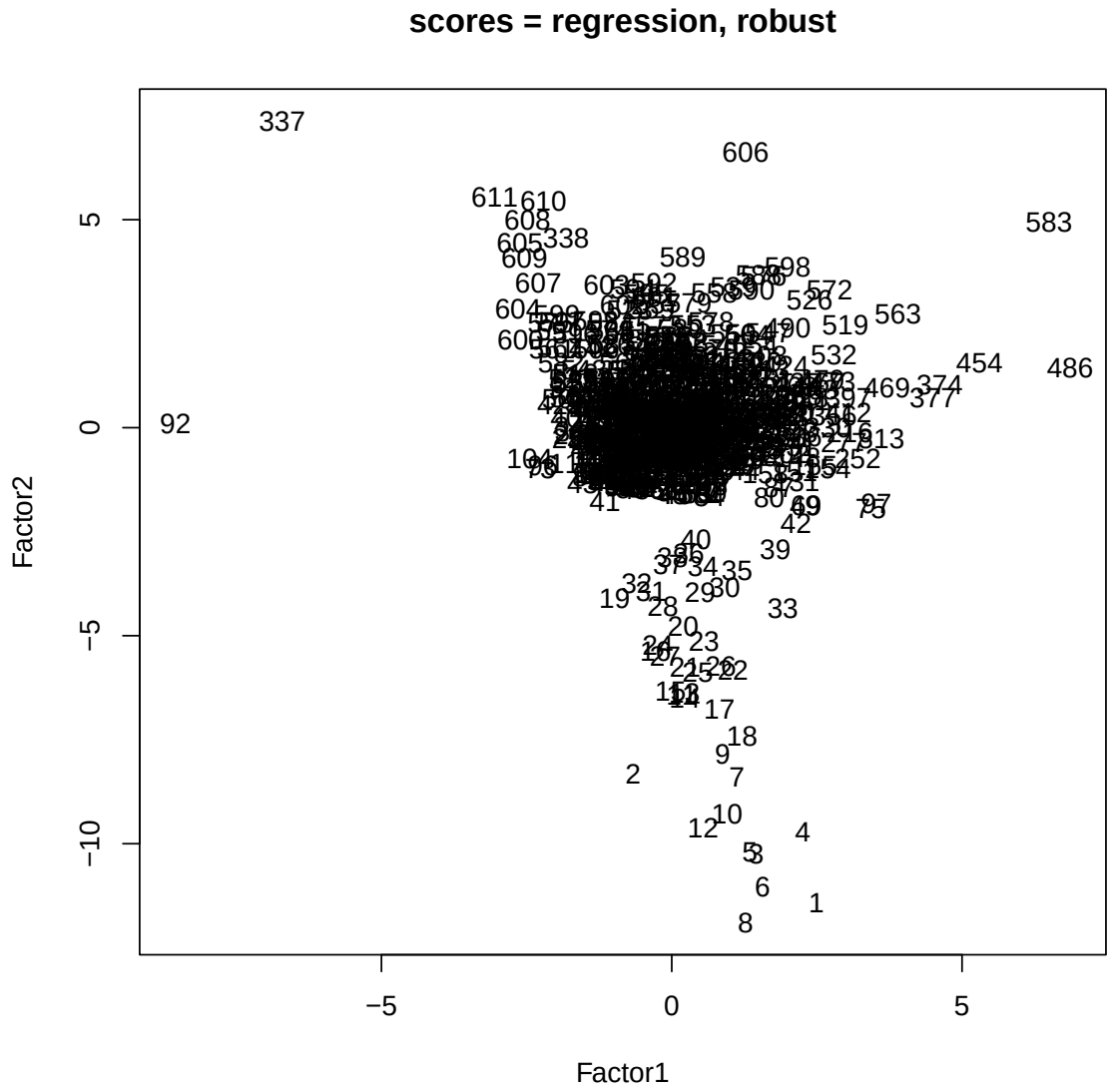
To get a clearer view of the scatterplots, we zoom in the scatterplots. We see that the scores of (2), (3), and (4) are the same, the scores of (6) and (8) are the same, in agree with Theorem 3.2. Note that the tolerance ellipses for (2), (3), and (4) cover the outliers, due to the outliers severely affected the eigenvalues of the running matrices $\boldsymbol{R}^c = \tilde{\boldsymbol{S}}^c = \tilde{\boldsymbol{R}}^c$. The tolerance ellipses of (6) and (8) well seperated the regular points and the outliers.

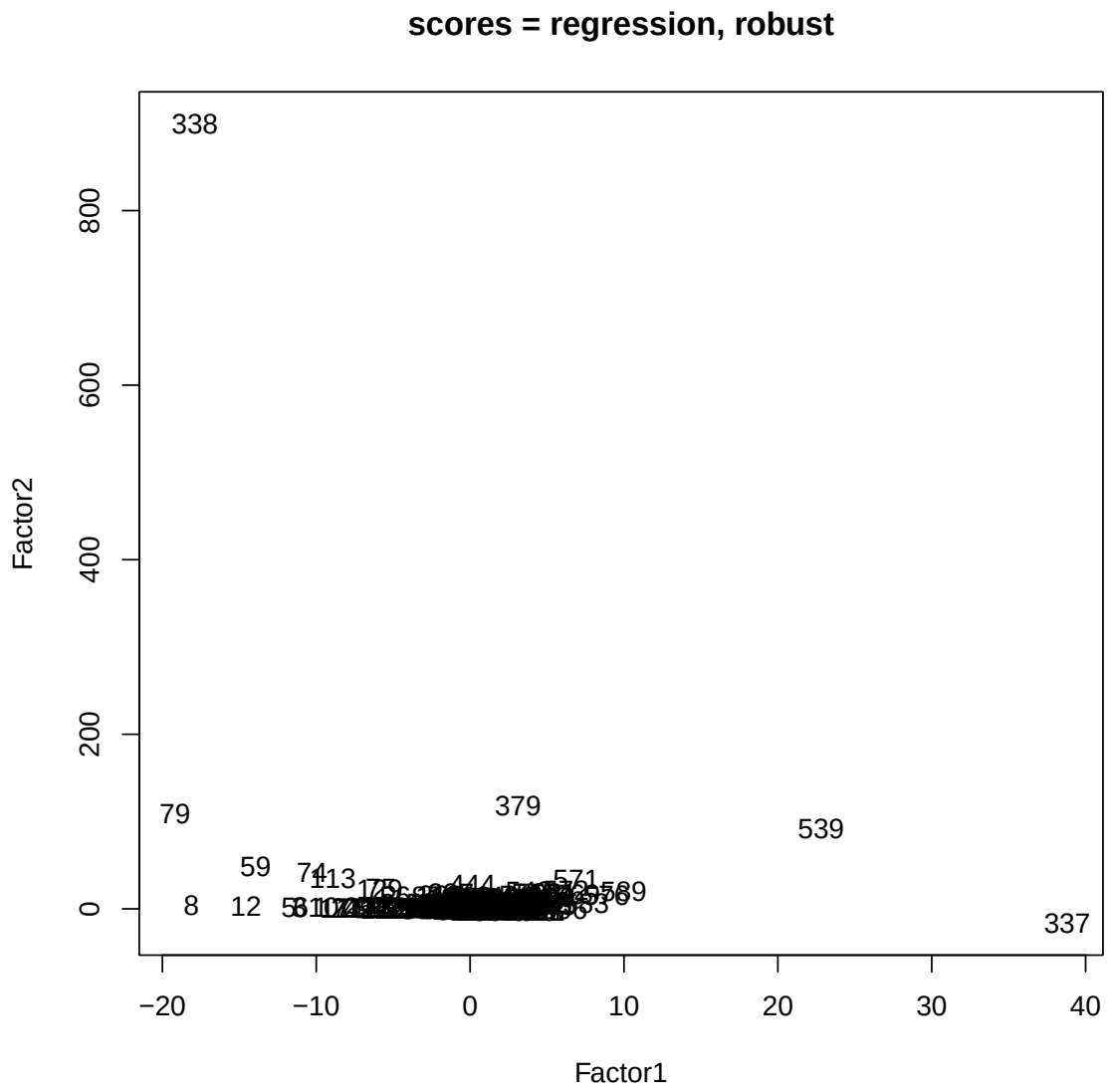

scores = regression, classical
Factor2
Factor1
4
2
0
−2
−4
0
5
10
15
20
583
337
606
486
539
79
379
338
10
8

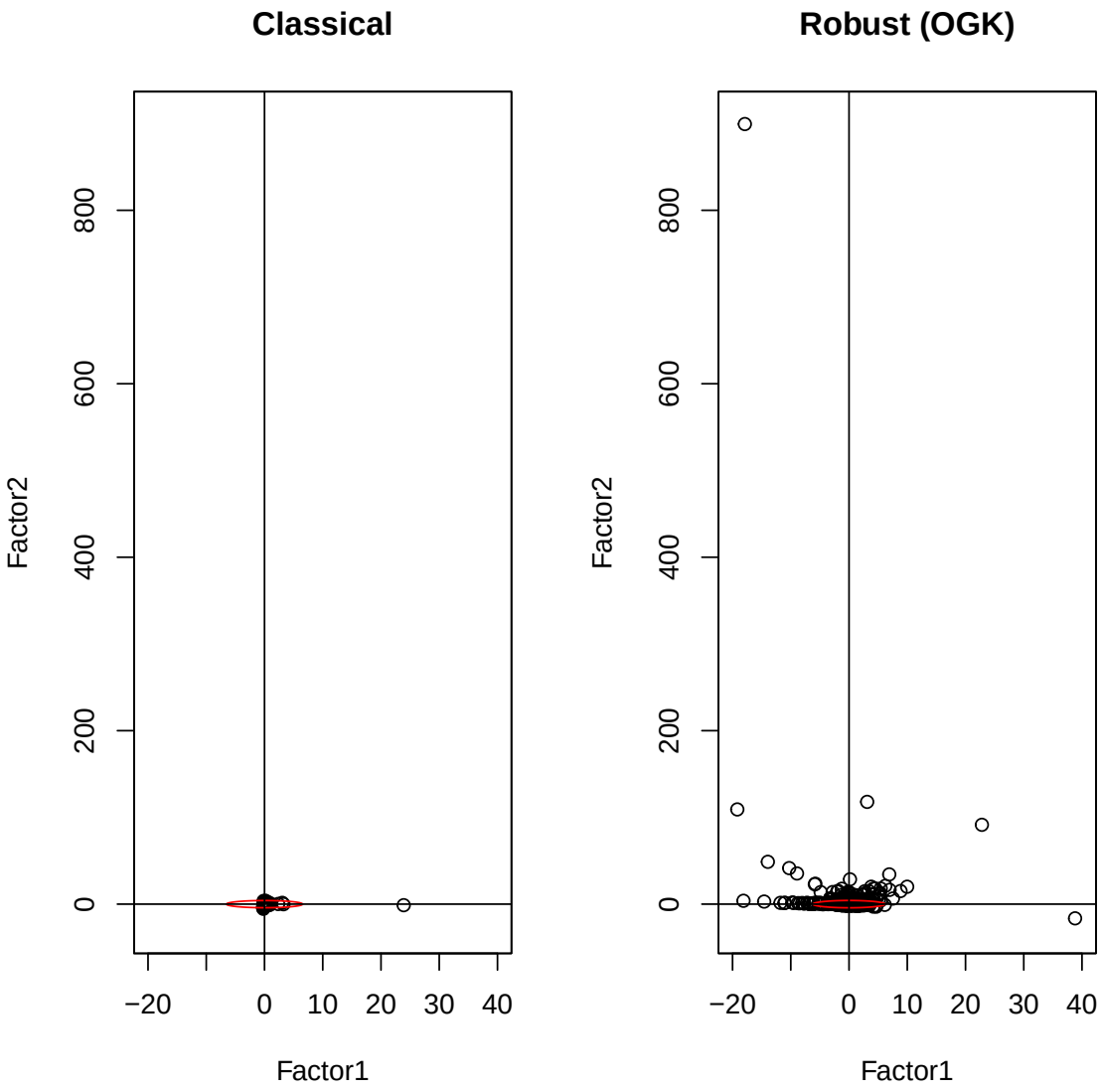

scores = regression, classical
Factor2
Factor1
4
2
0
−2
−4
0
5
10
15
20
583
337
606
486
539
79
379
338
10
8

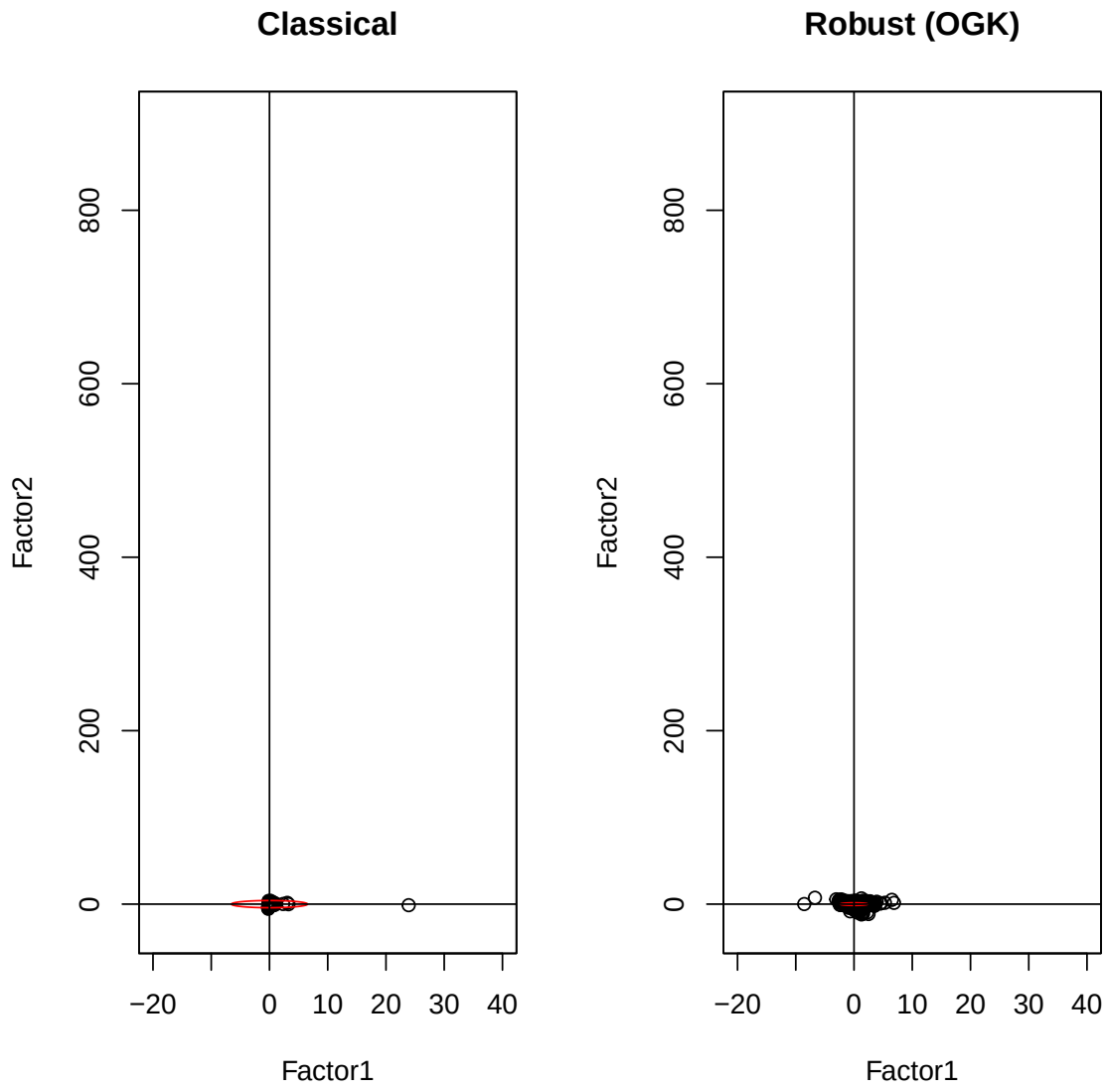

scores = regression, classical
Factor2
Factor1
4
2
0
−2
−4
0
5
10
15
20
583
337
606
486
539
79
379
338
10
8

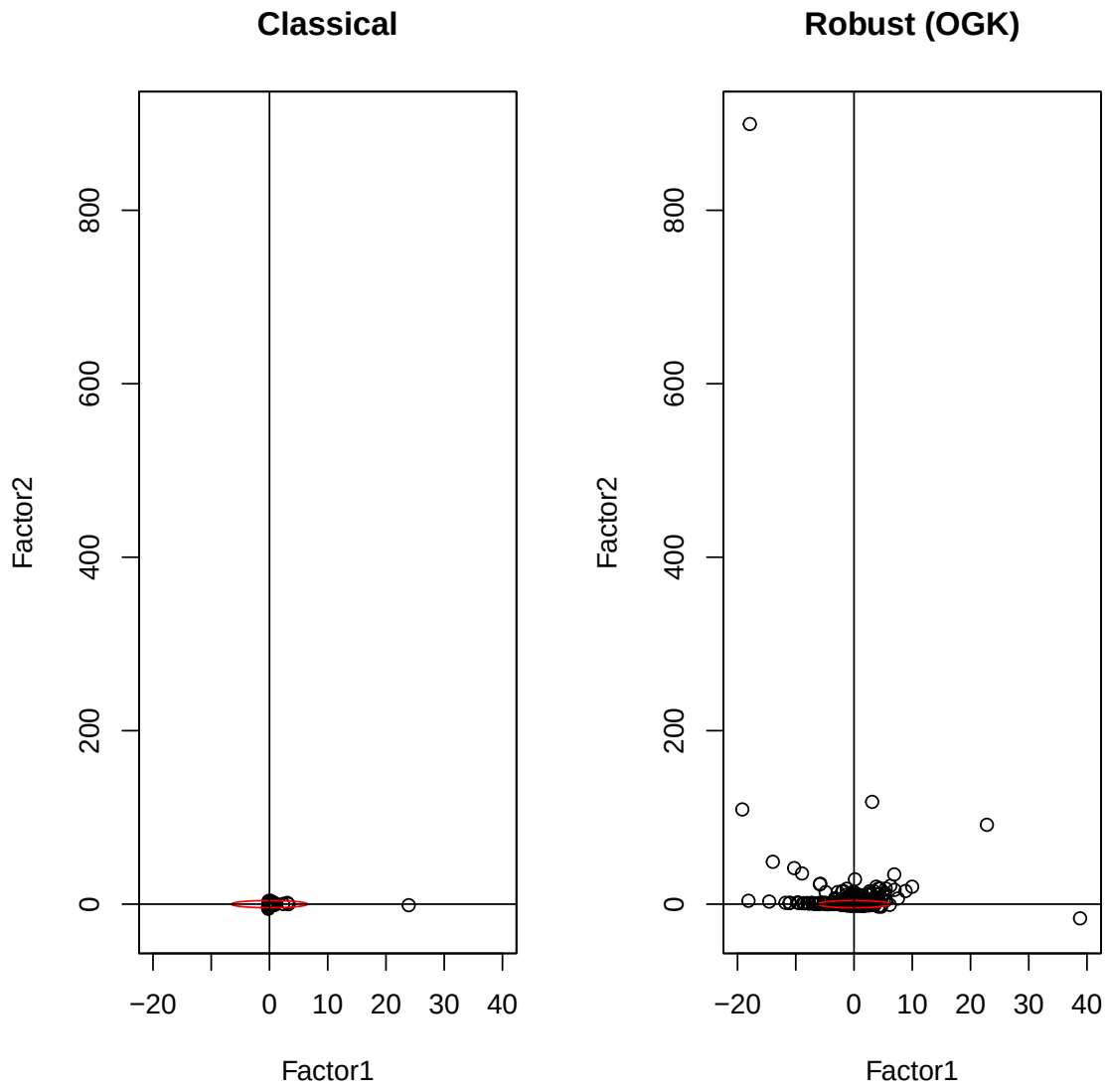

scores = regression, robust
Factor2
Factor1
338
79
379
539
59
8
12
337
0
200
400
600
800
−20
−10
0
10
20
30
40


scores = regression, robust
Factor1
Factor2
−5
0
5
−10


scores = regression, robust
338
800
600
400
200
0
Factor2
79
379
539
59
74
113
571
8
12
337
−20
−10
0
10
20
30
40
Factor1


Classical
Robust (OGK)
Factor2
Factor1
0
200
400
600
800
−20
0
10
20
30
40


Classical
Robust (OGK)
Factor2
Factor1
0
200
400
600
800
−20
0
10
20
30
40



Next we plot a `Cov-class` for the `stock611` data using the function `myplotDD()`. See Figure 14. The figure shows a distance-distance plot. We see that the robust (mahalanobis) distances are far larger than the (classical) mahalanobis distances. The outliers have large robust distances.

```
## cutoff = 4.525834
## id.n <- length(which(rd>cutoff))
## id.n = 287
## Here y is the robust distance (rd).
## sort.y = (To save space, only the smallest five and largest five
## elements of sort.y$x and sort.y$ix are shown.)
## $x
##          281          368          239          312          229
```

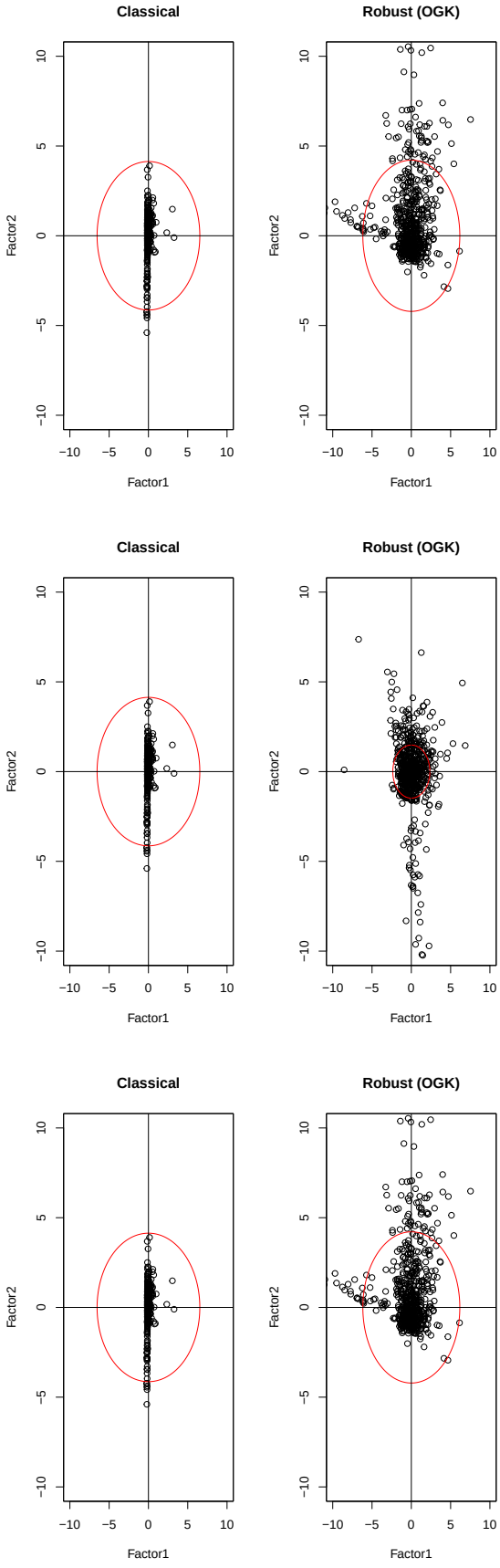


Figure 13: Classical and robust (OGK) scatterplots of the first two factor scores of the stock611 data with 97.5% tolerance ellipses. First row: (2) vs (6). Second row: (3) vs (7). Third row: (4) vs (8).

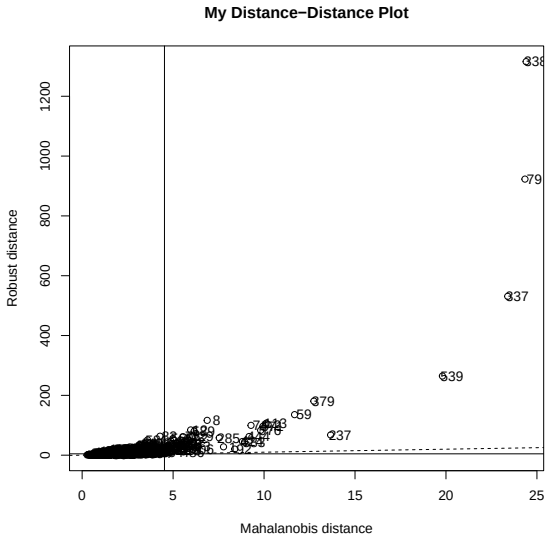


Figure 14: A distance-distance plot for stock611 data.

```
##    0.6880160    0.8109610    0.9234880    0.9763902    1.0429306
##          379          539          337           79          338
##  180.7741644  264.6799442  531.0887434  923.0346401 1315.6544592
##
## $ix
##  [1] 281 368 239 312 229 379 539 337  79 338
## ind =
##   [1] 601 577  36 202 429 387 466 290 570  46 203 566  29 559 313 397
##  [17] 582 122 597 517 192 596 489 261 417 133  30 266 105 549 585 462
##  [33] 493 565 591 547 469 513 289 420 222 502 382 163 194 406 252  41
##  [49]  69 532  73  34 602 435 372 150 481 276 148  38 278 139 401 350
##  [65] 340  70 273 132  72 374 314 366  96 586 394 169  81  49 506 407
##  [81] 184 173 603 524 391  39 141 356 561 599 102 357 538 418 568  37
##  [97] 220 124 187 293  84 578 590 437 459 144  33 518 523 225 156 198
## [113] 419 531  83 600 322 254 519 454 384 495 260  22 554  35  63  23
## [129] 351 609 354 399 448 449 213 174 478 557 188 607 166 442 503 190
## [145] 558 574 104 371 343 365 598 145 176  31 467 486  40 463 287 103
## [161] 567 253 346  61 608 563 117 140 552 409 182 378  26 579 575  28
## [177] 332  97 415 610  53 546 320 108 479 295 341 548  21  32 584  94
## [193]  20 272 471  42 611 544 160  18  87 377 526 288 339 321 138  44
## [209] 606 226  25 153  66  24  14 592 595  92  13 605   7 344  57  16
## [225] 175 309   9  11  19 193   4 413 345 303 504 206 604 216 472 553
## [241]  17  15   1 230 580   3 307  27  10   6 114 562 594   5  56 588
## [257] 541 492 490 540 583 542 424 593 581 569 572 285  75 444 129  82
## [273] 237 589 576  12 571  74   2 113   8  59 379 539 337  79 338
```

From the above results we see that the `cutoff` is computed as `4.525834`. There are `id.n =`

287 observations with robust distances larger than `cutoff`. `sort.y` is a list containing the sorted values of `y` (the robust distance). `sort.y$x` is arranged in increasing order. To save space, only the smallest five and largest five robust distances with their indices are shown. `sort.y$ix` contains the indices. `ind` shows the indices of the largest `id.n = 287` observations whose robust distances are larger than `cutoff`.

From the above results, we recommend (4) vs (8) when one needs to compare classical and robust factor analysis. The following code lines generate an object `faClassic4` of class `FaClassic`.

```
## (4) classical, x = scale(stock611[,3:12]), cor = TRUE (correlation matrix)
faClassic4 = FaClassic(x = scale(stock611[,3:12]), factors = 2, cor = TRUE,
                       method = "pca", scoresMethod = "regression"); str(faClassic4)

## Formal class 'FaClassic' [package "robustfa"] with 25 slots
##   ..@ call              : language FaClassic(x = scale(stock611[, 3:12]), factors = 2,
##   ..@ converged         : NULL
##   ..@ loadings          : num [1:10, 1:2] 0.9878 0.9915 0.9922 0.9849 0.0547 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "X1" "X2" "X3" "X4" ...
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   ..@ communality       : Named num [1:10] 0.976 0.983 0.987 0.976 0.916 ...
##   .. ..- attr(*, "names")= chr [1:10] "X1" "X2" "X3" "X4" ...
##   ..@ uniquenesses      : Named num [1:10] 0.0235 0.0168 0.0126 0.0239 0.0836 ...
##   .. ..- attr(*, "names")= chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ cor               : logi TRUE
##   ..@ covariance        : num [1:10, 1:10] 1 0.9932 0.9703 0.9561 0.0252 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ correlation       : num [1:10, 1:10] 1 0.9932 0.9703 0.9561 0.0252 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ usedMatrix        : num [1:10, 1:10] 1 0.9932 0.9703 0.9561 0.0252 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ reducedCorrelation: NULL
##   ..@ criteria          : NULL
##   ..@ factors           : num 2
##   ..@ dof               : NULL
##   ..@ method            : chr "pca"
##   ..@ scores            : num [1:611, 1:2] -0.17 -0.146 -0.211 -0.171 -0.219 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:611] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
```

```
##   ..@ scoresMethod      : chr "regression"
##   ..@ scoringCoef       : num [1:2, 1:10] 0.1712 -0.0209 0.1717 -0.0124 0.1712 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ meanF             : Named num [1:2] 1.22e-18 4.94e-18
##   .. ..- attr(*, "names")= chr [1:2] "Factor1" "Factor2"
##   ..@ corF              : num [1:2, 1:2] 1.00 -6.21e-15 -6.21e-15 1.00
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   ..@ STATISTIC         : NULL
##   ..@ PVAL              : NULL
##   ..@ n.obs             : int 611
##   ..@ center            : Named num [1:10] 6.30e-18 -8.75e-18 1.27e-17 -1.48e-18 3.23e-
##   .. ..- attr(*, "names")= chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ eigenvalues       : num [1:10] 5.79 2.319 1.009 0.574 0.143 ...
##   ..@ cov.control       : NULL

summary(faClassic4)

## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaClassic"
## Slot "call":
## FaClassic(x = scale(stock611[, 3:12]), factors = 2, cor = TRUE,
##     method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##         Factor1     Factor2
## X1   0.98781949 -0.02611587
## X2   0.99154548 -0.00633362
## X3   0.99222167  0.05353998
## X4   0.98486045  0.07865320
## X5   0.05473434  0.95570764
## X6  -0.01321817  0.62527581
## X7   0.01924642  0.48022545
## X8   0.04578987  0.88138600
## X9   0.94053884 -0.01792872
## X10  0.99068004 -0.04474676
##
## Slot "communality":
##        X1        X2        X3        X4        X5        X6        X7
```

```
## 0.9764694 0.9832026 0.9873704 0.9761364 0.9163729 0.3911446 0.2309869
##        X8        X9       X10
## 0.7789380 0.8849347 0.9834492
##
## Slot "uniquenesses":
##         x1         x2         x3         x4         x5         x6
## 0.02353062 0.01679744 0.01262963 0.02386358 0.08362706 0.60885544
##         x7         x8         x9        x10
## 0.76901309 0.22106200 0.11506525 0.01655078
##
## Slot "cor":
## [1] TRUE
##
## Slot "covariance":
##               x1           x2         x3         x4          x5
## x1   1.000000000  0.993167953 0.97034436 0.95609585 0.025163276
## x2   0.993167953  1.000000000 0.98346739 0.97164822 0.045110192
## x3   0.970344358  0.983467387 1.00000000 0.99749137 0.108682670
## x4   0.956095848  0.971648219 0.99749137 1.00000000 0.133591430
## x5   0.025163276  0.045110192 0.10868267 0.13359143 1.000000000
## x6  -0.015083014 -0.004994507 0.01946436 0.02927864 0.575741829
## x7   0.009780105  0.014165548 0.03527696 0.04429804 0.326885328
## x8   0.017155253  0.036225902 0.09193857 0.11691517 0.840607798
## x9   0.905256744  0.898571356 0.91651307 0.91138685 0.041581017
## x10  0.989449715  0.989605604 0.97531937 0.96400047 0.001814838
##               x6           x7         x8           x9          x10
## x1  -0.015083014 0.009780105 0.01715525  0.905256744  0.989449715
## x2  -0.004994507 0.014165548 0.03622590  0.898571356  0.989605604
## x3   0.019464360 0.035276960 0.09193857  0.916513072  0.975319373
## x4   0.029278642 0.044298039 0.11691517  0.911386851  0.964000474
## x5   0.575741829 0.326885328 0.84060780  0.041581017  0.001814838
## x6   1.000000000 0.013993207 0.29848940 -0.009413168 -0.046691411
## x7   0.013993207 1.000000000 0.35473163  0.012122296  0.007367798
## x8   0.298489402 0.354731631 1.00000000  0.010355218  0.016828080
## x9  -0.009413168 0.012122296 0.01035522  1.000000000  0.911435130
## x10 -0.046691411 0.007367798 0.01682808  0.911435130  1.000000000
##
## Slot "correlation":
##               x1           x2         x3         x4          x5
## x1   1.000000000  0.993167953 0.97034436 0.95609585 0.025163276
## x2   0.993167953  1.000000000 0.98346739 0.97164822 0.045110192
## x3   0.970344358  0.983467387 1.00000000 0.99749137 0.108682670
## x4   0.956095848  0.971648219 0.99749137 1.00000000 0.133591430
## x5   0.025163276  0.045110192 0.10868267 0.13359143 1.000000000
## x6  -0.015083014 -0.004994507 0.01946436 0.02927864 0.575741829
## x7   0.009780105  0.014165548 0.03527696 0.04429804 0.326885328
```

```
## x8   0.017155253  0.036225902 0.09193857 0.11691517 0.840607798
## x9   0.905256744  0.898571356 0.91651307 0.91138685 0.041581017
## x10  0.989449715  0.989605604 0.97531937 0.96400047 0.001814838
##               x6          x7         x8           x9          x10
## x1  -0.015083014 0.009780105 0.01715525  0.905256744  0.989449715
## x2  -0.004994507 0.014165548 0.03622590  0.898571356  0.989605604
## x3   0.019464360 0.035276960 0.09193857  0.916513072  0.975319373
## x4   0.029278642 0.044298039 0.11691517  0.911386851  0.964000474
## x5   0.575741829 0.326885328 0.84060780  0.041581017  0.001814838
## x6   1.000000000 0.013993207 0.29848940 -0.009413168 -0.046691411
## x7   0.013993207 1.000000000 0.35473163  0.012122296  0.007367798
## x8   0.298489402 0.354731631 1.00000000  0.010355218  0.016828080
## x9  -0.009413168 0.012122296 0.01035522  1.000000000  0.911435130
## x10 -0.046691411 0.007367798 0.01682808  0.911435130  1.000000000
##
## Slot "usedMatrix":
##               x1           x2         x3         x4          x5
## x1   1.000000000  0.993167953 0.97034436 0.95609585 0.025163276
## x2   0.993167953  1.000000000 0.98346739 0.97164822 0.045110192
## x3   0.970344358  0.983467387 1.00000000 0.99749137 0.108682670
## x4   0.956095848  0.971648219 0.99749137 1.00000000 0.133591430
## x5   0.025163276  0.045110192 0.10868267 0.13359143 1.000000000
## x6  -0.015083014 -0.004994507 0.01946436 0.02927864 0.575741829
## x7   0.009780105  0.014165548 0.03527696 0.04429804 0.326885328
## x8   0.017155253  0.036225902 0.09193857 0.11691517 0.840607798
## x9   0.905256744  0.898571356 0.91651307 0.91138685 0.041581017
## x10  0.989449715  0.989605604 0.97531937 0.96400047 0.001814838
##               x6          x7         x8           x9          x10
## x1  -0.015083014 0.009780105 0.01715525  0.905256744  0.989449715
## x2  -0.004994507 0.014165548 0.03622590  0.898571356  0.989605604
## x3   0.019464360 0.035276960 0.09193857  0.916513072  0.975319373
## x4   0.029278642 0.044298039 0.11691517  0.911386851  0.964000474
## x5   0.575741829 0.326885328 0.84060780  0.041581017  0.001814838
## x6   1.000000000 0.013993207 0.29848940 -0.009413168 -0.046691411
## x7   0.013993207 1.000000000 0.35473163  0.012122296  0.007367798
## x8   0.298489402 0.354731631 1.00000000  0.010355218  0.016828080
## x9  -0.009413168 0.012122296 0.01035522  1.000000000  0.911435130
## x10 -0.046691411 0.007367798 0.01682808  0.911435130  1.000000000
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
##
## Slot "factors":
```

```
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##           Factor1      Factor2
## 1   -1.699858e-01 -4.375026057
## 2   -1.456487e-01 -3.464717456
## 3   -2.113478e-01 -4.220859598
## 4   -1.706501e-01 -3.671691917
## 5   -2.194292e-01 -4.252586052
## 6   -1.783264e-01 -4.579283183
## 7   -1.870716e-01 -3.438303504
## 8   -1.982348e-01 -5.396957302
## 9   -1.533491e-01 -3.285587393
## 10  -1.638305e-01 -3.965385663
## 11  -1.513386e-01 -2.838850620
## 12  -1.938650e-01 -4.444923075
## 13  -1.510262e-01 -2.818374959
## 14  -1.626230e-01 -2.878305221
## 15  -1.711379e-01 -2.903302047
## 16  -1.573203e-01 -2.514850618
## 17  -2.192700e-01 -2.796501002
## 18  -1.698394e-01 -2.972721628
## 19  -1.311105e-01 -2.129288001
## 20  -1.422608e-01 -2.072237199
## 21  -1.637352e-01 -2.528128793
## 22  -1.588979e-01 -2.289897064
## 23  -1.561937e-01 -2.120779866
## 24  -1.481493e-01 -2.434010707
## 25  -1.711949e-01 -2.516587483
## 26  -1.739110e-01 -2.320519125
## 27  -1.903913e-01 -2.525406720
## 28  -1.465570e-01 -1.964111949
## 29  -1.476963e-01 -1.610528915
## 30  -1.550552e-01 -1.440698411
## 31  -1.446821e-01 -1.858083818
## 32  -1.251107e-01 -1.852237680
## 33  -1.617153e-01 -1.380173795
## 34  -1.433893e-01 -1.319327370
## 35  -1.298339e-01 -1.193449788
## 36  -1.389689e-01 -1.247948631
```

```
## 37  -1.391849e-01 -1.480990964
## 38  -1.420172e-01 -1.377621546
## 39  -1.567545e-01 -0.789628714
## 40  -1.024860e-01 -1.070246968
## 41  -1.335405e-01 -1.096268165
## 42  -2.874764e-02 -0.427021633
## 43  -1.357416e-01 -1.001899520
## 44   1.392604e-02 -0.714606352
## 45  -1.219326e-01 -0.692824672
## 46  -1.055244e-01 -0.678526470
## 47  -9.987343e-02 -0.820545711
## 48  -1.224491e-01 -0.900275824
## 49  -1.361519e-01 -0.189980667
## 50  -1.328078e-01 -0.822348909
## 51  -9.684450e-02 -0.853609288
## 52  -1.325418e-01 -0.857943421
## 53  -3.849310e-02 -0.577288346
## 54  -1.111086e-01 -0.542459613
## 55  -1.310137e-01 -0.749999962
## 56   1.652689e-01 -0.704407564
## 57   1.322257e-02 -0.897515439
## 58  -1.332818e-01 -0.581664143
## 59   8.830250e-01 -0.893910449
## 60  -1.187836e-01 -0.816838607
## 61  -8.237417e-02 -0.777048140
## 62  -1.048141e-01 -0.782247388
## 63  -6.153890e-02 -0.879277174
## 64  -1.349657e-01 -0.758826827
## 65  -9.320033e-02 -0.678012481
## 66  -3.980957e-02 -0.483644188
## 67  -1.374188e-01 -0.810850588
## 68  -1.377868e-01 -0.783782803
## 69  -1.493788e-01 -0.173329228
## 70  -9.754972e-02 -0.521734620
## 71  -1.160058e-01 -0.872340379
## 72  -9.466914e-02 -0.530227689
## 73  -1.267002e-01 -1.011264500
## 74   7.621726e-01 -0.925302803
## 75   3.605263e-01  0.028563544
## 76  -1.405925e-01 -0.851206934
## 77  -1.466037e-01 -0.514991126
## 78  -1.437034e-01 -0.827804637
## 79   2.334763e+00  0.173082392
## 80  -1.434937e-01 -0.266486958
## 81  -1.103633e-01 -0.522059846
## 82   1.911511e-01 -0.299968489
```

```
## 83  -1.046164e-01 -0.759543530
## 84  -6.961500e-02 -0.591615479
## 85  -1.380815e-01 -0.898751337
## 86  -1.201476e-01 -0.659329525
## 87  -6.175348e-03 -0.137818467
## 88  -1.239974e-01 -0.667500205
## 89  -1.317718e-01 -0.841743911
## 90  -1.345571e-01 -0.851447469
## 91  -1.245961e-01 -0.538274193
## 92  -8.409746e-02 -2.297051680
## 93  -9.619934e-02 -0.411968937
## 94   8.974033e-03 -0.644413571
## 95  -1.343479e-01 -0.703944111
## 96  -8.765212e-02 -0.980075108
## 97  -6.528285e-02  0.162251789
## 98  -1.314665e-01 -0.802726399
## 99  -1.409833e-01 -0.814043379
## 100 -8.414339e-02 -0.669925934
## 101 -1.066330e-01 -0.628069327
## 102 -7.874074e-02 -0.826095262
## 103 -8.473624e-02 -0.736376102
## 104 -1.060630e-01 -0.940828489
## 105 -7.707381e-02 -0.767840794
## 106 -1.428479e-01 -0.398502655
## 107 -8.228696e-02 -0.534000217
## 108 -3.538270e-02 -0.378480206
## 109 -1.254709e-01 -0.433284423
## 110 -1.060867e-01 -0.454042145
## 111 -9.381098e-02 -0.527454260
## 112 -1.324127e-01 -0.618897986
## 113  6.627497e-01 -0.783967006
## 114  2.201612e-01 -0.332531797
## 115 -1.320614e-01 -0.819287579
## 116 -1.308060e-01 -0.333878164
## 117 -5.172589e-02  0.220726572
## 118 -1.334037e-01 -0.434141631
## 119 -1.307498e-01 -0.726007127
## 120 -1.184207e-01 -0.377060991
## 121 -1.157299e-01 -0.562088963
## 122 -9.571889e-02 -0.488239852
## 123 -1.271553e-01 -0.472600422
## 124 -6.096781e-02 -0.144714397
## 125 -1.188934e-01 -0.484860878
## 126 -1.256559e-01 -0.388494768
## 127 -1.266494e-01 -0.628045724
## 128 -1.390007e-01 -0.710766525
```

```
## 129  3.956470e-01 -0.602174340
## 130 -1.321062e-01 -0.250412619
## 131 -9.587986e-02  0.022632364
## 132 -8.660562e-02 -0.143240688
## 133 -1.058073e-01 -0.312493641
## 134 -1.001683e-01 -0.494323317
## 135 -1.268752e-01 -0.680419286
## 136 -6.341631e-02 -0.568312479
## 137 -1.178805e-01 -0.193599553
## 138  5.702214e-02 -0.143731581
## 139 -6.274739e-02 -0.214840347
## 140 -3.916633e-02 -0.667046726
## 141 -3.511885e-02 -0.325392219
## 142 -1.138376e-01 -0.374982275
## 143 -7.514682e-02 -0.370672403
## 144 -7.203927e-02 -0.141652376
## 145  3.323594e-02 -0.276794940
## 146 -1.174657e-01 -0.247637190
## 147 -1.320091e-01 -0.289211089
## 148 -7.769902e-02  0.277190338
## 149 -1.119234e-01 -0.098511488
## 150 -8.288749e-02 -0.095664571
## 151 -1.434479e-01  0.124759385
## 152 -1.048938e-01 -0.459966801
## 153  1.497535e-01 -0.399960989
## 154 -1.481914e-01  0.315758060
## 155 -1.322389e-01  0.276790462
## 156 -7.923328e-03 -0.409555215
## 157 -1.228907e-01 -0.466420424
## 158 -1.138028e-01 -0.037757383
## 159 -1.177613e-01 -0.512242218
## 160  6.918949e-02 -0.002834825
## 161 -9.759709e-02 -0.328798984
## 162 -8.397076e-02 -0.247336957
## 163 -5.191669e-02 -0.209837998
## 164 -1.161426e-01 -0.404362391
## 165 -9.922097e-02 -0.282481054
## 166  3.444059e-03 -0.133769044
## 167 -1.264178e-01 -0.167498282
## 168 -1.187392e-01 -0.556546263
## 169 -6.089942e-02  0.031333503
## 170 -1.027806e-01 -0.129430285
## 171 -9.967122e-02 -0.205516756
## 172 -1.190664e-01 -0.005982868
## 173 -8.042214e-02 -0.616474673
## 174  4.098650e-03  0.162125514
```

```
## 175  1.226562e-01 -0.007934430
## 176  4.935833e-03  0.079359167
## 177 -1.244777e-01 -0.342607624
## 178 -1.360348e-01 -0.261240487
## 179 -1.301573e-01 -0.143369826
## 180 -8.750782e-02 -0.306461573
## 181 -1.401064e-01 -0.040864885
## 182  8.850255e-03  0.155499343
## 183 -1.339306e-01 -0.537416194
## 184 -6.009910e-02 -0.145510145
## 185 -9.712726e-02 -0.265014326
## 186 -1.145112e-01 -0.440573382
## 187 -4.752621e-02 -0.278502570
## 188 -2.458950e-02  0.088281386
## 189 -1.163822e-01 -0.253973577
## 190 -2.411332e-02 -0.122304284
## 191 -1.127843e-01 -0.444159489
## 192 -4.715574e-02 -0.071329171
## 193 -2.771145e-02 -0.403342738
## 194 -1.152567e-01 -0.190508161
## 195 -1.060693e-01 -0.422435319
## 196 -1.232014e-01 -0.522961411
## 197 -9.776248e-02 -0.231294913
## 198 -7.086980e-02  0.533592690
## 199 -1.168436e-01 -0.122987344
## 200 -9.083638e-02 -0.276916659
## 201 -1.078174e-01 -0.395418560
## 202 -1.012376e-01  0.273873644
## 203 -1.121714e-01  0.075878560
## 204 -1.314606e-01 -0.226283070
## 205 -9.087952e-02 -0.500436179
## 206  8.523304e-02 -0.078298889
## 207 -8.613211e-02 -0.342238470
## 208 -1.075000e-01 -0.183082004
## 209 -1.374349e-01 -0.007780306
## 210 -1.119170e-01  0.034023867
## 211 -1.029835e-01 -0.420792992
## 212 -1.322809e-01 -0.442211937
## 213  1.510835e-02  0.036674709
## 214 -1.106511e-01 -0.376250367
## 215 -1.320929e-01 -0.205395234
## 216  2.383230e-01  0.750089819
## 217 -9.003175e-02 -0.293516863
## 218 -1.277089e-01 -0.048332115
## 219 -1.207849e-01 -0.004745901
## 220 -3.509393e-02  0.237541509
```

```
## 221 -5.254541e-02 -0.166115307
## 222 -2.802711e-02 -0.292393349
## 223 -1.259965e-01 -0.136211596
## 224 -1.133531e-01 -0.160494008
## 225  2.102464e-02  0.127550023
## 226  8.254483e-02 -0.156618098
## 227 -1.182333e-01 -0.519757356
## 228 -7.135189e-02 -0.329597829
## 229 -1.151848e-01 -0.245971499
## 230  1.584133e-01  0.484968305
## 231 -1.352676e-01  0.385819621
## 232 -9.433564e-02  0.044630097
## 233 -6.835187e-02  0.091587614
## 234 -9.335572e-02 -0.283395447
## 235 -1.298918e-01  0.275739216
## 236 -1.078743e-01  0.083672432
## 237  3.233960e-01  0.547807741
## 238 -1.124966e-01 -0.035597330
## 239 -1.242279e-01 -0.196087483
## 240 -6.175974e-02  0.034414437
## 241 -1.272726e-01 -0.559608033
## 242 -1.182251e-01 -0.001394482
## 243 -7.683981e-02 -0.137135560
## 244 -1.326004e-01 -0.208881127
## 245 -9.044687e-02  0.596467003
## 246 -1.030000e-01 -0.158551194
## 247 -1.010226e-01  0.167867734
## 248 -1.364728e-01 -0.023827907
## 249 -1.337033e-01 -0.079332560
## 250 -1.385490e-01 -0.143620031
## 251 -1.346643e-01 -0.537451873
## 252 -1.581246e-01  0.557140224
## 253  1.684041e-02  0.015483121
## 254  4.007350e-02  0.333112487
## 255 -1.198381e-01 -0.160457416
## 256 -1.406797e-01 -0.325916476
## 257 -8.406956e-02 -0.183522375
## 258 -1.447710e-01  0.186625021
## 259 -1.229240e-01  0.298680952
## 260 -1.384406e-02 -0.325057628
## 261 -1.231326e-01  0.172613379
## 262 -1.258817e-01 -0.447980346
## 263 -9.332787e-02  0.048324831
## 264 -1.304774e-01 -0.484829673
## 265 -1.127033e-01 -0.186320570
## 266 -4.493758e-02 -0.213657387
```

```
## 267 -1.136006e-01 -0.009279834
## 268 -1.092382e-01 -0.444653165
## 269 -7.868630e-02 -0.122769841
## 270 -1.209871e-01  0.277108937
## 271 -5.352387e-02  0.294875708
## 272  8.092895e-02  0.467869993
## 273 -9.143511e-02 -0.479723640
## 274 -1.109390e-01 -0.464236831
## 275 -5.810359e-02 -0.072693363
## 276 -2.346998e-02 -0.204407133
## 277 -1.523955e-01  0.663499952
## 278 -2.386839e-02  0.162443061
## 279 -1.266187e-01 -0.420629250
## 280 -1.342638e-01 -0.007965081
## 281 -1.046248e-01 -0.091485636
## 282 -1.053444e-01 -0.273417797
## 283 -2.603839e-02 -0.157210199
## 284 -7.393122e-02 -0.077413358
## 285  2.581735e-01  0.910167214
## 286 -1.114537e-01  0.227038235
## 287  1.419434e-02  0.472808335
## 288  1.490657e-01 -0.078198073
## 289 -7.745228e-02  0.018440139
## 290 -2.067462e-02 -0.022656520
## 291 -1.118736e-01  0.458220003
## 292 -1.270679e-01  0.319487334
## 293  2.530751e-02  0.078210441
## 294 -1.443456e-01  0.081691047
## 295  8.751717e-02  0.862045140
## 296 -1.149315e-01 -0.383446839
## 297 -1.042341e-01 -0.489719598
## 298 -1.219742e-01 -0.225763724
## 299 -8.019848e-02 -0.156259182
## 300 -4.632951e-02 -0.182309737
## 301 -9.895387e-02 -0.393990169
## 302 -3.689056e-02  0.603294629
## 303  2.405751e-01  0.019733317
## 304 -1.107070e-01  0.064491546
## 305 -1.184152e-01 -0.046551853
## 306 -6.700154e-02 -0.025591330
## 307  2.885846e-01 -0.361096530
## 308 -5.437482e-02  0.455704297
## 309  4.026669e-02  0.295260806
## 310 -1.077109e-01  0.171811285
## 311 -6.868359e-02 -0.119573136
## 312 -1.056642e-01 -0.244476853
```

```
## 313 -1.467546e-01  0.882649248
## 314 -6.682986e-02  0.491513662
## 315 -9.831269e-02  0.267907783
## 316 -1.352379e-01  0.010317967
## 317 -1.078081e-01 -0.102567262
## 318 -1.035860e-01 -0.382221282
## 319 -4.687962e-02  0.336700551
## 320  2.978169e-03  0.632299726
## 321  6.501655e-02  0.393155972
## 322 -3.136637e-02 -0.409902202
## 323 -9.618055e-02  0.664335853
## 324 -5.030915e-02  0.129633205
## 325 -8.982273e-02  0.075424007
## 326 -4.958924e-02  0.046404934
## 327 -1.104828e-01  0.050673262
## 328 -6.607229e-02  0.319555346
## 329 -8.603984e-02  0.312706203
## 330 -1.156617e-01  0.745796666
## 331 -6.922546e-02  0.286926369
## 332  7.576977e-02 -0.110092077
## 333 -7.983738e-02  0.406943828
## 334 -1.295600e-01  0.282284100
## 335 -1.253160e-01  0.262450295
## 336 -1.424083e-01  0.515904879
## 337 -1.520371e-01  3.681513499
## 338  2.389674e+01 -1.229303681
## 339  6.055157e-02  0.294078771
## 340 -1.902759e-02  0.150213074
## 341 -1.690959e-02  0.921026105
## 342 -1.094799e-01  0.213239752
## 343  7.864334e-02  0.308501168
## 344  4.931215e-02  1.021202225
## 345  1.641955e-01 -0.019350192
## 346 -6.402503e-02 -0.409692554
## 347 -1.320028e-01  0.117322584
## 348 -1.089423e-01  0.307621062
## 349 -1.193887e-01  0.271909378
## 350 -2.997634e-02  0.509011544
## 351 -8.505199e-02 -0.067866079
## 352 -9.759649e-02 -0.137638798
## 353 -6.547051e-02 -0.091714142
## 354 -8.773088e-02  0.661742200
## 355 -8.844273e-02 -0.251464445
## 356 -4.288610e-02  0.886320555
## 357 -2.689220e-02  0.149741913
## 358 -9.117443e-02  0.321157522
```

```
## 359 -1.398758e-01  0.180876894
## 360 -1.118852e-01 -0.459870859
## 361 -6.680949e-02  0.136767155
## 362 -1.215268e-01  0.375156623
## 363 -1.190795e-01  0.360640825
## 364 -9.043280e-02  0.647984109
## 365  1.646310e-02  0.021276909
## 366 -3.289241e-02  0.637952238
## 367 -1.343001e-01  0.319673661
## 368 -9.205661e-02 -0.054471022
## 369 -1.288215e-01  0.186953301
## 370 -4.081818e-02  0.316111103
## 371  5.598894e-02  0.714492897
## 372 -1.910027e-02  0.122870212
## 373 -8.738651e-02 -0.084919007
## 374 -7.119482e-02  1.716547913
## 375 -4.263708e-02  0.363649741
## 376 -1.085357e-01  0.385274786
## 377 -4.245307e-03  1.539317349
## 378 -3.176215e-02  0.162173358
## 379  3.259157e+00 -0.101410005
## 380 -9.875922e-02  0.094135470
## 381 -1.319789e-01  0.722420469
## 382 -3.546746e-02  0.390186947
## 383 -7.248849e-02 -0.014795160
## 384  3.465890e-02 -0.033324193
## 385 -1.214325e-01  0.079601268
## 386 -1.328524e-01  0.411987654
## 387 -8.221125e-02  0.163109410
## 388 -1.313652e-01 -0.001041108
## 389 -9.558721e-02 -0.053702518
## 390 -1.361250e-01  0.173617925
## 391 -2.390717e-02  0.077496800
## 392 -1.038248e-01 -0.048359790
## 393 -1.173975e-01 -0.021357940
## 394 -1.298416e-02  0.323065716
## 395 -5.994674e-02 -0.093401848
## 396 -1.098203e-01  0.546081057
## 397 -1.337753e-01  1.155409707
## 398 -6.977782e-02 -0.186336908
## 399  6.641151e-02  0.537115840
## 400 -8.726016e-02  0.152357356
## 401 -9.348558e-02  0.549652812
## 402 -9.427061e-02  0.318797673
## 403 -1.273842e-01  0.182938912
## 404 -1.241300e-01 -0.329348206
```

```
## 405 -1.148913e-01  0.081559731
## 406 -1.588241e-02 -0.084983099
## 407 -5.072923e-02  0.166010086
## 408 -1.168968e-01  0.016011068
## 409  5.100245e-02  0.743589406
## 410 -1.199120e-01  0.764232169
## 411 -1.310339e-01  0.156120710
## 412 -1.161216e-01  1.001867865
## 413  2.108761e-01  1.036649208
## 414 -1.047511e-01  0.103386453
## 415  4.688066e-02  0.144004497
## 416 -1.090504e-01  0.234750807
## 417 -4.872864e-03  0.403398346
## 418  5.334868e-03  0.761322171
## 419 -5.010685e-02  0.467761335
## 420 -4.857389e-02 -0.102874126
## 421 -1.271516e-01  0.079383819
## 422 -7.996592e-02 -0.091932879
## 423 -1.242648e-01  0.704845751
## 424  3.008137e-01  1.155044757
## 425 -7.901924e-02  0.167162065
## 426 -7.094605e-02  0.661208932
## 427 -1.159275e-01  0.095829041
## 428 -1.212416e-01  0.955430987
## 429 -1.175266e-01 -0.238570827
## 430 -1.106754e-01  0.744684387
## 431 -1.149546e-01 -0.217640181
## 432 -1.440611e-01  0.440826605
## 433 -1.166215e-01  0.176602205
## 434 -1.224440e-01  0.291755771
## 435 -8.941962e-02  1.008148096
## 436 -1.092944e-01  0.483427513
## 437 -2.949041e-02  1.066803765
## 438 -8.179488e-02  0.333894180
## 439 -1.296353e-01  0.162506688
## 440 -1.055226e-01  0.749426559
## 441 -1.136039e-01 -0.027015873
## 442 -6.844458e-02  0.536399063
## 443 -1.282159e-01 -0.036232683
## 444  7.219354e-01  0.019175567
## 445 -9.812406e-02  0.375126449
## 446 -8.914250e-02  0.220161567
## 447 -5.395658e-02  0.559130517
## 448  4.839988e-02  0.465082710
## 449  7.123672e-02  0.368946771
## 450 -1.235634e-01  0.293459230
```

```
## 451 -5.088348e-02  0.979429116
## 452 -7.252980e-02 -0.150712671
## 453 -9.029436e-02 -0.150303772
## 454 -1.118824e-01  2.129091957
## 455 -9.979869e-02  0.546968688
## 456 -1.415352e-01  0.278271315
## 457 -1.209505e-01 -0.182257871
## 458 -1.123430e-01  0.204134895
## 459 -3.794455e-03  0.970967451
## 460 -1.186985e-01  0.528885916
## 461 -1.078725e-01 -0.110664169
## 462 -1.842012e-02  0.366078461
## 463  1.902636e-02  1.066358635
## 464 -9.003325e-02  0.144233128
## 465 -1.303649e-01  0.509299315
## 466 -1.391292e-01  0.725878993
## 467  6.174288e-02  1.171657011
## 468 -1.386492e-01  1.018681527
## 469 -1.397769e-01  1.443448456
## 470 -1.013370e-01  0.195213747
## 471  5.437016e-02  0.877946612
## 472  2.440195e-01  0.068543314
## 473 -1.059153e-01  1.247567380
## 474 -9.591861e-02  0.149600486
## 475 -5.847864e-02  0.395235066
## 476 -1.257831e-01  0.315211075
## 477 -1.248600e-01  0.425852076
## 478  4.978452e-02  0.664407454
## 479  8.912579e-02  1.197665719
## 480 -1.170556e-01  0.820187311
## 481 -3.779764e-02  0.141538152
## 482 -5.259075e-02  0.790669645
## 483 -4.616968e-02  0.514985815
## 484 -1.279578e-01  0.224857479
## 485 -1.000789e-01  0.162634001
## 486 -1.257069e-01  2.503726238
## 487 -9.575461e-02  0.497246381
## 488 -1.310252e-01  0.922981170
## 489 -3.684837e-02  0.996691397
## 490  3.547583e-01  1.564036158
## 491 -7.622568e-02  0.174640794
## 492  2.887174e-01 -0.124129195
## 493 -9.752027e-02  0.305417153
## 494 -1.125330e-01  0.518189633
## 495  1.420886e-02  0.800653526
## 496 -1.143336e-01  0.092166142
```

```
## 497 -1.231945e-01  0.466586934
## 498 -5.881197e-02  0.738987924
## 499 -8.694054e-02  0.655423117
## 500 -8.585928e-02  0.763008896
## 501 -1.303518e-01  0.128367829
## 502 -5.791207e-02  0.906897614
## 503 -1.234215e-02  0.809089188
## 504  2.246829e-01  0.211526095
## 505 -1.171222e-01 -0.117893919
## 506 -1.082093e-01  0.532102646
## 507 -1.252662e-01 -0.164156473
## 508 -9.956136e-02  1.200340447
## 509 -9.367923e-02  0.232430429
## 510 -4.444845e-02  0.749584060
## 511 -1.139515e-01  0.444768104
## 512 -8.138947e-02  0.194458091
## 513 -4.737003e-02  0.155186146
## 514 -7.915728e-02 -0.024595757
## 515 -8.032226e-02  0.427058627
## 516 -1.112645e-01  0.653700401
## 517 -5.780805e-02  0.583879634
## 518  2.033531e-02  0.650268537
## 519 -6.480029e-02  1.897799371
## 520 -1.040528e-01  1.117246524
## 521 -9.614179e-02  0.865797213
## 522 -1.427771e-01  0.692628595
## 523 -5.368158e-02  1.089299566
## 524 -6.617615e-02  0.780889733
## 525 -7.188806e-02  0.039442156
## 526  1.440485e-05  1.995514895
## 527 -6.491203e-02  0.077159132
## 528 -1.081334e-01  0.824910170
## 529 -9.176328e-02  0.569079610
## 530 -1.033959e-01  0.612638937
## 531  9.494557e-03  1.068154514
## 532 -6.207050e-02  1.530161498
## 533 -1.130852e-01  0.055283754
## 534 -8.801560e-02  0.583669862
## 535 -1.197920e-01  0.861763003
## 536 -7.207752e-02  0.505098036
## 537 -7.524738e-02  0.234923432
## 538  5.066232e-03  0.698794324
## 539  3.050051e+00  1.484652813
## 540  4.854092e-01  1.072892169
## 541  3.830448e-01  0.701138535
## 542  5.591613e-01  0.642941048
```

```
## 543 -1.201189e-01  0.228979442
## 544  1.035898e-01  0.694510680
## 545 -8.380238e-02  0.690707497
## 546  7.204234e-02  0.831828835
## 547 -3.839763e-02  1.456051343
## 548  1.082709e-01  0.422980804
## 549 -1.889227e-02  0.873576340
## 550 -9.344368e-02  0.752076830
## 551 -9.877324e-02  0.669575428
## 552  5.721848e-02  0.865850442
## 553  2.893582e-01  1.161715864
## 554  3.600660e-02  1.259094354
## 555 -4.758881e-02  0.960328697
## 556 -9.595833e-02  0.749222924
## 557  2.777701e-02  1.259408884
## 558 -1.864156e-02  1.638010143
## 559 -1.056741e-01  1.294644039
## 560 -1.004306e-01  0.478782445
## 561 -1.355056e-01  0.331695331
## 562  3.119507e-01  0.935889094
## 563 -8.903074e-03  2.252889966
## 564 -9.223348e-02  1.346183766
## 565 -1.006569e-01  0.778397897
## 566 -4.654563e-02  0.635096896
## 567 -3.165365e-04  0.354079163
## 568  8.441829e-03  0.962099855
## 569  5.297428e-01  1.134362500
## 570 -4.503482e-02  1.042928982
## 571  9.807258e-01  0.748803749
## 572  5.649819e-01  2.123973830
## 573 -6.425032e-02  1.031385831
## 574  6.642005e-02  0.758984300
## 575 -5.651554e-03  0.599641537
## 576  4.921993e-01  1.988067465
## 577 -1.177053e-01  0.962816503
## 578 -1.581167e-02  1.310880566
## 579  4.427409e-02  1.401156610
## 580  1.735693e-01  1.075512413
## 581  5.299281e-01  0.776038589
## 582 -6.542631e-02  0.183815299
## 583  1.442520e-01  3.898163160
## 584  9.166087e-02  0.837267670
## 585 -2.575637e-02  0.789038612
## 586 -5.542486e-02  0.604822281
## 587 -9.559155e-02  0.627229420
## 588  1.021676e-01  2.006510215
```

```
## 589  6.714384e-01  1.810774856
## 590 -2.530425e-02  1.834658644
## 591 -9.562191e-02  0.530890509
## 592  1.492804e-01  1.446148208
## 593  5.134927e-01  0.730868281
## 594  3.305292e-01  1.272596023
## 595  1.734700e-01  1.288851594
## 596 -9.087303e-02  0.571143189
## 597 -8.182204e-02  0.641902111
## 598 -1.065788e-01  2.247477956
## 599 -1.072657e-01  0.688351413
## 600 -1.039020e-01  0.254752288
## 601 -6.868008e-02  1.315411020
## 602 -3.362040e-02  1.082662226
## 603 -6.975064e-02  1.215959107
## 604  1.207499e-01  0.544253890
## 605  1.083651e-01  1.301377411
## 606 -5.357695e-02  3.263503306
## 607 -1.074402e-01  0.941640003
## 608 -8.015572e-02  1.600033654
## 609 -7.833723e-02  1.143775321
## 610 -1.320783e-01  1.849107917
## 611 -1.251390e-01  1.705250372
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                  x1          x2         x3         x4           x5
## Factor1  0.17122515  0.17167709 0.17121013 0.16969112 0.0001516377
## Factor2 -0.02088721 -0.01240016 0.01339039 0.02428247 0.4112424684
##                  x6           x7            x8          x9
## Factor1 -0.00838655 -0.001352842 -0.0006715478  0.16296203
## Factor2  0.26953535  0.206722283  0.3793074794 -0.01689853
##                 x10
## Factor1  0.17190197
## Factor2 -0.02894241
##
## Slot "meanF":
##      Factor1      Factor2
## 1.222966e-18 4.939064e-18
##
## Slot "corF":
##               Factor1       Factor2
## Factor1  1.000000e+00 -6.209475e-15
## Factor2 -6.209475e-15  1.000000e+00
```

```
##
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 611
##
## Slot "center":
##            x1            x2            x3            x4            x5
##  6.298537e-18 -8.747733e-18  1.273260e-17 -1.475531e-18  3.230922e-17
##            x6            x7            x8            x9           x10
##  2.103691e-17  9.398678e-18 -2.298509e-17  3.827742e-18 -9.092899e-19
##
## Slot "eigenvalues":
##  [1] 5.7900498488 2.3189552681 1.0086871619 0.5741391921 0.1432389680
##  [6] 0.0991672355 0.0516167794 0.0094032935 0.0039695244 0.0007727282
##
## Slot "cov.control":
## NULL
##
##
## Slot "importance":
##                Factor1 Factor2
## SS loadings      5.785   2.324
## Proportion Var   0.579   0.232
## Cumulative Var   0.579   0.811
```

Since we use the correlation matrix (`cor = TRUE`) as the running matrix, the entries of the loading matrix are limitted between 0 and 1, which makes the explanations of the factors easy. From the `show` result of `faClassic4`, we see that its `Factor1` explains variables `x1-x4`, `x9`, `x10`; its `Factor2` explains variables `x5-x8` (with loadings larger than 0.48). From the `summary` result of `faClassic4`, we see that the first two factors accout for about 81.1% of its total variance.

Next we generate an object `faCov8` of class `FaCov` using the same data set.

```
## (8) robust, x = scale(stock611[,3:12]), cor = TRUE (correlation matrix)
faCov8 = FaCov(x = scale(stock611[,3:12]), factors = 2, cor = TRUE, method = "pca",
               scoresMethod = "regression", cov.control = rrcov::CovControlOgk()); str(faC

## Formal class 'FaCov' [package "robustfa"] with 25 slots
##   ..@ call              : language FaCov(x = scale(stock611[, 3:12]), factors = 2, cor
##   ..@ converged         : NULL
##   ..@ loadings          : num [1:10, 1:2] 0.119 0.359 0.758 0.76 0.955 ...
```

```
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "X1" "X2" "X3" "X4" ...
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   ..@ communality       : Named num [1:10] 0.61 0.781 0.93 0.922 0.916 ...
##   .. ..- attr(*, "names")= chr [1:10] "X1" "X2" "X3" "X4" ...
##   ..@ uniquenesses      : Named num [1:10] 0.3902 0.2186 0.0697 0.078 0.0835 ...
##   .. ..- attr(*, "names")= chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ cor               : logi TRUE
##   ..@ covariance        : num [1:10, 1:10] 0.00097 0.00069 0.000579 0.000716 0.003544 .
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ correlation       : num [1:10, 1:10] 1 0.761 0.461 0.437 0.22 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ usedMatrix        : num [1:10, 1:10] 1 0.761 0.461 0.437 0.22 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ reducedCorrelation: NULL
##   ..@ criteria          : NULL
##   ..@ factors           : num 2
##   ..@ dof               : NULL
##   ..@ method            : chr "pca"
##   ..@ scores            : num [1:611, 1:2] -9.709 -0.495 -11.094 -7.743 -11.764 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:611] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   ..@ scoresMethod      : chr "regression"
##   ..@ scoringCoef       : num [1:2, 1:10] -0.0502 0.2387 0.0137 0.2244 0.1501 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   .. .. ..$ : chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ meanF             : Named num [1:2] -0.0896 3.8692
##   .. ..- attr(*, "names")= chr [1:2] "Factor1" "Factor2"
##   ..@ corF              : num [1:2, 1:2] 1 -0.215 -0.215 1
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   .. .. ..$ : chr [1:2] "Factor1" "Factor2"
##   ..@ STATISTIC         : NULL
##   ..@ PVAL              : NULL
##   ..@ n.obs             : int 611
##   ..@ center            : Named num [1:10] -0.0947 -0.0941 -0.1038 -0.111 -0.0211 ...
##   .. ..- attr(*, "names")= chr [1:10] "x1" "x2" "x3" "x4" ...
##   ..@ eigenvalues       : num [1:10] 5.158 2.405 1.156 0.684 0.215 ...
```

```
##   ..@ cov.control       :Formal class 'CovControlOgk' [package "rrcov"] with 8 slots
##   .. .. ..@ niter   : num 2
##   .. .. ..@ beta    : num 0.9
##   .. .. ..@ mrob    : NULL
##   .. .. ..@ vrob    :function (x1, x2, ...)
##   .. .. ..@ smrob   : chr "scaleTau2"
##   .. .. ..@ svrob   : chr "gk"
##   .. .. ..@ trace   : logi FALSE
##   .. .. ..@ tolSolve: num 1e-14
```

```
summary(faCov8)
```

```
## An object of class "SummaryFa"
## Slot "faobj":
## An object of class "FaCov"
## Slot "call":
## FaCov(x = scale(stock611[, 3:12]), factors = 2, cor = TRUE, cov.control = rrcov::CovCon
##     method = "pca", scoresMethod = "regression")
##
## Slot "converged":
## NULL
##
## Slot "loadings":
##        Factor1      Factor2
## X1   0.1192720  0.771765320
## X2   0.3585258  0.808017502
## X3   0.7582825  0.596037255
## X4   0.7604049  0.586322722
## X5   0.9546105  0.072062014
## X6   0.4300313  0.111234443
## X7   0.8787653 -0.002537877
## X8   0.9349237 -0.028219013
## X9   0.0457114  0.931164519
## X10 -0.0712708  0.827516094
##
## Slot "communality":
##        X1        X2        X3        X4        X5        X6        X7
## 0.6098475 0.7814330 0.9302528 0.9219899 0.9164741 0.1973000 0.7722348
##        X8        X9       X10
## 0.8748785 0.8691569 0.6898624
##
## Slot "uniquenesses":
##         x1         x2         x3         x4         x5         x6
## 0.39015249 0.21856697 0.06974721 0.07801012 0.08352593 0.80269995
##         x7         x8         x9        x10
## 0.22776518 0.12512145 0.13084311 0.31013759
```

```
##
## Slot "cor":
## [1] TRUE
##
## Slot "covariance":
##               x1           x2           x3           x4           x5
## x1  0.0009697048 0.0006900809 0.0005791774 0.0007158345  0.003544139
## x2  0.0006900809 0.0008475793 0.0008140561 0.0010254135  0.006331124
## x3  0.0005791774 0.0008140561 0.0016246432 0.0020883490  0.015305586
## x4  0.0007158345 0.0010254135 0.0020883490 0.0027643850  0.019935095
## x5  0.0035441392 0.0063311244 0.0153055857 0.0199350950  0.267683690
## x6  0.0034527831 0.0056571414 0.0103198072 0.0133316938  0.230608732
## x7  0.0006109680 0.0012763008 0.0035366951 0.0046154544  0.054943764
## x8  0.0008058348 0.0038998309 0.0133393406 0.0176619690  0.197793421
## x9  0.0006637225 0.0006948153 0.0008016902 0.0010431006  0.002709252
## x10 0.0003900011 0.0004472703 0.0006135993 0.0008176474 -0.001724620
##               x6             x7            x8            x9
## x1   0.003452783  6.109680e-04  0.0008058348  0.0006637225
## x2   0.005657141  1.276301e-03  0.0038998309  0.0006948153
## x3   0.010319807  3.536695e-03  0.0133393406  0.0008016902
## x4   0.013331694  4.615454e-03  0.0176619690  0.0010431006
## x5   0.230608732  5.494376e-02  0.1977934210  0.0027092521
## x6   0.566707641  3.842233e-03  0.0818914671  0.0061225970
## x7   0.003842233  1.859508e-02  0.0549794490  0.0001098947
## x8   0.081891467  5.497945e-02  0.2210443502 -0.0007394073
## x9   0.006122597  1.098947e-04 -0.0007394073  0.0011415830
## x10 -0.003368852 -4.577043e-05  0.0007303139  0.0007642267
##               x10
## x1   3.900011e-04
## x2   4.472703e-04
## x3   6.135993e-04
## x4   8.176474e-04
## x5  -1.724620e-03
## x6  -3.368852e-03
## x7  -4.577043e-05
## x8   7.303139e-04
## x9   7.642267e-04
## x10  9.284809e-04
##
## Slot "correlation":
##             x1        x2        x3        x4         x5          x6
## x1  1.00000000 0.7611847 0.4614374 0.4372135  0.2199784  0.14728889
## x2  0.76118465 1.0000000 0.6937215 0.6698994  0.4203197  0.25812337
## x3  0.46143741 0.6937215 1.0000000 0.9854278  0.7339391  0.34010503
## x4  0.43721352 0.6698994 0.9854278 1.0000000  0.7328382  0.33682671
## x5  0.21997840 0.4203197 0.7339391 0.7328382  1.0000000  0.59208664
```

```
## x6  0.14728889 0.2581234 0.3401050 0.3368267  0.5920866  1.00000000
## x7  0.14387973 0.3214873 0.6434570 0.6437483  0.7787685  0.03742873
## x8  0.05504103 0.2849154 0.7039076 0.7144966  0.8131322  0.23137659
## x9  0.63083097 0.7063589 0.5886726 0.5871826  0.1549832  0.24071458
## x10 0.41101680 0.5041890 0.4995965 0.5103644 -0.1093946 -0.14686419
##              x7          x8          x9         x10
## x1   0.14387973  0.05504103  0.63083097  0.41101680
## x2   0.32148727  0.28491537  0.70635894  0.50418901
## x3   0.64345704  0.70390761  0.58867265  0.49959648
## x4   0.64374835  0.71449656  0.58718260  0.51036435
## x5   0.77876850  0.81313223  0.15498324 -0.10939464
## x6   0.03742873  0.23137659  0.24071458 -0.14686419
## x7   1.00000000  0.85755425  0.02385196 -0.01101538
## x8   0.85755425  1.00000000 -0.04654685  0.05097807
## x9   0.02385196 -0.04654685  1.00000000  0.74230430
## x10 -0.01101538  0.05097807  0.74230430  1.00000000
##
## Slot "usedMatrix":
##             x1        x2        x3        x4         x5          x6
## x1  1.00000000 0.7611847 0.4614374 0.4372135  0.2199784  0.14728889
## x2  0.76118465 1.0000000 0.6937215 0.6698994  0.4203197  0.25812337
## x3  0.46143741 0.6937215 1.0000000 0.9854278  0.7339391  0.34010503
## x4  0.43721352 0.6698994 0.9854278 1.0000000  0.7328382  0.33682671
## x5  0.21997840 0.4203197 0.7339391 0.7328382  1.0000000  0.59208664
## x6  0.14728889 0.2581234 0.3401050 0.3368267  0.5920866  1.00000000
## x7  0.14387973 0.3214873 0.6434570 0.6437483  0.7787685  0.03742873
## x8  0.05504103 0.2849154 0.7039076 0.7144966  0.8131322  0.23137659
## x9  0.63083097 0.7063589 0.5886726 0.5871826  0.1549832  0.24071458
## x10 0.41101680 0.5041890 0.4995965 0.5103644 -0.1093946 -0.14686419
##              x7          x8          x9         x10
## x1   0.14387973  0.05504103  0.63083097  0.41101680
## x2   0.32148727  0.28491537  0.70635894  0.50418901
## x3   0.64345704  0.70390761  0.58867265  0.49959648
## x4   0.64374835  0.71449656  0.58718260  0.51036435
## x5   0.77876850  0.81313223  0.15498324 -0.10939464
## x6   0.03742873  0.23137659  0.24071458 -0.14686419
## x7   1.00000000  0.85755425  0.02385196 -0.01101538
## x8   0.85755425  1.00000000 -0.04654685  0.05097807
## x9   0.02385196 -0.04654685  1.00000000  0.74230430
## x10 -0.01101538  0.05097807  0.74230430  1.00000000
##
## Slot "reducedCorrelation":
## NULL
##
## Slot "criteria":
## NULL
```

```
##
## Slot "factors":
## [1] 2
##
## Slot "dof":
## NULL
##
## Slot "method":
## [1] "pca"
##
## Slot "scores":
##           Factor1       Factor2
## 1    -9.709273573  1.894644e+00
## 2    -0.495116920 -2.014924e+00
## 3   -11.093939034  1.268609e+00
## 4    -7.742860023  9.018253e-01
## 5   -11.764249628  1.481259e+00
## 6   -11.001222960  1.554386e+00
## 7    -8.478971277  9.630272e-01
## 8   -18.119086883  3.854959e+00
## 9    -6.925200238  5.068011e-01
## 10   -9.513458417  1.356250e+00
## 11   -6.141387527  2.836343e-01
## 12  -14.568293554  2.954136e+00
## 13   -6.049592921  2.268206e-01
## 14   -6.905278840  5.384901e-01
## 15   -7.712117894  7.361095e-01
## 16   -6.123535843  3.449819e-01
## 17   -8.765547514  1.138934e+00
## 18   -6.741766213  4.736048e-01
## 19   -4.793294532  4.429266e-01
## 20   -5.708701959  1.802224e+00
## 21   -6.140348785  4.126910e-01
## 22   -5.221411911  3.566274e-01
## 23   -4.480122527 -1.751604e-01
## 24   -6.233095970  1.082891e+00
## 25   -7.207729763  1.552873e+00
## 26   -6.104847714  7.116941e-01
## 27   -8.042807610  1.292826e+00
## 28   -5.243334441  1.105778e+00
## 29   -3.832871804  2.566659e-01
## 30   -3.581853045 -1.389221e-02
## 31   -4.614581824  4.794754e-01
## 32   -5.000797249  1.663769e+00
## 33   -3.741278611  1.231880e-01
## 34   -3.399518421  3.407733e-01
```

```
## 35   -3.269016628  8.917028e-01
## 36   -3.044673649  2.204128e-01
## 37   -3.485460729  2.068878e-01
## 38   -3.320057025  1.077264e-01
## 39   -2.134370384 -6.364339e-01
## 40   -3.229580164  2.101295e+00
## 41   -2.059880354 -5.867230e-01
## 42   -2.888405781  5.528192e+00
## 43   -1.509316856 -1.054223e+00
## 44   -3.256779480  6.707567e+00
## 45   -1.646766238  1.080238e-01
## 46   -1.755051653  9.345861e-01
## 47   -1.747689250  8.811488e-01
## 48   -1.468221624 -4.068874e-01
## 49   -1.224857524 -6.217521e-03
## 50   -1.533242437 -6.550575e-01
## 51   -1.893896290  1.027133e+00
## 52   -1.560769269 -6.765955e-01
## 53   -2.379474086  4.172651e+00
## 54   -1.682326478  7.810233e-01
## 55   -1.556891688 -4.590369e-01
## 56   -4.864046880  1.384586e+01
## 57   -3.130073548  6.257578e+00
## 58   -1.315536327 -4.337819e-01
## 59  -13.935817486  4.877925e+01
## 60   -1.647581180  4.608505e-03
## 61   -2.411397790  2.057264e+00
## 62   -1.652538561  5.819656e-01
## 63   -2.240665899  2.559629e+00
## 64   -1.455043752 -7.206381e-01
## 65   -1.756830665  1.319835e+00
## 66   -2.400397341  4.298580e+00
## 67   -1.395529791 -9.451651e-01
## 68   -1.357032458 -9.326995e-01
## 69   -1.092359396 -6.603371e-01
## 70   -1.615358012  1.211697e+00
## 71   -1.532035033 -9.728141e-02
## 72   -1.744405987  1.492668e+00
## 73   -0.973387004 -1.061662e+00
## 74  -10.272510646  4.154875e+01
## 75   -5.784326963  2.348737e+01
## 76   -1.348725438 -1.172900e+00
## 77   -1.180463022 -1.028665e+00
## 78   -1.157553780 -1.420674e+00
## 79  -19.183214317  1.091884e+02
## 80   -1.103916263 -6.046023e-01
```

```
## 81   -1.174063530  3.201193e-01
## 82   -2.803424568  1.391055e+01
## 83   -1.883825589  7.051680e-01
## 84   -1.773973051  2.274490e+00
## 85   -1.289457142 -1.199329e+00
## 86   -1.327518941 -1.393159e-01
## 87   -1.638028713  5.508093e+00
## 88   -1.139044547 -4.290177e-01
## 89   -1.209097756 -9.455365e-01
## 90   -1.244483932 -1.031667e+00
## 91   -1.241943049 -1.845131e-01
## 92   -2.295780211 -5.978992e-01
## 93   -1.381643578  1.242699e+00
## 94   -1.923244947  5.447168e+00
## 95   -1.188878693 -8.443247e-01
## 96   -1.315049109  7.666900e-01
## 97   -1.466404371  3.247086e+00
## 98   -1.331925353 -8.009298e-01
## 99   -1.148260962 -1.324971e+00
## 100  -1.628729616  1.432521e+00
## 101  -1.303680397  3.789472e-01
## 102  -1.798791073  1.578384e+00
## 103  -1.985340595  1.533886e+00
## 104  -0.756395801 -2.720401e-01
## 105  -1.582072047  1.592349e+00
## 106  -0.965471632 -9.503469e-01
## 107  -1.259343016  1.450653e+00
## 108  -0.974437066  3.134039e+00
## 109  -1.045730269 -2.853621e-01
## 110  -1.190482227  5.555796e-01
## 111  -1.326577422  9.976322e-01
## 112  -1.228839899 -6.929636e-01
## 113  -8.931125376  3.537131e+01
## 114  -2.040304908  1.445926e+01
## 115  -0.996303115 -1.122608e+00
## 116  -0.947824895 -4.693887e-01
## 117  -0.161284198  2.651590e+00
## 118  -1.009960091 -6.260907e-01
## 119  -1.204260237 -7.858008e-01
## 120  -1.049006359  8.634428e-02
## 121  -0.764703512 -3.591435e-01
## 122  -1.331959455  9.329429e-01
## 123  -1.054524515 -4.267423e-01
## 124  -0.933451913  2.435594e+00
## 125  -0.594213901 -4.953255e-01
## 126  -0.867846711 -4.306409e-01
```

```
## 127  -0.656224094 -9.453749e-01
## 128  -0.935372726 -1.314354e+00
## 129  -5.868908291  2.284905e+01
## 130  -0.775304237 -5.857076e-01
## 131  -0.929161411  1.370434e+00
## 132  -0.857983276  1.360348e+00
## 133   0.005125746 -2.569849e-01
## 134  -0.965339503  4.500199e-01
## 135  -1.115573863 -6.852450e-01
## 136  -1.024592946  1.844059e+00
## 137  -0.733433435  2.904823e-02
## 138  -1.191991821  7.002085e+00
## 139  -1.137471265  2.363847e+00
## 140  -1.786588055  3.114282e+00
## 141  -1.138057226  3.246217e+00
## 142  -0.846981922  2.073984e-02
## 143  -0.884028404  1.522766e+00
## 144  -1.067378742  1.936993e+00
## 145  -1.292169843  6.243854e+00
## 146  -0.629632198 -1.365367e-01
## 147  -0.613245165 -7.781885e-01
## 148  -0.325574581  1.770806e+00
## 149  -0.534107271  2.211233e-01
## 150  -0.282825589  9.806819e-01
## 151  -0.436448609 -7.463709e-01
## 152  -0.839084592  1.709363e-01
## 153  -1.394575614  1.038246e+01
## 154  -0.199532043 -8.949564e-01
## 155  -0.251095061 -3.212150e-01
## 156  -1.431787014  4.343784e+00
## 157  -0.749676626 -5.783470e-01
## 158  -0.761624165  3.803690e-01
## 159  -0.799571369 -3.904217e-01
## 160  -0.148608765  7.034498e+00
## 161  -0.709511632  4.843395e-01
## 162  -0.803593789  1.190497e+00
## 163  -1.035989762  2.788364e+00
## 164  -0.923548432 -1.076412e-01
## 165  -0.688179453  5.473896e-01
## 166  -0.525348801  4.403111e+00
## 167  -0.391149339 -6.384283e-01
## 168  -0.960984859 -3.882964e-01
## 169  -0.508674368  2.110244e+00
## 170  -0.426985812  3.571417e-01
## 171  -0.691535279  5.724399e-01
## 172  -0.367735487 -1.139945e-01
```

```
## 173  -0.992792019  1.021947e+00
## 174  -0.367109427  4.677239e+00
## 175  -0.942855715  9.130077e+00
## 176  -0.767983146  4.799850e+00
## 177  -0.603143778 -6.251909e-01
## 178  -0.645624423 -8.660237e-01
## 179  -0.571028546 -5.775182e-01
## 180  -0.811520537  1.047259e+00
## 181  -0.235510454 -1.148483e+00
## 182  -0.108335362  4.571079e+00
## 183  -0.836373182 -1.013692e+00
## 184  -0.586459901  1.882657e+00
## 185  -0.778249224  6.298425e-01
## 186  -0.557473292 -3.874559e-01
## 187  -1.026747902  2.805737e+00
## 188  -0.377191006  3.728909e+00
## 189  -0.595500616 -2.066057e-01
## 190  -0.653502514  3.472727e+00
## 191  -0.681144196 -2.531212e-01
## 192  -0.474395210  2.509862e+00
## 193  -0.778304635  3.616557e+00
## 194   0.312995458 -8.073104e-01
## 195  -0.402921815 -2.265724e-01
## 196  -0.686455624 -6.976653e-01
## 197  -0.691599349  6.265397e-01
## 198   0.080635747  1.936362e+00
## 199  -0.671794429 -5.806207e-03
## 200  -0.639270205  7.160559e-01
## 201  -0.552022144 -1.339319e-01
## 202  -0.208747172  6.665768e-01
## 203   0.504986127 -6.006193e-01
## 204  -0.365587017 -9.408646e-01
## 205  -0.984063541  7.488115e-01
## 206   0.538120245  6.613821e+00
## 207  -0.650049003  8.135071e-01
## 208  -0.613464900  2.034097e-01
## 209  -0.385825579 -8.318326e-01
## 210  -0.413794710  2.352487e-01
## 211  -0.362928516 -1.627375e-01
## 212  -0.707680367 -9.635601e-01
## 213  -0.467564977  5.077512e+00
## 214  -0.177268821 -5.289137e-01
## 215  -0.509514367 -8.368704e-01
## 216  -0.155481078  1.418679e+01
## 217  -0.595639262  7.323814e-01
## 218  -0.163135690 -7.341219e-01
```

```
## 219  -0.222420625 -3.629724e-01
## 220  -0.117063185  2.965093e+00
## 221  -0.493585424  2.150855e+00
## 222  -0.391997710  2.788117e+00
## 223  -0.424751632 -5.490925e-01
## 224  -0.512536062 -5.126792e-02
## 225  -0.345460819  5.196338e+00
## 226  -0.541526610  7.000386e+00
## 227  -0.611612529 -6.140935e-01
## 228  -0.338569639  1.171576e+00
## 229  -0.343571626 -3.846232e-01
## 230  -0.399502199  1.053600e+01
## 231   0.077070791 -6.719024e-01
## 232  -0.079801464  4.579052e-01
## 233  -0.069197213  1.462778e+00
## 234  -0.621290358  6.207464e-01
## 235   0.185739449 -7.286148e-01
## 236  -0.050828656  7.453566e-02
## 237  -1.241620313  1.772097e+01
## 238   0.257261231 -5.455460e-01
## 239  -0.352515420 -6.636670e-01
## 240  -0.160842280  1.728565e+00
## 241  -0.811009564 -8.697788e-01
## 242  -0.280673917 -2.781701e-01
## 243  -0.518847813  1.321761e+00
## 244  -0.367336256 -9.775089e-01
## 245   0.655559405  6.918306e-01
## 246  -0.620532255  3.708048e-01
## 247  -0.069518882  4.011541e-01
## 248  -0.245962516 -9.727214e-01
## 249  -0.224137329 -9.582106e-01
## 250  -0.209984623 -1.246087e+00
## 251  -0.381478747 -1.436409e+00
## 252   0.098633673 -1.254959e+00
## 253   0.178464216  4.098234e+00
## 254  -0.271984666  5.935371e+00
## 255  -0.432907912 -3.797161e-01
## 256  -0.541659041 -1.279945e+00
## 257  -0.341451300  8.359774e-01
## 258  -0.128521903 -1.117316e+00
## 259   0.405639446 -6.613736e-01
## 260  -0.414230950  3.548690e+00
## 261   0.704469204 -1.116045e+00
## 262  -0.353547105 -1.020906e+00
## 263   0.071656667  4.119685e-01
## 264  -0.257541285 -1.303703e+00
```

```
## 265  -0.514398257 -1.007718e-01
## 266  -0.681563424  2.443514e+00
## 267  -0.006867482 -3.825940e-01
## 268  -0.625102102 -2.322984e-01
## 269  -0.066042177  9.309561e-01
## 270   0.157189673 -3.731588e-01
## 271   0.245403838  1.995249e+00
## 272   0.079025238  7.058375e+00
## 273  -0.905306235  6.055546e-01
## 274  -0.487359838 -4.037243e-01
## 275  -0.548825965  2.127753e+00
## 276  -0.443838028  2.994940e+00
## 277   0.331222114 -1.174073e+00
## 278  -0.237330626  3.361606e+00
## 279  -0.435471477 -9.978391e-01
## 280  -0.071948593 -1.062287e+00
## 281  -0.177106458  6.778622e-03
## 282  -0.583910581  5.313294e-02
## 283  -0.414421304  2.958941e+00
## 284  -0.028474681  9.872433e-01
## 285   2.714869581  1.235338e+01
## 286   0.182536114 -1.400475e-01
## 287   0.460334893  4.539986e+00
## 288   0.329284553  8.967412e+00
## 289   0.408991043  6.498834e-01
## 290  -0.149235735  3.052465e+00
## 291   0.226666878  1.058238e-01
## 292   0.077755453 -5.002938e-01
## 293  -0.071814898  4.804248e+00
## 294  -0.093432124 -1.286268e+00
## 295   0.990712917  7.376180e+00
## 296  -0.448419770 -5.176528e-01
## 297  -0.674339357 -7.931325e-02
## 298  -0.199543467 -7.902153e-01
## 299  -0.021605809  7.430680e-01
## 300  -0.029452614  1.882766e+00
## 301  -0.503499491  1.020254e-01
## 302   0.579461957  2.725483e+00
## 303  -0.045678829  1.250099e+01
## 304  -0.110224799 -8.131766e-02
## 305   0.530939549 -1.028863e+00
## 306  -0.106010003  1.263398e+00
## 307  -1.869121963  1.524363e+01
## 308   0.471220655  1.879896e+00
## 309  -0.411116681  6.074208e+00
## 310   0.090070920 -3.942923e-02
```

```
## 311   0.124459720  1.011561e+00
## 312  -0.290601499 -1.448610e-01
## 313   0.537841616 -9.195515e-01
## 314   0.587945003  1.443059e+00
## 315   0.439591901  7.660219e-02
## 316  -0.109302059 -1.064743e+00
## 317  -0.161537725 -1.859567e-01
## 318  -0.549539725 -2.412773e-02
## 319   0.684396421  1.889189e+00
## 320   1.105922460  4.014491e+00
## 321   1.226062758  5.199749e+00
## 322  -0.766797951  2.734413e+00
## 323   0.625328505  5.548625e-01
## 324   0.307109008  1.781026e+00
## 325   0.054695419  5.802498e-01
## 326   0.066916647  1.898046e+00
## 327   0.237988064 -4.145327e-01
## 328   0.366390348  1.489762e+00
## 329   0.484022581  5.405433e-01
## 330   0.606798777 -4.572287e-02
## 331   0.690042641  9.543101e-01
## 332  -0.128313172  6.237592e+00
## 333   0.726989705  7.237261e-01
## 334   0.150384305 -7.294504e-01
## 335   0.188107873 -6.162475e-01
## 336   0.263304692 -1.018002e+00
## 337  38.809138774 -1.639942e+01
## 338 -17.884384041  8.994281e+02
## 339   1.291136397  5.458006e+00
## 340  -0.040112857  3.303552e+00
## 341   1.154579369  3.661960e+00
## 342   0.382463002 -3.479828e-01
## 343   1.075296639  6.188986e+00
## 344   0.919543984  5.848943e+00
## 345  -0.053234708  1.033046e+01
## 346  -0.585113433  1.502621e+00
## 347   0.217913336 -1.115819e+00
## 348   0.123137487  2.271104e-02
## 349   0.221858610 -4.735066e-01
## 350   0.613015263  2.571292e+00
## 351   0.394754672  4.647275e-01
## 352   0.016003506 -2.962507e-02
## 353  -0.092099941  1.339668e+00
## 354   0.625231503  7.317998e-01
## 355  -0.268949122  4.312353e-01
## 356   1.038088923  2.185031e+00
```

```
## 357   0.754524491  2.069048e+00
## 358   0.459876861  4.001199e-01
## 359   0.296990708 -1.408422e+00
## 360  -0.515767939 -4.880129e-01
## 361   0.320187190  1.207077e+00
## 362   0.196653269 -4.242796e-01
## 363   0.453795154 -5.258136e-01
## 364   0.704614138  6.072769e-01
## 365   0.630014516  3.983032e+00
## 366   0.939721301  2.510277e+00
## 367   0.297563996 -1.018071e+00
## 368   0.085090076  2.241255e-01
## 369   0.296339472 -1.014663e+00
## 370   0.588160663  2.203861e+00
## 371   1.300220373  5.308845e+00
## 372   0.040839312  3.265392e+00
## 373   0.010892375  4.589645e-01
## 374   2.031232560  1.331031e+00
## 375   0.360986012  2.284021e+00
## 376   0.485667072 -1.255676e-01
## 377   2.331448303  3.204760e+00
## 378   1.008522498  2.275719e+00
## 379   3.114493609  1.179138e+02
## 380   0.160503222  6.177295e-02
## 381   0.633719483 -7.385287e-01
## 382   0.563428502  2.419961e+00
## 383  -0.144940118  1.120515e+00
## 384  -0.051039010  4.745049e+00
## 385   0.140367044 -7.482513e-01
## 386   0.427820819 -9.279786e-01
## 387   0.856655848  2.791518e-01
## 388  -0.028706792 -1.056182e+00
## 389  -0.043129178  1.430032e-01
## 390   0.239143519 -1.243294e+00
## 391   0.488304378  2.483525e+00
## 392  -0.015675135 -5.537215e-02
## 393   0.269000698 -8.008418e-01
## 394   0.648072526  3.129648e+00
## 395   0.525621119  9.940220e-01
## 396   0.821321089 -3.289148e-01
## 397   1.209930661 -8.664291e-01
## 398  -0.275417158  1.114929e+00
## 399   1.269570005  5.548280e+00
## 400   0.164727000  5.332701e-01
## 401   0.602307706  5.980289e-01
## 402   0.480467108  2.973481e-01
```

```
## 403   0.299138807 -9.660077e-01
## 404  -0.019426901 -1.157683e+00
## 405   0.305310478 -6.307973e-01
## 406  -0.041883618  2.958463e+00
## 407   0.674543481  1.620410e+00
## 408   0.428094084 -9.005171e-01
## 409   1.017263256  5.559850e+00
## 410   0.890340949 -5.100975e-01
## 411   0.298105010 -1.139266e+00
## 412   1.004708235 -2.071384e-01
## 413   2.459271856  1.045853e+01
## 414   0.269935905 -2.857224e-01
## 415   0.869421543  4.635922e+00
## 416   0.502418969 -4.027566e-01
## 417   0.885368629  3.089016e+00
## 418   1.205841928  3.763873e+00
## 419   0.602739618  2.147385e+00
## 420  -0.155522407  1.804109e+00
## 421   0.176463091 -1.006561e+00
## 422   0.054416960  6.532590e-01
## 423   0.838105261 -7.007646e-01
## 424   2.691448303  1.479638e+01
## 425   0.187630863  8.294930e-01
## 426   0.852080438  1.198513e+00
## 427   0.350906648 -6.701958e-01
## 428   1.089277548 -4.271451e-01
## 429   0.419880529 -1.132994e+00
## 430   0.750474108 -1.444288e-01
## 431  -0.131995331 -6.319479e-01
## 432   0.503938227 -1.443271e+00
## 433   0.258830990 -6.285829e-01
## 434   0.394654115 -7.754727e-01
## 435   1.782204249 -3.620367e-04
## 436   0.638606144 -2.444443e-01
## 437   1.838585913  2.144164e+00
## 438   0.493651174  6.444212e-01
## 439   0.294477283 -1.088941e+00
## 440   0.876908661 -6.300675e-02
## 441   0.302836564 -8.208965e-01
## 442   2.282236946  2.532844e-02
## 443   0.349315846 -1.297876e+00
## 444   0.162783335  2.857266e+01
## 445   0.666294638 -1.473960e-02
## 446   0.537924363  1.482730e-01
## 447   0.991470628  1.470585e+00
## 448   1.796213896  4.255850e+00
```

```
## 449   0.659995190  5.826022e+00
## 450   0.548906324 -9.518071e-01
## 451   1.643515406  1.451219e+00
## 452   0.113715110  7.774116e-01
## 453   0.185711137  2.006228e-02
## 454   2.416026558  4.556659e-02
## 455   0.940552745 -1.650126e-01
## 456   0.341873548 -1.435290e+00
## 457   0.051796670 -9.356162e-01
## 458   0.369445208 -5.651776e-01
## 459   1.798626559  3.189763e+00
## 460   0.866298355 -8.114762e-01
## 461   0.188745688 -5.547383e-01
## 462   0.782926821  2.569678e+00
## 463   1.537639285  4.407093e+00
## 464   0.685077950 -4.070827e-02
## 465   0.712955873 -1.097684e+00
## 466   1.201396615 -1.616873e+00
## 467   2.160752136  5.226450e+00
## 468   1.216739915 -1.212721e+00
## 469   1.529430377 -1.013702e+00
## 470   0.510115379 -2.660585e-01
## 471   1.972236219  5.278565e+00
## 472   0.557248026  1.144029e+01
## 473   1.601226440 -1.587018e-01
## 474   0.819583527 -3.498441e-01
## 475   1.004315914  1.181515e+00
## 476   1.056068416 -1.448975e+00
## 477   0.856174699 -1.119874e+00
## 478   1.670520164  4.506835e+00
## 479   2.304015011  6.274915e+00
## 480   1.146198461 -6.815664e-01
## 481   0.587193615  2.004310e+00
## 482   1.351012854  1.438180e+00
## 483   1.305809277  1.412172e+00
## 484   0.687131823 -1.314527e+00
## 485   0.810063673 -4.813144e-01
## 486   2.607053955 -4.808046e-02
## 487   0.954257103 -1.057140e-01
## 488   1.282940507 -1.172059e+00
## 489   1.916524454  1.699298e+00
## 490   5.293386122  1.307552e+01
## 491   0.493206393  5.509526e-01
## 492   0.021628714  1.331858e+01
## 493   0.856664357 -1.605668e-01
## 494   0.888077643 -5.811581e-01
```

```
## 495   1.658438029  3.567915e+00
## 496   0.341073263 -7.173174e-01
## 497   0.869203212 -1.016552e+00
## 498   1.414584231  1.019256e+00
## 499   1.262458153  1.047887e-01
## 500   1.213761008  2.979585e-01
## 501   0.596461155 -1.451830e+00
## 502   1.282039850  1.466766e+00
## 503   1.839733875  2.532237e+00
## 504   1.332689695  1.020143e+01
## 505   0.290604902 -1.001074e+00
## 506   1.866033575 -1.187739e+00
## 507   0.255048045 -1.305835e+00
## 508   1.879921749 -3.049530e-01
## 509   0.938029071 -3.358036e-01
## 510   1.452801784  1.514682e+00
## 511   0.924263267 -8.510906e-01
## 512   0.653372243  2.993126e-01
## 513   1.267533274  9.347383e-01
## 514   0.612865873  1.531433e-01
## 515   1.247123300  7.378730e-02
## 516   1.185454650 -6.804273e-01
## 517   1.318603997  1.154894e+00
## 518   2.136857631  3.177205e+00
## 519   2.716658886  1.087386e+00
## 520   1.756923562 -4.917903e-01
## 521   1.648711843 -3.414058e-01
## 522   1.139923801 -1.756291e+00
## 523   2.917920524  3.580970e-01
## 524   1.598733701  8.389111e-01
## 525   0.699262961  4.035205e-01
## 526   3.157231798  3.436270e+00
## 527   0.900210216  5.991203e-01
## 528   1.479859496 -6.949599e-01
## 529   1.351673455 -2.175209e-01
## 530   1.209574878 -5.304618e-01
## 531   2.370172538  2.858443e+00
## 532   2.393712105  8.628296e-01
## 533   0.702702439 -1.010347e+00
## 534   1.171058790 -8.026188e-03
## 535   1.349441947 -9.391240e-01
## 536   1.277494161  4.191773e-01
## 537   0.930790682  3.220260e-01
## 538   1.720350223  3.030796e+00
## 539  22.810632713  9.147826e+01
## 540   4.178088856  1.787539e+01
```

```
## 541   3.388158823  1.499789e+01
## 542   3.816791591  2.007573e+01
## 543   0.968647727 -1.289853e+00
## 544   1.981596171  6.092902e+00
## 545   1.592709386 -1.446849e-01
## 546   2.048565021  5.192004e+00
## 547   2.642155058  1.403959e+00
## 548   1.688565282  6.093259e+00
## 549   2.229982195  1.790582e+00
## 550   1.511737500 -3.458239e-01
## 551   1.466000986 -5.614511e-01
## 552   2.474601936  4.156156e+00
## 553   3.965110925  1.150869e+01
## 554   2.633789096  3.867048e+00
## 555   2.104231144  1.018257e+00
## 556   1.576803577 -4.885420e-01
## 557   3.651250065  2.556187e+00
## 558   3.055408745  2.054780e+00
## 559   2.126832228 -6.639268e-01
## 560   1.247183342 -6.445463e-01
## 561   1.626013960 -2.200049e+00
## 562   4.336809938  1.122344e+01
## 563   3.708495652  2.513576e+00
## 564   2.271223635 -3.215369e-01
## 565   2.095426291 -9.606363e-01
## 566   1.658453623  9.417615e-01
## 567   1.939391076  2.008271e+00
## 568   2.271670884  2.950355e+00
## 569   6.961883694  1.645512e+01
## 570   2.079855373  1.211237e+00
## 571   6.904236557  3.427124e+01
## 572   6.224582938  2.104175e+01
## 573   2.401268114  2.175641e-01
## 574   2.680810288  4.342330e+00
## 575   2.844670498  1.730392e+00
## 576   8.842041745  1.509597e+01
## 577   1.859072045 -1.247122e+00
## 578   2.594371840  2.015532e+00
## 579   3.510222649  3.708443e+00
## 580   4.006106532  6.430514e+00
## 581   5.437047773  1.793417e+01
## 582   0.981362069  4.086365e-01
## 583   7.537310061  6.476780e+00
## 584   3.330499882  4.695081e+00
## 585   2.031678387  1.555498e+00
## 586   1.754565436  4.623660e-01
```

```
## 587   1.584916438 -6.459443e-01
## 588   5.110319451  5.134597e+00
## 589   9.979105212  2.007192e+01
## 590   3.293911719  1.678997e+00
## 591   1.873467630 -9.429836e-01
## 592   4.716890441  6.179160e+00
## 593   4.504687150  1.797889e+01
## 594   5.261002911  1.175355e+01
## 595   3.986178097  7.404255e+00
## 596   1.707806393 -6.592585e-01
## 597   2.166973875 -5.801492e-01
## 598   3.556069672 -1.020010e+00
## 599   2.131351924 -1.407344e+00
## 600   1.195136315 -9.652539e-01
## 601   2.837908099 -1.619571e-01
## 602   2.758138217  9.703685e-01
## 603   2.786224446 -2.102319e-01
## 604   2.962097241  5.526348e+00
## 605   5.415261230  4.006615e+00
## 606   6.135566627 -8.571620e-01
## 607   2.500179523 -1.554264e+00
## 608   4.655421083 -1.630570e+00
## 609   3.281481976 -9.819970e-01
## 610   4.148457451 -2.836173e+00
## 611   4.660734273 -2.940146e+00
##
## Slot "scoresMethod":
## [1] "regression"
##
## Slot "scoringCoef":
##                 x1        x2        x3        x4          x5
## Factor1 -0.0502335 0.0136591 0.1500977 0.1517575  0.26283314
## Factor2  0.2386850 0.2244497 0.1117847 0.1083856 -0.08046737
##                 x6          x7         x8          x9        x10
## Factor1  0.1098215  0.24945181  0.2681726 -0.08845196 -0.1103341
## Factor2 -0.0105610 -0.09653429 -0.1110252  0.29867707  0.2776176
##
## Slot "meanF":
##     Factor1     Factor2
## -0.08958614  3.86917512
##
## Slot "corF":
##            Factor1    Factor2
## Factor1  1.0000000 -0.2154951
## Factor2 -0.2154951  1.0000000
##
```

```
## Slot "STATISTIC":
## NULL
##
## Slot "PVAL":
## NULL
##
## Slot "n.obs":
## [1] 611
##
## Slot "center":
##           x1           x2           x3           x4           x5
## -0.094674117 -0.094057722 -0.103796078 -0.111001987 -0.021050707
##           x6           x7           x8           x9          x10
## -0.070920649  0.006621815  0.096704667 -0.117127534 -0.092529714
##
## Slot "eigenvalues":
##  [1] 5.15840672 2.40502329 1.15641490 0.68381812 0.21455429 0.18508773
##  [7] 0.11888878 0.03343847 0.03126596 0.01310173
##
## Slot "cov.control":
## An object of class "CovControlOgk"
## Slot "niter":
## [1] 2
##
## Slot "beta":
## [1] 0.9
##
## Slot "mrob":
## NULL
##
## Slot "vrob":
## function (x1, x2, ...)
## {
##     (.mrobTau(x1 + x2, ...)[2]^2 - .mrobTau(x1 - x2, ...)[2]^2)/4
## }
## <bytecode: 0x7f9ffa4277b0>
## <environment: namespace:rrcov>
##
## Slot "smrob":
## [1] "scaleTau2"
##
## Slot "svrob":
## [1] "gk"
##
## Slot "trace":
## [1] FALSE
```

```
##
## Slot "tolSolve":
## [1] 1e-14
##
##
##
## Slot "importance":
##                Factor1 Factor2
## SS loadings      4.046   3.518
## Proportion Var   0.405   0.352
## Cumulative Var   0.405   0.756
```

From the `show` result of `faCov8`, we see that its `Factor1` explains variables `x3-x8` (with loadings larger than 0.43); its `Factor2` explains variables `x1-x4`, `x9`, `x10`. Thus `Factor1` (`Factor2`) of `faClassic4` and `Factor2` (`Factor1`) of `faCov8` are similar. In fact, this is not the general case. That is, in most of the situations, the explanations of the factors of `faClassic4` and `faCov8` should be the same. From the `summary` result of `faCov8`, we see that the first two factors accout for about 75.6% of its total variance.

The classical and robust scatterplots of the first two factor scores of the `stock611` data are shown in Figure 15. From the figure we see that `Factor1` (`Factor2`) of `faClassic4` and `Factor2` (`Factor1`) of `faCov8` are similar in patterns.

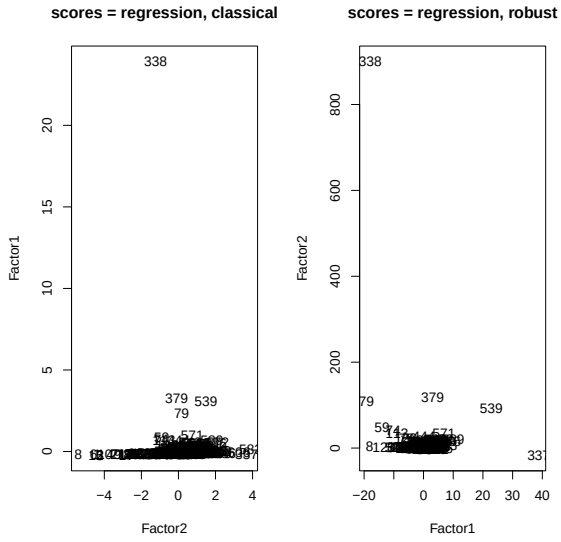


Figure 15: Classical and robust scatterplots of the first two factor scores of the stock611 data.

Furthermore, by inspecting the classical and robust ordered scores, we find that they are quite different. In the following, `orderedFsC[[1]]` and `orderedFsOgk[[1]]` are decreasing according to their first column; `orderedFsC[[2]]` and `orderedFsOgk[[2]]` are decreasing according to their second column.

```
orderedFsC = fsOrder(faClassic4@scores[,2:1]); orderedFsC
```

```
Lst=list(orderedFsC[[1]][1:10,], orderedFsC[[2]][1:10,]); Lst

## [[1]]
##      Factor2       Factor1
## 583 3.898163  1.442520e-01
## 337 3.681513 -1.520371e-01
## 606 3.263503 -5.357695e-02
## 486 2.503726 -1.257069e-01
## 563 2.252890 -8.903074e-03
## 598 2.247478 -1.065788e-01
## 454 2.129092 -1.118824e-01
## 572 2.123974  5.649819e-01
## 588 2.006510  1.021676e-01
## 526 1.995515  1.440485e-05
##
## [[2]]
##         Factor2    Factor1
## 338 -1.22930368 23.8967397
## 379 -0.10141001  3.2591570
## 539  1.48465281  3.0500510
## 79   0.17308239  2.3347627
## 571  0.74880375  0.9807258
## 59  -0.89391045  0.8830250
## 74  -0.92530280  0.7621726
## 444  0.01917557  0.7219354
## 589  1.81077486  0.6714384
## 113 -0.78396701  0.6627497
```

```
orderedFsOgk = fsOrder(faCov8@scores[,1:2]); orderedFsOgk
```

```
Lst=list(orderedFsOgk[[1]][1:10,], orderedFsOgk[[2]][1:10,]); Lst

## [[1]]
##       Factor1    Factor2
## 337 38.809139 -16.399416
## 539 22.810633  91.478265
## 589  9.979105  20.071925
## 576  8.842042  15.095968
## 583  7.537310   6.476780
## 569  6.961884  16.455123
## 571  6.904237  34.271241
```

```
## 572  6.224583  21.041747
## 606  6.135567  -0.857162
## 581  5.437048  17.934172
##
## [[2]]
##         Factor1   Factor2
## 338 -17.8843840 899.42807
## 379   3.1144936 117.91379
## 79  -19.1832143 109.18835
## 539  22.8106327  91.47826
## 59  -13.9358175  48.77925
## 74  -10.2725106  41.54875
## 113  -8.9311254  35.37131
## 571   6.9042366  34.27124
## 444   0.1627833  28.57266
## 75   -5.7843270  23.48737
```

4. Conclusions

We presented an object-oriented solution for robust factor analysis developed in the S4 class system of the programming environment R. In the solution, new S4 classes `"Fa"`, `"FaClassic"`, `"FaRobust"`, `"FaCov"`, `"SummaryFa"` are created. The classical factor analysis function `FaClassic()` and the robust factor analysis function `FaCov()` both can deal with maximum likelihood estimation, principal component analysis, and principal factor analysis methods. Consider the factor analysis methods (classical or robust), the data input (data or the scaled data), and the running matrix (covariance or correlation) all together, there are 8 combinations. We recommend (4) (classical factor analysis using the correlation matrix of the scaled data as the running matrix) vs (8) (robust factor analysis using the correlation matrix of the scaled data as the running matrix) for theoretical and computational reasons. The application of the solution to multivariate data analysis is demonstrated on the `hbk` data and the `stock611` data which themselves are parts of the package **robustfa**. A major design goal of the object-oriented solution was the openness to extensions by the development of new robust factor analysis methods in the package **robustfa** or in other packages depending on **robustfa**.

References

Chambers JM (1998). *Programming with Data: A Guide to the S Language*. Springer-Verlag, New York.

Davies PL (1987). "Asymptotic Behavior of S-Estimators of Multivariate Location Parameters and Dispersion Matrices." *The Annals of Statistics*, **15**, 1269–1292.

Donoho DL (1982). "Breakdown Properties of Multivariate Location Estimators." *Techni-*

cal report, Harvard University, Boston. URL http://www-stat.stanford.edu/~donoho/Reports/Oldies/BPMLE.pdf.

Filzmoser P, Fritz H, Kalcher K (2013). ***pcaPP**: Robust PCA by Projection Pursuit*. R package version 1.9-49, URL http://CRAN.R-project.org/package=pcaPP.

He X, Wang G (1997). "A qualitative robustness of S*-estimators of multivariate location and dispersion." *Statistica Neerlandica*, **51**, 257–268.

Johnson RA, Wichern DW (2007). *Applied Multivariate Statistical Analysis*. Pearson, New Jersey. 6th edition.

Maronna RA, Martin D, Yohai VJ (2006). *Robust Statistics: Theory and Methods*. John Wiley & Sons, New York.

Maronna RA, Yohai VJ (1995). "The Behavoiur of the Stahel-Donoho Robust Multivariate Estimator." *Journal of the American Statistical Association*, **90**, 330–341.

Maronna RA, Zamar RH (2002). "Robust Estimation of Location and Dispersion for High-Dimensional Datasets." *Technometrics*, **44**, 307–317.

OMG (2009a). "OMG Unified Modeling Language (OMG UML), Infrastructure, V2.2." *Current formally adopted specification*, Object Management Group. URL http://www.omg.org/spec/UML/2.2/Infrastructure/PDF.

OMG (2009b). "OMG Unified Modeling Language (OMG UML), Superstructure, V2.2." *Current formally adopted specification*, Object Management Group. URL http://www.omg.org/spec/UML/2.2/Superstructure/PDF.

Pison G, Rousseeuw PJ, Filzmoser P, Croux C (2003). "Robust Factor Analysis." *Journal of Multivariate Analysis*, **84**, 145–172.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Robbins JE (1999). "Cognitive Support Features for Software Development Tools." *Phd thesis*, University of California, Irvine. URL http://argouml.tigris.org/docs/robbins-dissertation/.

Robbins JE, Redmiles DF (2000). "Cognitive Support, UML Adherence, and XMI Interchange in Argo/UML." *Information and Software Technology*, **42**, 79–89.

Rocke DM (1996). "Robustness Properties of S-Estimators of Multivariate Location and Shape in High Dimension." *The Annals of Statistics*, **24**, 1327–1345.

Rousseeuw PJ (1985). "Multivariate Estimation with High Breakdown Point." In W Grossmann, G Pflug, I Vincze, W Wertz (eds.), *Mathematical Statistics and Applications Vol. B*, pp. 283–297. Reidel Publishing, Dordrecht.

Rousseeuw PJ, Croux C, Todorov V, Ruckstuhl A, Salibian-Barrera M, Verbeke T, Maechler M (2013). ***robustbase**: Basic Robust Statistics*. R package version 0.9-10, URL http://CRAN.R-project.org/package=robustbase.

Rousseeuw PJ, Driessen KV (1999). "A Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics*, **41**, 212–223.

Ruppert D (1992). "Computing S-Estimators for Regression and Multivariate Location/Dispersion." *Journal of Computational and Graphical Statistics*, **1**, 253–270.

Salibian-Barrera M, Yohai VJ (2006). "A Fast Algorithm for S-regression Estimates." *Journal of Computational and Graphical Statistics*, **15**, 414–427.

Stahel WA (1981a). "Breakdown of Covariance Estimators." *Research Report 31*, ETH Zurich. Fachgruppe für Statistik.

Stahel WA (1981b). "Robuste Schätzungen: Infinitesimale Optimalität und Schätzungen von Kovarianzmatrizen." *Phd thesis no. 6881*, Swiss Federal Institute of Technology (ETH), Zürich. URL http://e-collection.ethbib.ethz.ch/view/eth:21890.

Todorov V (2013). ***rrcov**: Scalable Robust Estimators with High Breakdown Point*. R package version 1.3-4, URL http://CRAN.R-project.org/package=rrcov.

Todorov V, Filzmoser P (2009). "An Object-Oriented Framework for Robust Multivariate Analysis." *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Wang JH, Zamar R, Marazzi A, Yohai VJ, Salibian-Barrera M, Maronna R, Zivot E, Rocke D, Martin D, Konis K (2013). ***robust**: Insightful Robust Library*. R package version 0.4-15, URL http://CRAN.R-project.org/package=robust.

Wang XM (2009). *Applied Multivariate Analysis*. Shanghai University of Finance & Economics Press, Shanghai. 3rd edition (This is a Chinese book).

Woodruff DL, Rocke DM (1994). "Computable Robust Estimation of Multivariate Location and Shape in High Dimension Using Compound Estimators." *Journal of the American Statistical Association*, **89**, 888–896.

Xue Y, Chen LP (2007). *Statistical Modeling and R Software*. Tsinghua University Press, Beijing. (This is a Chinese book).

Zhang YY (2013). ***robustfa**: An Object Oriented Solution for Robust Factor Analysis*. R package version 1.0-5, URL http://CRAN.R-project.org/package=robustfa.

Affiliation:

Ying-Ying Zhang
Department of Statistics and Actuarial Science
College of Mathematics and Statistics
Chongqing University
Chongqing, China
E-mail: robertzhangyying@qq.com
URL: http://baike.baidu.com/view/7694173.htm