東海大學應用統計研究所

碩士論文

指導教授：魏文翔　　博士

Sensitivity analysis and robustness in principal component analysis

研究生：林正茹

中華民國九十二年七月

致　謝

承蒙指導教授魏文翔博士在本人研究期間的指導和協助，得以順利完成本論文。在研究過程中，不僅在學識上有所斬獲，也讓我自己對未來的規劃有更明確的目標和方向。在此獻上我最誠摯的謝意和祝福。

論文口試期間，對於黃教授連成和黃助理教授郁芬百忙之中仍抽空審閱本論文，並給予我許多寶貴的意見，使之更加完善。在此致上由衷的感謝。此外，也要感謝所上所有老師在我求學期間對我知識的傳授和協助。

最後，感謝所上所有助教、同窗好友及林庭安、陳梓庭等摯友，在這兩年中對我的關懷和協助，更要感謝家人背後給予我的支持和鼓勵，使我得以順利完成學業，在此以本篇論文獻給所有曾經默默幫助過我的人。

Summary

Lemma 2.1 of Sibson (1979) is generalized to the second order perturbation of a symmetric matrix. Thus, the second order theoretical influence functions for the eigenvalues and eigenvectors can be developed to detect the masked influential observations in principal component analysis. In addition, a robust principal component can thus be developed by downweighting the identified influential observations. Numerical example illustrates the techniques.

Keywords: influence function, second order influence function, principal component analysis, outlier, robustness, kernel function.

Contents

1. Introduction

A principal component analysis is concerned with explaining the variance-covariance structure of a set of variables through a few linear combinations of these variables. Its general objectives are data reduction and interpretation. Although p components are required to reproduce the total system variability, often much of this variability can be accounted for by a small number k of the principal components. If so, there is almost as much information in the k components as there is in the original p variables. The k principal components can then replace the initial p variables, and the original data set, consisting of n measurements on p variables, is reduced to a data set consisting of n measurements on k principal components. An analysis of principal components often reveals relationships that were not previously suspected and thereby allows interpretations that would not ordinarily result.

Algebraically, principal components are particular linear combinations of the p random variables. Geometrically, these linear combinations represent the selection of a new coordinate system obtained by rotating the original system with the p random variables as the coordinate axes. The new axes represent the directions with maximum variability and provide a simpler and more parsimonious description of the covariance structure. ***The eigenvalues and eigenvectors of the covariance (correlation) matrix are the essence of a principal component analysis.*** The eigenvectors determine the directions of maximum variability, and the eigenvalues specify the variances. When the first few eigenvalues are much larger than the rest, most of the total variance can be explained in fewer than p dimensions.

There are recently been interest in influence analysis in principal component analysis. Several authors, such as Critchley (1985), Shi (1997), Croux & Haesbroeck (1999) and C.Croux & A.Ruiz-Gazen, in a Université Libre de Bruxelles technical report, have suggested statistical diagnostics and graphical displays for detecting outliers in principal component analysis for one population, such as side-by-side boxplots of the scores obtained from a robust principal component analysis and index plots based on empirical influence functions. Radhakrishnan & Kshirsagar (1981), Critchley (1985), Jolliffe (1986), Tanaka (1988), Brooks (1994) among others discussed the use of the influence functions of eigenvalues and eigenvectors.

Geometric intuition indicates that principal component analyses are heavily prone to influential observations. That is, a big enough outlier can by itself determine an entire component. Worse still, this component can even have one of the largest eigenvalues! Therefore, more emphases should be put on the detection of the outliers. There are several effective methods available for identifying these influential observations. Hampel's (1974) influence function is a useful tool to develop influence diagnostics. Based on Lemma 2.1 of Sibson (1979), Critchley (1985) has derived the influence function in principal component analysis. However, the above methods put more emphasis on the influence of single observations than on the 『masking effect』 of two observations. Unfortunately, sometimes the influence of the masked observations is evident only after a masked observation is deleted. That is, the delete-one diagnostic based on the original data for a masked observation is significantly different from the one for this observation based on the data without some observation.

In section 2, we extend Lemma 2.1 of Sibson (1979) from the first order perturbation to the second order perturbation of a symmetric matrix. The explicit expressions of each coefficient can thus be obtained. Further, the result can be used to derive the second order influence functions for characterizing the masking effect in principal component analysis. Also, two sample versions of these influence functions are given. In section 3, a robust method is proposed to downweight the outlying observations. An illustrative example is presented in section 4. Finally, concluding discussions are given in section 5.

2. Influence function

To derive the second order influence functions for unmasking the masked observations, the generalization of Lemma 2.1 of Sibson (1979) can be stated as follows.

Theorem 1: Let B,C_1,C_2,D_1,D_2,D_{12} be symmetric matrices, λ be a simple eigenvalue of B and ***e*** be an associated eigenvector of unit length. Let B be perturbed to

$$B(e_1,e_2) = B + e_1 C_1 + e_2 C_2 + \frac{1}{2} e_1{}^2 D_1 + \frac{1}{2} e_2{}^2 D_2 + \frac{1}{2} e_1 e_2 D_{12} + O(e^3)$$

where $e = \max(e_1, e_2)$ and assume that the corresponding perturbations of λ,e are

$$l(e_1,e_2) = l + e_1 m_1 + e_2 m_2 + \frac{1}{2} e_1{}^2 v_1 + \frac{1}{2} e_2{}^2 v_2 + \frac{1}{2} e_1 e_2 v_{12} + O(e^3)$$

$$e(e_1,e_2) = e + e_1 f_1 + e_2 f_2 + \frac{1}{2} e_1{}^2 g_1 + \frac{1}{2} e_2{}^2 g_2 + \frac{1}{2} e_1 e_2 g_{12} + O(e^3)$$

Then

$$m_1 = e^T C_1 e,\quad m_2 = e^T C_2 e,\quad f_1 = -(B - lI)^+ C_1 e,\quad f_2 = -(B - lI)^+ C_2 e$$

$$v_1 = e^T (D_1 - 2C_1 (B - lI)^+ C_1) e,\quad v_2 = e^T (D_2 - 2C_2 (B - lI)^+ C_2) e$$

$$g_1 = (B - lI)^+ \left[(2m_1 I - 2C_1) f_1 + (v_1 I - D_1) e \right],\quad g_2 = (B - lI)^+ \left[(2m_2 I - 2C_2) f_2 + (v_2 I - D_2) e \right]$$

$$v_{12} = e^T (D_{12} - 2(B - lI)^+ (C_1 C_2 + C_2 C_1)) e$$

$$g_{12} = (B - lI)^+ \left[(2m_1 I - 2C_1) f_2 + (2m_2 I - 2C_2) f_1 + (v_{12} - D_{12}) e \right]$$

See the proof in Appendix 1.

Let F be a cumulative distribution function defined on R^p, let $m(F) = \int x dF(x)$ and let $\Omega(F) = \int \{x - m(F)\}\{x - m(F)\}^T dF(x)$ denote its covariance matrix assumed to exist, have distinct eigenvalues $l_1(F) > ... > l_p(F)$ and corresponding orthonormal eigenvectors $e_1(F),...,e_p(F)$ all of which we stack into a single vector $T(F) \in R^{p(p+1)}$. Theorem 1 can be used to derive the second order influence functions. Let d_{Z_1} and d_{Z_2} denote the distributions giving unit masses to points Z_1 and Z_2 in R^p, respectively. Then for $0 \le e_1, e_2 \le 1$ we have the identity

$$\begin{aligned}
&\Omega(F_{e_1,e_2}) \\
&= \Omega\{(1-e_1-e_2)F + e_1 d_{Z_1} + e_2 d_{Z_2}\} \\
&= \Omega(F) + e_1\{(Z_1 - m(F))(Z_1 - m(F))^T - \Omega(F)\} + e_2\{(Z_2 - m(F))(Z_2 - m(F))^T - \Omega(F)\} \\
&\quad - e_1^2(Z_1 - m(F))(Z_1 - m(F))^T - e_2^2(Z_2 - m(F))(Z_2 - m(F))^T - 2e_1e_2(Z_1 - m(F))(Z_2 - m(F))^T
\end{aligned}$$

where $F_{e_1,e_2} = (1-e_1-e_2)F + e_1 d_{z_1} + e_2 d_{z_2}$. Assume that the corresponding perturbations of l_j and e_j for j=1,…,p are

$$l_j(e_1,e_2) = l_j + e_1 m_{1j} + e_2 m_{2j} + \frac{1}{2}e_1{}^2 v_{1j} + \frac{1}{2}e_2{}^2 v_{2j} + \frac{1}{2}e_1 e_2 v_{12,j} + O(e^3),\ \ e = \max(e_1,e_2)$$

$$e_j(e_1,e_2) = e_j + e_1 f_{1j} + e_2 f_{2j} + \frac{1}{2}e_1{}^2 g_{1j} + \frac{1}{2}e_2{}^2 g_{2j} + \frac{1}{2}e_1 e_2 g_{12,j} + O(e^3),\ \ e = \max(e_1,e_2)$$

where the rank order $l_1(e_1,e_2) > \Lambda > l_p(e_1,e_2)$ is unchanged for sufficiently small e_1 and e_2. By writing $Z_1 - m(F) = \sum a_{1j} e_j$ and $Z_2 - m(F) = \sum a_{2j} e_j$ and then simplifying, we have the first order terms,

$$m_{1j} = a_{1j}^2 - l_j,\ \ m_{2j} = a_{2j}^2 - l_j,$$

$$f_{1j} = -a_{1j} \Sigma_{(j)} a_{1k} (l_k - l_j)^{-1} e_k,\ f_{2j} = -a_{2j} \Sigma_{(j)} a_{2k} (l_k - l_j)^{-1} e_k$$

and the second order terms,

$$v_{1j} = -2a_{1j}^2\{1 + \Sigma_{(j)} a_{1k}^2 (l_k - l_j)^{-1}\},\ \ v_{2j} = -2a_{2j}^2\{1 + \Sigma_{(j)} a_{2k}^2 (l_k - l_j)^{-1}\},\ v_{12,j} = -4a_{1j}a_{2j},$$

$$g_{1j} = -2\Sigma_{(j)} a_{1k}^2 (l_k - l_j)^{-1} f_{1j} - 2a_{1j}^3 \Sigma_{(j)} a_{1k} (l_k - l_j)^{-2} e_k,$$

$$g_{2j} = -2\Sigma_{(j)} a_{2k}^2 (l_k - l_j)^{-1} f_{2j} - 2a_{2j}^3 \Sigma_{(j)} a_{2k} (l_k - l_j)^{-2} e_k$$

and

$$g_{12,j} = -2a_{2j}[\Sigma_{(j)} a_{2k} (l_k - l_j)^{-2}(m_{1j} + l_k)]e_k - 2a_{1j}[\Sigma_{(j)} a_{1k} (l_k - l_j)^{-2}(m_{2j} + l_k)]e_k$$

$$-2\Sigma_{(j)} a_{1k} a_{2k} (l_k - l_j)^{-1} [\Sigma_{(j)} (a_{1k} + a_{2k})(l_k - l_j)^{-1}] e_k - 4a_{2j} \Sigma_{(j)} a_{1k} (l_k - l_j)^{-1} e_k$$

where $\sum_{(j)}$ denotes summation over all terms except j. The details for deriving these terms are given in Appendix 2.

The proposed second order influence function $I(Z_1, Z_2; T, F)$ of the statistical functional T at F is equivalent to the second order partial derivative of $T\{(1-e_1-e_2)F + e_1 d_{Z_1} + e_2 d_{Z_2}\}$ with respect to e_1 and e_2 evaluated at $e_1 = e_2 = 0$, that is,

$$I(Z_1, Z_2; T, F) = \{\frac{\partial^2 T\{(1-e_1-e_2)F + e_1 d_{Z_1} + e_2 d_{Z_2}\}}{\partial e_1 \partial e_2}\}_{e_1=e_2=0}$$

$I(Z_1, Z_2; T, F)$ is also equivalent to the second order partial derivative of $T\{(1-e_1^*)[(1-e_2^*) F + e_2^* d_{Z_2}] + e_1^* d_{Z_1}\}$ with respect to e_1^* and e_2^* evaluated at $e_1^* = e_2^* = 0$, that is,

$$I(Z_1, Z_2; T, F) = \{\frac{\partial^2 T\{(1-e_1^*)[(1-e_2^*)F + e_2^* d_{Z_2}] + e_1^* d_{Z_1}\}}{\partial e_1^* \partial e_2^*}\}_{e_1^*=e_2^*=0}$$

where $e_1^* = e_1$ and $e_2^* = e_2/(1-e_1)$.

Thus we have at once the influence functions for any eigenvalues or eigenvectors are given by

$$I(Z_1, Z_2; l_j, F) = [\frac{\partial^2 l_j(e_1, e_2)}{\partial e_1 \partial e_2}]_{e_1=e_2=0} = \frac{1}{2} v_{12,j}$$

$$I(Z_1, Z_2; e_j, F) = [\frac{\partial^2 e_j(e_1, e_2)}{\partial e_1 \partial e_2}]_{e_1=e_2=0} = \frac{1}{2} g_{12,j}$$

Note that the influence function for T(F) is unbounded in each of its components.

F is unknown but can be estimated by $\hat{F}$, the empirical cumulative distribution function based on a random sample $x_1, ..., x_n$ from F. Note that $m(\hat{F}) = \bar{x}$, the sample mean,

$\Omega(\hat{F}) = n^{-1}\sum(x_i - \overline{x})(x_i - \overline{x})^T$ and write $\hat{l}_j$ and $\hat{e}_j$ for $l_j(\hat{F})$ and $\hat{e}_j(\hat{F})$ respectively. The original observations x_{ij} and x_{kj} $(i = 1,...n, k = 1,..., n, i \neq k, j = 1,..., p)$ are transformed to new principal component values y_{ij} and y_{kj} given by $y_{ij} = (x_i - \overline{x})^T \hat{e}_j$ and $y_{kj} = (x_k - \overline{x})^T \hat{e}_j$. With sample versions of influence functions, interest focuses on their values at the observations. Now putting $Z_1 = x_i$, $Z_2 = x_k$ and replacing F with $\hat{F}$ in $Z_1 - m(F) = \sum a_{1j} e_j$ and $Z_2 - m(F) = \sum a_{2j} e_j$, a_{1j} and a_{2j} become y_{ij} and y_{kj}, respectively.

The proposed empirical influence function $\hat{I}$ at $Z_1 = x_i$ and $Z_2 = x_k$ $(i = 1,..., n,$ $k = 1,..., n, \mathrm{i} \neq \mathrm{k})$ is obtained by replacing F by $\hat{F}$ in I is given by

$$\hat{I}_{ik,j}(l) \equiv \hat{I}(x_i, x_k; l_j, \hat{F}) = -2 y_{ij} y_{kj} = -2(x_i - \overline{x})^T \hat{e}_j (x_k - \overline{x})^T \hat{e}_j .$$

Let d_i, d_k denote the distributions giving unit masses to x_i and x_k, respectively. The deleted versions of $\hat{F}$ are given by,

$$\hat{F}_{(i)} = \{1 + (n-1)^{-1}\}\hat{F} - (n-1)^{-1} d_i \ , \quad \hat{F}_{(k)} = \{1 + (n-1)^{-1}\}\hat{F} - (n-1)^{-1} d_k$$

$$\hat{F}_{(i,k)} = \{1 + (n-2)^{-1} + (n-2)^{-1}\}\hat{F} - (n-2)^{-1} d_i - (n-2)^{-1} d_k$$

By taking $F = \hat{F}$ and $e = -1/(n-1)$, and then approximating the influence function $I(x_i; T, F)$ by the corresponding sample influence function, $(n-1)\{T(\hat{F}) - T(\hat{F}_{(i)})\}$, and approximating the influence function $I(x_i; T, (1 - e^*)F + e^* d_k)$ by the corresponding sample influence function, $(n-1)\{T(\hat{F}_{(k)}) - T(\hat{F}_{(i,k)})\}$, then the sample interaction influence function values, denoted $\tilde{I}$, are defined by

$$\tilde{I}(x_i, x_k; T, F) = (n-1)^2 \{T(\hat{F}) - T(\hat{F}_{(i)}) - T(\hat{F}_{(k)}) + T(\hat{F}_{(i,k)})\}$$

The approximation used in the second influence function $I(x_i;T,(1-e^*)F+e^*d_k)$ requires e^* to be 1/(n-1), which is equivalent to the absolute value of e used in the approximation of $I(x_i;T,F)$.

They measure the effects of omitting the ith observation, the kth observation, and the ith and the kth observations at once from a random sample of size n. Thus, we have

$$\tilde{I}_{ik,j}(l) \equiv \tilde{I}(x_i, x_k; l_j, F) = (n-1)^2\{\hat{l}_j - \hat{l}_{(i)j} - \hat{l}_{(k)j} + \hat{l}_{(i,k)j}\}$$

$$\tilde{I}_{ik,j}(e) \equiv \tilde{I}(x_i, x_k; e_j, F) = (n-1)^2\{\hat{e}_j - \hat{e}_{(i)j} - \hat{e}_{(k)j} + \hat{e}_{(i,k)j}\}$$

where $\hat{l}_{(i)j}$ is the j-th largest eigenvalue of $\Omega(\hat{F}_{(i)})$, again assumed simple, and $\hat{e}_{(i)j}$ the corresponding normalized eigenvector. We can similarly define $\hat{l}_{(k)j}$, $\hat{e}_{(k)j}$, $\hat{l}_{(i,k)j}$ and $\hat{e}_{(i,k)j}$.

3. **Robustness**

In this section, we propose identifying influential observations first, then apply a robust method to downweight these influential observations. Since the sample influence function for observation ***i*** is

$$\tilde{I}_i(l) \equiv \tilde{I}(x_i; l, \hat{F}) = -(n-1)\{\hat{l}_{(i)} - \hat{l}\},$$

the sample covariance matrix obtained by downweighting these influential observations is

$$\Omega_v(\hat{F}) = n^{-1}\sum_{i=1}^{n} v\left(\frac{\tilde{I}_i - m}{s}\right)(x_i - \bar{x})(x_i - \bar{x})^T,$$

where $\hat{l}$ is some specified eigenvalue of $\Omega_v(\hat{F})$, v(t) is some weight function and m and s are the sample median and sample median absolute deviation of $\tilde{I}_1,...,\tilde{I}_n$, respectively. The observations with large values of the sample influence function $\tilde{I}_i$ are downweighted by v(t). Several weight functions are available. The following choices of v(t) are recommended.

(1) box kernel

$$v_b(t) = \begin{cases} 1 & \|t\| \le 1 \\ 0 & \|t\| > 1 \end{cases}$$

(2) triangular kernel

$$v_t(t) = \begin{cases} 1-\|t\| & \|t\| \le 1 \\ 0 & \|t\| > 1 \end{cases}$$

(3) Parzen type of kernel

$$v_p(t) = \begin{cases} 1-\dfrac{3t^2}{16} & \|t\| \le 2 \\ \dfrac{-t^2+5\|t\|-6}{4} & 2 < \|t\| \le 3 \\ 0 & \|t\| > 3 \end{cases}$$

(4) Gaussian kernel

$$v_g(t) = \frac{1}{\sqrt{2p}} e^{-\frac{t^2}{2}}$$

(5) biweight kernel

$$v_{bi}(t) = \begin{cases} \dfrac{15}{16}\{1-t^2\}^2 & \|t\| \le 1 \\ 0 & \|t\| > 1 \end{cases}$$

(6) Epanachnikov kernel

$$v_e(t) = \begin{cases} \dfrac{3\sqrt{5}}{4}(1-0.2t^2) & \|t\| \le \sqrt{5} \\ 0 & \|t\| > \sqrt{5} \end{cases}$$

4. Example

We apply the above techniques to the data, including some statistics for the states in USA (『state.x77』 in SPLUS and given in Appendix 3.). There are 50 observations and 8 variables.

Now there is always the question of how many components to retain. For the moment, we shall employ the statistic found in Anderson (1963), which is

$$c^2 = -(v)\sum_{j=k+1}^{p}\ln(l_j) + (v)(p-k)\ln\left[\frac{\sum_{j=k+1}^{p} l_j}{(p-k)}\right]$$

where c^2 has (1/2)(p-k-1)(p-k+2) degrees of freedom and v represents the number of degrees of freedom associated with the covariance matrix. The above rule is based on the assumption that the characteristic roots associated with the deleted principal components are not significantly different from each other, that is, the last (p-k) roots we should neglect them. The first test that could be applied is for the hypothesis $H_0 : l_1 = l_2 = l_3 = l_4 = l_5 = l_6 = l_7 = l_8$ against the alternative that at least one root was different. For this hypothesis,

$c^2 = -(49)[\ln(3.60) + \ln(1.63) + ... + \ln(0.11)] + (49)(8)\ln$[(3.60+1.63+…+0.11)/8]=233.77

with 35 degrees of freedom, which is highly significant. A test of the next hypothesis $H_0 : l_2 = l_3 = l_4 = l_5 = l_6 = l_7 = l_8$ which says, “given the l_1 is different from the others, are the others equal? ”, produces a value of $c^2 = 136.99$. This implies, with 27 degrees of freedom, it is also quite significant at 5% level. Then repeat the process until testing the hypothesis $H_0 : l_6 = l_7 = l_8$ with a value of $c^2 = 11.0705$ with 5 degrees of freedom. It is not significant at 5% level. Thus, the last 3 principal components would be deleted and the first 5 principal components account for 92.94% of the total sample variance.

Based on the visual inspection of the plot of the sample influence functions (Figure 1), the potential outliers with the associated values of the sample influence functions for each component is summarized as follows.

Table 1. *Outliers with the associated values of the sample influence functions*

	No. of the outliers	Values of the sample influence functions, respectively
1st principal component	1, 10, 18, 24, 40	10.98, 7.42, 14.58, 12.65, 10.23
2nd principal component	2, 5, 43	7.35, 6.85, -1.18
3rd principal component	2, 5	37.61, 10.43
4th principal component	11	8.16
5th principal component	20, 43	1.85, 2.93

Next, by examining the perspective plot of $\tilde{I}_{ikj}(l) \equiv (n-1)^2\{\hat{l}_j - \hat{l}_{(i)j} - \hat{l}_{(k)j} + \hat{l}_{(i,k)j}\}$ for individual principal components (Figure 2), the potential masked observations with their associated values of the second order influence function are given as follows:

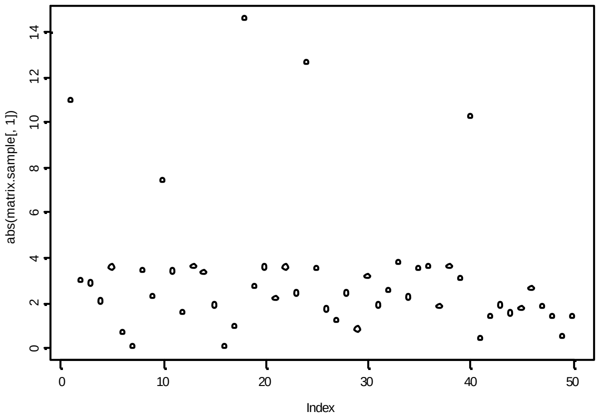
Table 2. *Masked outliers with the associated values of the sample influence functions*

	Pairs of the masked observations	Values of the sample influence functions, respectively
1st principal component	None	None
2nd principal component	(2,5) (2,32)	-618.08, -275.09
3rd principal component	(2,5) (2,11)	543.90, -385.57
4th principal component	(2,11)	386.12
5th principal component	(2,28)	-178.94

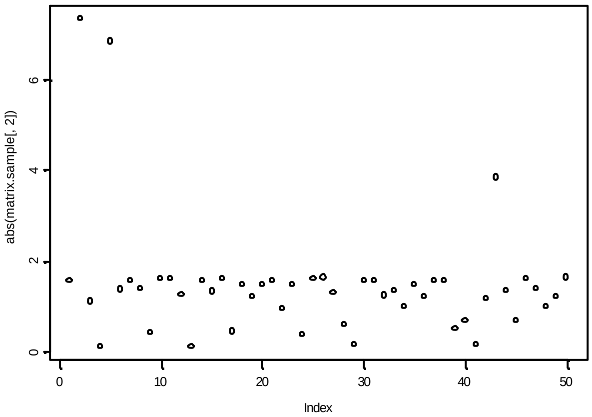
We also want to know whether the statistical inference will be different or not after deleting the masked observations. For the 5th principal component, we delete the pair of observations (2,28) and apply the Anderson's procedure to decide how many principal components we should retain. We find that only 4 principal components will be retained. The first 4 principal components account for 78.24% of the total sample variance. The conclusions based on the deletion of the pair of observations are significantly different from the original analyses. As we choose to retain the first 4 principal components for interpretation, it might provide us with new information on the analysis. From the above, we know that the diagnostics based on the interaction influence function indeed provide additional informative diagnostic information. As we choose to retain the first 4 principal components for interpretation, it might provide us with new information on the analysis.

Figure 1. *Plot of the1st order sample influence functions*

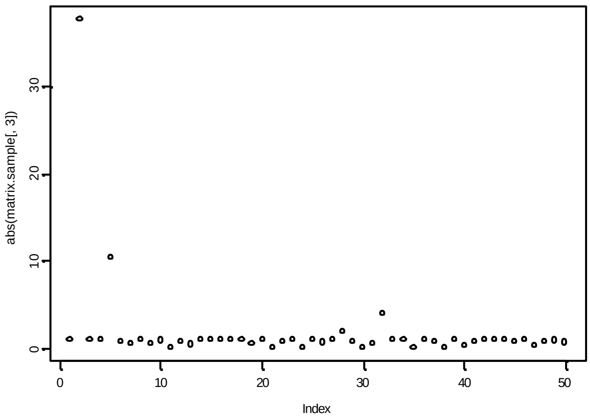
The sample influence functions for the 1st pc



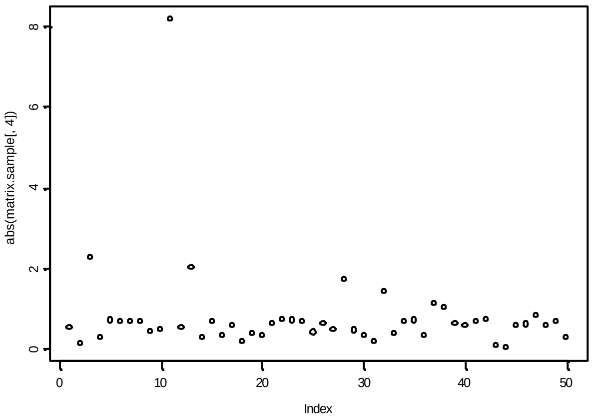
The sample influence functions for the 2nd pc



The sample influence functions for the 3rd pc



The sample influence functions for the 4th pc



The sample influence functions for the 5th pc

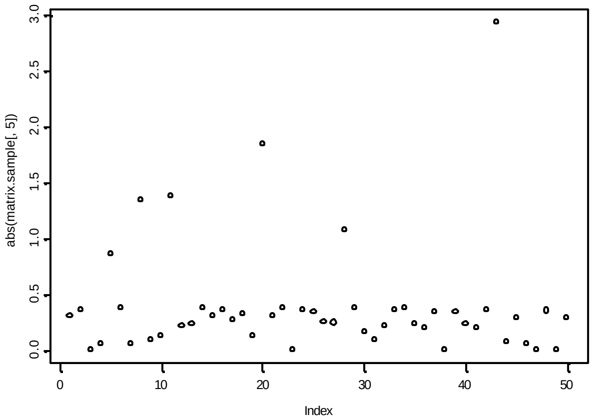
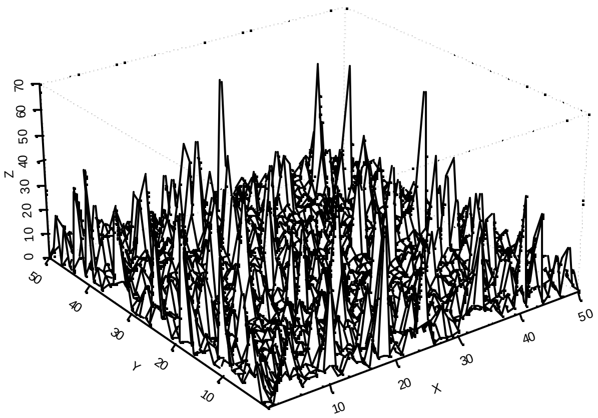
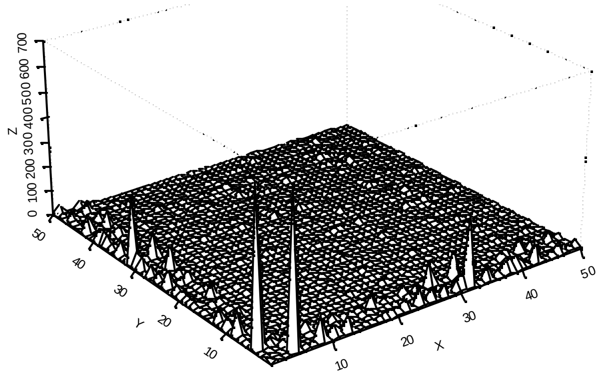


Figure 2. *Plot of the2nd order sample influence functions*

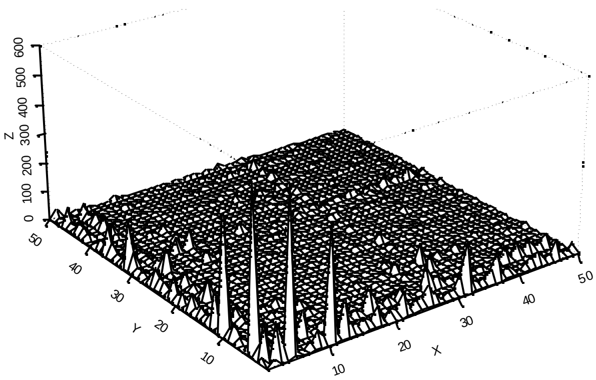
The perspective plot of the 1st pc



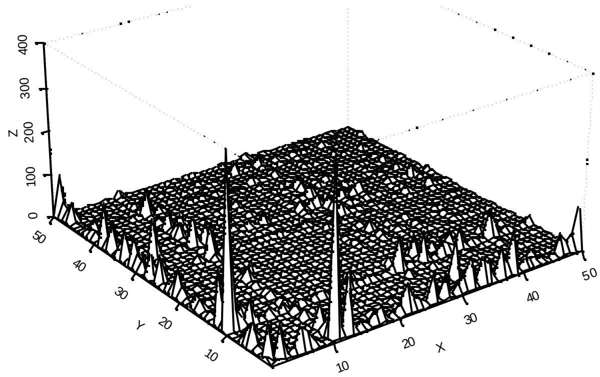
The perspective plot of the 2nd pc



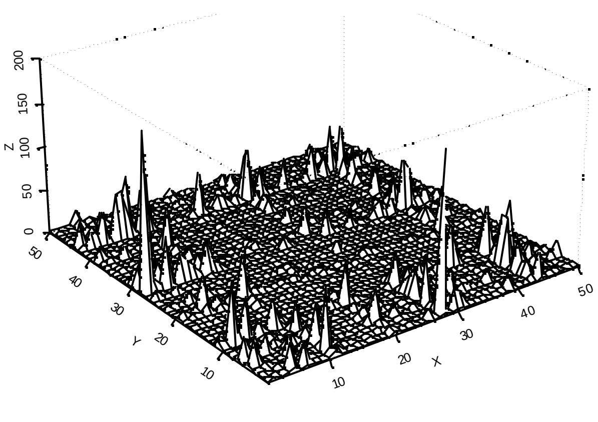
The perspective plot of the 3rd pc



The perspective plot of the 4th pc



The perspective plot of the 5th pc



5. Discussions

Detecting outlying observations is crucial in statistical analyses. To identify the masked observation, the second order influence function is proposed as diagnostic tool. In addition, a robust method is proposed by downweighting the outliers identified by the sample influence function.

In this thesis, the influential observations are judged by eye balls. The objective warning limit might be needed. However, since the influential observations come from unknown distributions, it might be very difficult to obtain an objective benchmark. One possible resolution is based on intensive simulation study.

Anderson's statistic is used for choosing the number of the principal components retained. However, the conclusions obtained by Anderson's method are frequently different from the ones based on scree plot or 90% of the total sample variance. Therefore, to determine whether some observations are influential, it might be sensible that the eigenvectors of the sample covariance matrix obtained by deleting these observations need to be examined, not only Anderson's statistic.

The robust method has been implemented in Splus function and given in Appendix. To explore the effectiveness of the robust method, more simulations need to be done in the future. Although the diagnostics and the robust methods proposed in principal component, it can also be developed for the other methods in multivariate analysis, for example, discriminant analysis and factor analysis.

References

Anderson, T. W. (1963). Asymptotic theory for principal component analysis. *Ann. Math. Statist. 34, 122-48.*

Brooks, S. P. (1994), Diagnostics for principal components: influence functions as diagnostic tools. *The Statistician. 43, No. 4, 483-94.*

Critchley, F. (1985). Influence in principal component analysis. *Biometrika 72, 627-36.*

Croux & Haesbroeck (1999). Principal component analysis based on robust estimators of the Covariance or correlation matrix: influence function and efficiencies. *Soumis pour publication.*

Johnson, R. A. & Wichern, D.W. (1982). Applied Multivariate Statistical Analysis, 5^{th} ed. *Prentice-Hall.*

Jolliffe (1986). Principal component analysis. *Springer-Verlag.*

Radhakrishnan & Kshirsagar (1981). Influence functions for certain parameters in multivariate analysis. *Commun. Statist. B 10, 515-29.*

Shi, L. (1997). Local influence in principal component analysis. *Biometrika 84, 175-86.*

Sibson, R. (1979). Studies in the robustness of multidimensional scaling: perturbational analysis of classical scaling. *J. R. Statist. Soc. B 41, 217-29.*

Tanaka, Y. (1988). Sensitivity analysis in principal component analysis: influence on the subspace spanned by principal component. *Commun. Statist. A 17, 3157-75.*

Appendix 1.

Proof of theorem 1:

Equating coefficients of $e_1, e_2, \frac{1}{2}e_1{}^2, \frac{1}{2}e_2{}^2, \frac{1}{2}e_1e_2$ in $B(e_1,e_2)e(e_1,e_2) = l(e_1,e_2)e(e_1,e_2)$ gives

$$Bf_1 + C_1e = lf_1 + em_1 \qquad (1)$$

$$Bf_2 + C_2e = lf_2 + em_2 \qquad (2)$$

$$Bg_1 + 2C_1f_1 + D_1e = lg_1 + 2m_1f_1 + v_1e \qquad (3)$$

$$Bg_2 + 2C_2f_2 + D_2e = lg_2 + 2m_2f_2 + v_2e \qquad (4)$$

$$Bg_{12} + 2C_1f_2 + 2C_2f_1 + D_{12}e = lg_{12} + 2m_1f_2 + 2m_2f_1 + v_{12}e \qquad (5)$$

and equating coefficients of e_1 and e_2 in $e(e_1,e_2)^T e(e_1,e_2) = 1$ gives

$$e^T f_1 = 0 \quad \text{and} \quad e^T f_2 = 0$$

Premultiplying (1) by e^T and using $Be = le, e^Te = 1, e^Tf_1 = 0$ gives

$$m_1 = e^TC_1e$$

Premultiplying (2) by e^T and using $Be = le, e^Te = 1, e^Tf_2 = 0$ gives

$$m_2 = e^TC_2e$$

Now using (1) $\Rightarrow Bf_1 - lf_1 = m_1e - C_1e$

$$(B - lI)f_1 = m_1e - C_1e$$

$$(B - lI)^+(B - lI)f_1 = m_1(B - lI)^+e - (B - lI)^+C_1e$$

$\therefore f_1 = -(B - lI)^+C_1e$

Similarly, $f_2 = -(B - lI)^+C_2e$

Premultiplying (3) by e^T gives

$e^TBg_1 + 2e^TC_1f_1 + e^TD_1e = le^Tg_1 + 2m_1e^Tf_1 + e^Tv_1e$

$e^Tv_1e = 2e^TC_1f_1 + e^TD_1e - 2m_1e^Tf_1$

So, $v_1 = e^T(2ee^TC_1f_1e^T + ee^TD_1ee^T)e$

$= e^T(2C_1f_1e^T + D_1)e$

$= e^T(D_1 - 2C_1(B - lI)^+C_1ee^T)e$

$= e^T(D_1 - 2C_1(B - lI)^+C_1)e$

$\therefore v_1 = e^T(D_1 - 2C_1(B - lI)^+C_1)e$

Similarly, $v_2 = e^T(D_2 - 2C_2(B - lI)^+C_2)e$

Using (3) $\Rightarrow$

$(B - lI)g_1 = 2m_1f_1 + v_1e - 2C_1f_1 - D_1e$

$= (2m_1I - 2C_1)f_1 + (v_1I - D_1)e$

$\therefore \quad g_1 = (B - lI)^+\left[(2m_1I - 2C_1)f_1 + (v_1I - D_1)e\right]$

Similarly, $g_2 = (B - lI)^+\left[(2m_2I - 2C_2)f_2 + (v_2I - D_2)e\right]$

Premultiplying (5) by e^T gives

$e^TBg_{12} + 2e^TC_1f_2 + 2e^TC_2f_1 + e^TD_{12}e = e^Tlg_{12} + 2e^Tm_1f_2 + 2e^Tm_2f_1 + e^Tv_{12}e$

$$\Theta\; e^T Bg_{12} = e^T\, l g_{12} \;\;,\;\; e^T m_1 f_2 = m_1 e^T f_2 = 0 \;\;,\;\; e^T m_2 f_1 = m_2 e^T f_1 = 0$$

$$\therefore e^T v_{12} e = 2e^T C_1 f_2 + 2e^T C_2 f_1 + e^T D_{12} e$$

$$\Rightarrow v_{12} = e^T (2ee^T C_1 f_2 e^T + 2ee^T C_2 f_1 e^T + ee^T D_{12} ee^T) e$$

$$= e^T (2C_1 f_2 e^T + 2C_2 f_1 e^T + D_{12}) e$$

$$= e^T (D_{12} - 2(B - lI)^+ (C_1 C_2 + C_2 C_1)) e$$

$$\therefore \quad v_{12} = e^T (D_{12} - 2(B - lI)^+ (C_1 C_2 + C_2 C_1)) e$$

Using (5), we get

$$(B - lI) g_{12} = 2m_1 f_2 + 2m_2 f_1 + v_{12} e - 2C_1 f_2 - 2C_2 f_1 - D_{12} e$$

$$\Rightarrow \quad g_{12} = (B - lI)^+ [(2m_1 I - 2C_1) f_2 + (2m_2 I - 2C_2) f_1 + (v_{12} - D_{12}) e]$$

Appendix 2.

Derivation of the second order influence function:

(1) $m_{1j} = e_j^t C_1 e_j = \; e_j^t[(\sum a_{1j}e_j)(\sum a_{1j}e_j)^t - \sum l_j e_j e_j^t]e_j = \; a_{1j}^2 - l_j$

(2) $m_{2j} = e_j^t C_2 e_j = \; e_j^t[(\sum a_{2j}e_j)(\sum a_{2j}e_j)^t - \sum l_j e_j e_j^t]e_j = \; a_{2j}^2 - l_j$

(3)

$$f_{1j} = -(B - l_j I)^+ C_1 e_j = -(\Omega(F) - l_j I)^+ C_1 e_j$$

$$= -(\sum_{(j)}(l_k - l_j)^{-1} e_k e_k^t)[(\sum a_{1j}e_j)(\sum a_{1j}e_j)^t - \sum l_j e_j e_j^t]e_j$$

$$= -(\sum_{(j)}(l_k - l_k)^{-1} e_k e_k^t)[a_{1j}(\sum a_{1j}e_j) - l_j e_j]$$

$$= -a_{1j} \sum_{(j)} a_{1k}(l_k - l_j)^{-1} e_k$$

where $\sum_{(j)}$ denotes summation over all terms except j.

(4) $f_{2j} = -(B - l_j I)^+ C_2 e_j = -(\Omega(F) - l_j I)^+ C_2 e_j$

$= -(\sum_{(j)}(l_k - l_j)^{-1} e_k e_k^t)[(\sum a_{2j}e_j)(\sum a_{2j}e_j)^t - \sum l_j e_j e_j^t]e_j$

$= -(\sum_{(j)}(l_k - l_k)^{-1} e_k e_k^t)[a_{2j}(\sum a_{2j}e_j) - l_j e_j]$

$= -a_{2j} \sum_{(j)} a_{2k}(l_k - l_j)^{-1} e_k$

The second-order terms are given by,

(5) $v_{1j} = e_j^t D_1 e_j - 2e_j^t C_1 (B - l_j I)^+ C_1 e_j$

$= -2e_j^t(\sum a_{1j}e_j)(\sum a_{1j}e_j)^t e_j - 2e_j^t[(\sum a_{1j}e_j)(\sum a_{1j}e_j)^t - \sum l_j e_j e_j^t]a_{1j}\sum_{(j)} a_{1k}(l_k - l_j)^{-1} e_k$

$= -2a_{1j}^2 - 2a_{1j}^2(\sum a_{1j}e_j)^t \sum_{(j)} a_{1k}(l_k - l_j)^{-1} e_k + 2 l_j e_j^t a_{1j} \sum_{(j)} a_{1k}(l_k - l_j)^{-1} e_k$

$= -2a_{1j}^2 - 2a_{1j}^2 \sum_{(j)} a_{1k}^2 (l_k - l_j)^{-1}$

$= -2a_{1j}^2\{1 + \sum_{(j)} a_{1k}^2 (l_k - l_j)^{-1}\}$

$$(6)\, v_{2j} = e_j^t D_2 e_j - 2e_j^t C_2 (B - l_j I)^+ C_2 e_j$$

$$= \; -2e_j^t (\sum a_{2j} e_j)(\sum a_{2j} e_j)^t e_j - 2e_j^t [(\sum a_{2j} e_j)(\sum a_{2j} e_j)^t - \sum l_j e_j e_j^t] a_{2j} \sum_{(j)} a_{2k} (l_k - l_j)^{-1} e_k$$

$$= -2a_{2j}^2 - 2a_{2j}^2 (\sum a_{2j} e_j)^t \sum_{(j)} a_{2k} (l_k - l_j)^{-1} e_k + 2 l_j e_j^t a_{2j} \sum_{(j)} a_{2k} (l_k - l_j)^{-1} e_k$$

$$= -2a_{2j}^2 - 2a_{2j}^2 \sum_{(j)} a_{2k}^2 (l_k - l_j)^{-1}$$

$$= -2a_{2j}^2 \{1 + \sum_{(j)} a_{2k}^2 (l_k - l_j)^{-1}\}$$

$$(7)\, v_{12,j} = e_j^t D_{12} e_j - 2e_j^t (B - l_j I)^+ (C_1 C_2 + C_2 C_1) e_j$$

$$= -4e_j^t (\sum a_{1j} e_j)(\sum a_{2j} e_j)^t e_j - 2e_j^t (\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t) C_1 C_2 e_j - 2e_j^t (\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t) C_2 C_1 e_j$$

$$= -4a_{1j} a_{2j}$$

$$(8)\, g_{1j} = (B - l_j I)^+ [(2m_{1j} I - 2C_1) f_{1j} + (v_{1j} I - D_1) e_j]$$

$$= 2(\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t)[(a_{1j}^2 - l_j) I - (\sum a_{1j} e_j)(\sum a_{1j} e_j)^t + \sum l_j e_j e_j^t][-a_{1j} \sum_{(j)} a_{1k} (l_k - l_j)^{-1} e_k]$$

$$+ (\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t)[-2a_{1j}^2 I - 2a_{1j}^2 \sum_{(j)} a_{1k}^2 (l_k - l_j)^{-1} I + 2(\sum a_{1j} e_j)(\sum a_{1j} e_j)^t] e_j$$

$$= -2\sum_{(j)} a_{1k}^2 (l_k - l_j)^{-1} f_{1j} - 2a_{1j}^3 \sum_{(j)} a_{1k} (l_k - l_j)^{-2} e_k$$

$$(9)\, g_{2j} = (B - l_j I)^+ [(2m_{2j} I - 2C_2) f_{2j} + (v_{2j} I - D_2) e_j]$$

$$= 2(\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t)[(a_{2j}^2 - l_j) I - (\sum a_{2j} e_j)(\sum a_{2j} e_j)^t + \sum l_j e_j e_j^t][-a_{2j} \sum_{(j)} a_{2k} (l_k - l_j)^{-1} e_k]$$

$$+ (\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t)[-2a_{2j}^2 I - 2a_{2j}^2 \sum_{(j)} a_{2k}^2 (l_k - l_j)^{-1} I + 2(\sum a_{2j} e_j)(\sum a_{2j} e_j)^t] e_j$$

$$= -2\sum_{(j)} a_{2k}^2 (l_k - l_j)^{-1} f_{2j} - 2a_{2j}^3 \sum_{(j)} a_{2k} (l_k - l_j)^{-2} e_k$$

$$(10)\, g_{12,j} = (B - l_j I)^+ [(2m_{1j} I - 2C_1) f_{2j} + (2m_{2j} I - 2C_2) f_{1j} + (v_{12,j} - D_{12}) e]$$

$$= 2[\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t][(a_{1j}^2 - l_j) I - (\sum a_{1j} e_j)(\sum a_{1j} e_j)^t + \sum l_j e_j e_j^t][-a_{2j} \sum_{(j)} a_{2k} (l_k - l_j)^{-1} e_k]$$

$$+ 2[\sum_{(j)} (l_k - l_j)^{-1} e_k e_k^t][(a_{2j}^2 - l_j) I - (\sum a_{2j} e_j)(\sum a_{2j} e_j)^t + \sum l_j e_j e_j^t][-a_{1j} \sum_{(j)} a_{1k} (l_k - l_j)^{-1} e_k]$$

$$-4[\Sigma_{(j)}(l_k - l_j)^{-1} e_k e_k^t][a_{1j}a_{2j}I - (\Sigma a_{1j}e_j)(\Sigma a_{2j}e_j)^t]e_j$$

$$= -2a_{2j}(a_{1j}^2 - l_j)\Sigma_{(j)}a_{2k}(l_k - l_j)^{-2}e_k - 2\Sigma_{(j)}a_{1k}(l_k - l_j)^{-1}e_k[-a_{2j}\Sigma_{(j)}a_{1k}a_{2k}(l_k - l_j)^{-1}]$$

$$+2\Sigma_{(j)}l_k(l_k - l_j)^{-1}e_k e_k^t[-a_{2j}\Sigma_{(j)}a_{2k}(l_k - l_j)^{-1}e_k]$$

$$-2a_{1j}(a_{2j}^2 - l_j)\Sigma_{(j)}a_{1k}(l_k - l_j)^{-2}e_k - 2\Sigma_{(j)}a_{2k}(l_k - l_j)^{-1}e_k[-a_{1j}\Sigma_{(j)}a_{1k}a_{2k}(l_k - l_j)^{-1}]$$

$$+2\Sigma_{(j)}l_k(l_k - l_j)^{-1}e_k e_k^t[-a_{1j}\Sigma_{(j)}a_{1k}(l_k - l_j)^{-1}e_k]$$

$$-4a_{2j}\Sigma_{(j)}a_{1k}(l_k - l_j)^{-1}e_k$$

$$= -2a_{1j}^2a_{2j}\Sigma_{(j)}a_{2k}(l_k - l_j)^{-2}e_k + 2a_{2j}l_j\Sigma_{(j)}a_{2k}(l_k - l_j)^{-2}e_k \,--$$

$$-2a_{1j}a_{2j}^2\Sigma_{(j)}a_{1k}(l_k - l_j)^{-2}e_k + 2a_{1j}l_j\Sigma_{(j)}a_{1k}(l_k - l_j)^{-2}e_k$$

$$-2\Sigma_{(j)}a_{1k}a_{2k}(l_k - l_j)^{-1}[\Sigma_{(j)}(a_{1k} + a_{2k})(l_k - l_j)^{-1}]e_k$$

$$-2a_{2j}\Sigma_{(j)}a_{2k}l_k(l_k - l_j)^{-2}e_k - 2a_{1j}\Sigma_{(j)}a_{1k}l_k(l_k - l_j)^{-2}e_k$$

$$-4a_{2j}\Sigma_{(j)}a_{1k}(l_k - l_j)^{-1}e_k$$

$$= -2a_{2j}[\Sigma_{(j)}a_{2k}(l_k - l_j)^{-2}(m_{1j} + l_k)]e_k - 2a_{1j}[\Sigma_{(j)}a_{1k}(l_k - l_j)^{-2}(m_{2j} + l_k)]e_k$$

$$-2\Sigma_{(j)}a_{1k}a_{2k}(l_k - l_j)^{-1}[\Sigma_{(j)}(a_{1k} + a_{2k})(l_k - l_j)^{-1}]e_k - 4a_{2j}\Sigma_{(j)}a_{1k}(l_k - l_j)^{-1}e_k$$

Appendix 3.

State.x77 in Splus

	Population	Income	Illiteracy	Life Exp	Murder	HS Grad	Frost	Area
Alabama	3615	3624	2.1	69.05	15.1	41.3	20	50708
Alaska	365	6315	1.5	69.31	11.3	66.7	152	566432
Arizona	2212	4530	1.8	70.55	7.8	58.1	15	113417
Arkansas	2110	3378	1.9	70.66	10.1	39.9	65	51945
California	21198	5114	1.1	71.71	10.3	62.6	20	156361
Colorado	2541	4884	0.7	72.06	6.8	63.9	166	103766
Connecticut	3100	5348	1.1	72.48	3.1	56.0	139	4862
Delaware	579	4809	0.9	70.06	6.2	54.6	103	1982
Florida	8277	4815	1.3	70.66	10.7	52.6	11	54090
Georgia	4931	4091	2.0	68.54	13.9	40.6	60	58073
Hawaii	868	4963	1.9	73.60	6.2	61.9	0	6425
Idaho	813	4119	0.6	71.87	5.3	59.5	126	82677
Illinois	11197	5107	0.9	70.14	10.3	52.6	127	55748
Indiana	5313	4458	0.7	70.88	7.1	52.9	122	36097
Iowa	2861	4628	0.5	72.56	2.3	59.0	140	55941
Kansa	2280	4669	0.6	72.58	4.5	59.9	114	81787
Kentucky	3387	3712	1.6	70.10	10.6	38.5	95	39650
Louisiana	3806	3545	2.8	68.76	13.2	42.2	12	44930
Maine	1058	3694	0.7	70.39	2.7	54.7	161	30920
Maryland	4122	5299	0.9	70.22	8.5	52.3	101	9891
Massachusetts	5814	4755	1.1	71.83	3.3	58.5	103	7826
Michigan	9111	4751	0.9	70.63	11.1	52.8	125	56817
Minnesota	3921	4675	0.6	72.96	2.3	57.6	160	79289
Mississippi	2341	3098	2.4	68.09	12.5	41.0	50	47296
Missouri	4767	4254	0.8	70.69	9.3	48.8	108	68995
Montana	746	4347	0.6	70.56	5.0	59.2	155	145587
Nebraska	1544	4508	0.6	72.60	2.9	59.3	139	76483
Nevada	590	5149	0.5	69.03	11.5	65.2	188	109889
New Hampshire	812	4281	0.7	71.23	3.3	57.6	174	9027
New Jersey	7333	5237	1.1	70.93	5.2	52.5	115	7521
New Mexico	1144	3601	2.2	70.32	9.7	55.2	120	121412
New York	18076	4903	1.4	70.55	10.9	52.7	82	47831
	Population	Income	Illiteracy	Life Exp	Murder	HS Grad	Frost	Area

North Carolina	5441	3875	1.8	69.21	11.1	38.5	80	48798
North Dakota	637	5087	0.8	72.78	1.4	50.3	186	69273
Ohio	10735	4561	0.8	70.82	7.4	53.2	124	40975
Oklahoma	2715	3983	1.1	71.42	6.4	51.6	82	68782
Oregon	2284	4660	0.6	72.13	4.2	60.0	44	96184
Pennsylvania	11860	4449	1.0	70.43	6.1	50.2	126	44966
Rhode Island	931	4558	1.3	71.90	2.4	46.4	127	1049
South Carolina	2816	3635	2.3	67.96	11.6	37.8	65	30225
South Dakota	681	4167	0.5	72.08	1.7	53.3	172	75955
Tennessee	4173	3821	1.7	70.11	11.0	41.8	70	41328
Texas	12237	4188	2.2	70.90	12.2	47.4	35	262134
Utah	1203	4022	0.6	72.90	4.5	67.3	137	82096
Vermont	472	3907	0.6	71.64	5.5	57.1	168	9267
Virginia	4981	4701	1.4	70.08	9.5	47.8	85	39780
Washington	3559	4864	0.6	71.72	4.3	63.5	32	66570
West Virginia	1799	3617	1.4	69.48	6.7	41.6	100	24070
Wisconsin	4589	4468	0.7	72.48	3.0	54.5	149	54464
Wyoming	376	4566	0.6	70.29	6.9	62.9	173	97203

Appendix 4.

Program Codes

(1) Robustness in Section 3

```
##### analyses of the original data set #####
ori.cov<-matrix(c(1,0.692,0.318,0.037,0.382,0.692,1,0.333,0.295,0.638,
0.318,0.333,1,0.029,0.302,0.037,0.295,0.029,1,0.358,0.382,0.638,0.302,0.358,1),ncol=5)

oridataeigen<-eigen(ori.cov)
print(oridataeigen)

##### generate randomly a data set and add an outlier ######
data<-rmvnorm(50, mean=rep(0,5),cov=ori.cov)

add<-rmvnorm(1, mean=rep(5,5),cov=ori.cov)

data<-matrix(c(data,add),ncol=5,byrow=T)

data.prc<-princomp(data,cor=T)
summary(data.prc,loadings=T)
cor(data)
dataeigen<-eigen(cor(data))
print(dataeigen)

##### the first order empirical influence function #####
column.mean<-apply(data,2,mean)
meanmatrix<-matrix(rep(column.mean,rep(51,5)),51,5)
matrix.1<-data-meanmatrix
matrix.2<-dataeigen$vectors
square.matrix.12<-(matrix.1%*%matrix.2)^2
matrix.eigenvalue<-matrix(rep(dataeigen$values,rep(51,5)),ncol=5,nrow=51)
print(matrix.eigenvalue)
matrix.empirical<-square.matrix.12-matrix.eigenvalue
print(matrix.empirical)

##### the first order sample influence function #####
matrix.pi.column1<-(-2)*square.matrix.12[,1]-2*square.matrix.12[,1]*
(square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,1])^(-1)+
```

```
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,1])^(-1))
print(matrix.pi.column1)

matrix.pi.column2<-(-2)*square.matrix.12[,2]-2*square.matrix.12[,2]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,2])^(-1))
print(matrix.pi.column2)

matrix.pi.column3<-(-2)*square.matrix.12[,3]-2*square.matrix.12[,3]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,3])^(-1))
print(matrix.pi.column3)

matrix.pi.column4<-(-2)*square.matrix.12[,4]-2*square.matrix.12[,4]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,4])^(-1))
print(matrix.pi.column4)

matrix.pi.column5<-(-2)*square.matrix.12[,5]-2*square.matrix.12[,5]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,5])^(-1))
print(matrix.pi.column5)

matrix.pi<-matrix(c(matrix.pi.column1,matrix.pi.column2,matrix.pi.column3,
matrix.pi.column4,matrix.pi.column5),nrow=51)
print(matrix.pi)

matrix.deleted<-matrix.empirical-(1/50)*(matrix.pi-matrix.empirical)
```

```
print(matrix.deleted)
matrix.sample<-matrix.empirical-0.5*(1/50)*matrix.pi
print(matrix.sample)

##### standardize the sample influence function of the 1st PC #####
me<-median(matrix.sample[,1])
std<-mad(matrix.sample[,1])
standard<-(matrix.sample[,1]-me)/std
print(standard)
dim(standard)<-c(51,1)

##### using the box kernel function to downweight the influential observations and calculate
    the eigenvalues and eigenvectors #####
box.kernel<-rep(0,51)
dim(box.kernel)<-c(51,1)
for(i in 1:51){
    if(abs(standard[i,1])>1){box.kernel[i,1]<-0}
    else{box.kernel[i,1]<-1}
}
print(box.kernel)

box.kernel<-matrix(box.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*box.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
    for(j in 1:5){
    weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}
weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
    for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}
```

```
wcoreigen<-eigen(weight.cor)
print(wcoreigen)

##### using the triangular kernel function to downweight the influential observations and
      calculate the eigenvalues and eigenvectors #####
triangular.kernel<-rep(0,51)
dim(triangular.kernel)<-c(51,1)
for(i in 1:51){
      if(abs(standard[i,1])>1){triangular.kernel[i,1]<-0}
      else{triangular.kernel[i,1]<-1-abs(standard[i,1])}
}
print(triangular.kernel)

triangular.kernel<-matrix(triangular.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*triangular.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
      for(j in 1:5){
      weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}
weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
      for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}

wcoreigen<-eigen(weight.cor)
print(wcoreigen)



##### using the Parzen type of kernel function to downweight the influential observations
      and calculate the eigenvalues and eigenvectors #####
parzen.kernel<-rep(0,51)
```

```
dim(parzen.kernel)<-c(51,1)
for(i in 1:51){
    if(abs(standard[i,1])>3){parzen.kernel[i,1]<-0}
    else
    if(2<abs(standard[i,1])<=3){parzen.kernel[i,1]<-(-standard[i,1]^2+5*abs(standard[i,1])-6)
    /4}
    else{parzen.kernel[i,1]<-1-(3*standard[i,1]^2)/16}
}
print(parzen.kernel)

parzen.kernel<-matrix(parzen.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*parzen.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
    for(j in 1:5){
    weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}
weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
    for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}

wcoreigen<-eigen(weight.cor)
print(wcoreigen)

##### using the Gaussian kernel function to downweight the influential observations and
    calculate the eigenvalues and eigenvectors #####
gaussian.kernel<-rep(0,51)
dim(gaussian.kernel)<-c(51,1)
for(i in 1:51){
    gaussian.kernel[i,1]<-(1/sqrt(2*pi))*exp(-(standard[i,1]^2)/2)
}
print(gaussian.kernel)
```

```
gaussian.kernel<-matrix(gaussian.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*gaussian.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
    for(j in 1:5){
    weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}

weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
    for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}

wcoreigen<-eigen(weight.cor)
print(wcoreigen)

##### using the biweight kernel function to downweight the influential observations and
    calculate the eigenvalues and eigenvectors #####
biweight.kernel<-rep(0,51)
dim(biweight.kernel)<-c(51,1)
for(i in 1:51){
    if(abs(standard[i,1])<=1){biweight.kernel[i,1]<-(15/16)*(1-standard[i,1]^2)^2}
    else{biweight.kernel[i,1]<-0}
}
print(biweight.kernel)

biweight.kernel<-matrix(biweight.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*biweight.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
```

```
    for(j in 1:5){
    weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}
weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
    for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}

wcoreigen<-eigen(weight.cor)
print(wcoreigen)

##### using the Epanechnikov kernel function to downweight the influential observations and
    calculate the eigenvalues and eigenvectors #####
epa.kernel<-rep(0,51)
dim(epa.kernel)<-c(51,1)
for(i in 1:51){
    if(abs(standard[i,1])<=sqrt(5)){epa.kernel[i,1]<-(3*sqrt(5)/4)*(1-0.2*standard[i,1]^2)}
    else{epa.kernel[i,1]<-0}
}
print(epa.kernel)

epa.kernel<-matrix(epa.kernel,nrow=51,ncol=5,byrow=F)
weimatrix.1<-matrix.1*epa.kernel

weight.cov<-rep(0,25)
dim(weight.cov)<-c(5,5)
for(i in 1:5){
    for(j in 1:5){
    weight.cov[i,j]<-(1/50)*sum(weimatrix.1[,i]*matrix.1[,j])
}
}
weight.cor<-rep(0,25)
dim(weight.cor)<-c(5,5)

for(i in 1:5){
```

```
    for(j in 1:5){weight.cor[i,j]<-weight.cov[i,j]/sqrt(weight.cov[i,i]*weight.cov[j,j])}
}

wcoreigen<-eigen(weight.cor)
print(wcoreigen)
```

(2) Example in Section 4

```
##### Standardize the original data and apply principal component to it and calculate
the eigenvalues and eigenvectors of the covariance matrix #####
state.or<-state.x77
column.mean<-apply(state.or,2,mean)
column.var<-apply(state.or,2,var)
meanmatrix<-matrix(rep(column.mean,rep(50,8)),50,8)
varmatrix<-matrix(rep(column.var,rep(50,8)),50,8)
state<-(state.or-meanmatrix)/sqrt(varmatrix)
princomp(state)
state.prc<-princomp(state)
summary(state.prc,loadings=T)
var(state)
stateeigen<-eigen(var(state))

##### The first order empirical influence function #####
column.mean.2<-apply(state,2,mean)
meanmatrix.2<-matrix(rep(column.mean.2,rep(50,8)),50,8)
matrix.1<-state-meanmatrix.2
matrix.2<-stateeigen$vectors
square.matrix.12<-(matrix.1%*%matrix.2)^2
matrix.eigenvalue<-matrix(rep(stateeigen$values,rep(50,8)),ncol=8,nrow=50)
print(matrix.eigenvalue)
matrix.empirical<-square.matrix.12-matrix.eigenvalue
print(matrix.empirical)

##### The first order sample influence function #####
matrix.pi.column1<-(-2)*square.matrix.12[,1]-2*square.matrix.12[,1]*
(square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,1])^(-1)+
```

```
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,1])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,1])^(-1))
print(matrix.pi.column1)
matrix.pi.column2<-(-2)*square.matrix.12[,2]-2*square.matrix.12[,2]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,2])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,2])^(-1))
print(matrix.pi.column2)

matrix.pi.column3<-(-2)*square.matrix.12[,3]-2*square.matrix.12[,3]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,3])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,3])^(-1))
print(matrix.pi.column3)

matrix.pi.column4<-(-2)*square.matrix.12[,4]-2*square.matrix.12[,4]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,4])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,4])^(-1))
print(matrix.pi.column4)

matrix.pi.column5<-(-2)*square.matrix.12[,5]-2*square.matrix.12[,5]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,5])^(-1)+
```

```
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,5])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,5])^(-1))
print(matrix.pi.column5)

matrix.pi.column6<-(-2)*square.matrix.12[,6]-2*square.matrix.12[,6]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,6])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,6])^(-1))
print(matrix.pi.column6)

matrix.pi.column7<-(-2)*square.matrix.12[,7]-2*square.matrix.12[,7]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,7])^(-1)+
square.matrix.12[,8]*(matrix.eigenvalue[1,8]-matrix.eigenvalue[1,7])^(-1))
print(matrix.pi.column7)

matrix.pi.column8<-(-2)*square.matrix.12[,8]-2*square.matrix.12[,8]*
(square.matrix.12[,1]*(matrix.eigenvalue[1,1]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,2]*(matrix.eigenvalue[1,2]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,3]*(matrix.eigenvalue[1,3]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,4]*(matrix.eigenvalue[1,4]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,5]*(matrix.eigenvalue[1,5]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,6]*(matrix.eigenvalue[1,6]-matrix.eigenvalue[1,8])^(-1)+
square.matrix.12[,7]*(matrix.eigenvalue[1,7]-matrix.eigenvalue[1,8])^(-1))
print(matrix.pi.column8)

matrix.pi<-matrix(c(matrix.pi.column1,matrix.pi.column2,matrix.pi.column3,
```

```
matrix.pi.column4,matrix.pi.column5,matrix.pi.column6,matrix.pi.column7,matrix.pi.column
    8),nrow=50)
print(matrix.pi)

matrix.deleted<-matrix.empirical-(1/49)*(matrix.pi-matrix.empirical)
print(matrix.deleted)
matrix.sample<-matrix.empirical-0.5*(1/49)*matrix.pi
print(matrix.sample)

##### The second order empirical influence function #####
empirical<-rep(0,50*50*8)
dim(empirical)<-c(50,50,8)
matrix.3<-matrix.1%*%matrix.2
for(j in 1:8)
{
    empirical[,,j]<-as.matrix(matrix.3[,j])%*%matrix.3[,j]
}

##### The second order sample influence function #####
sample<-rep(0,50*50*8)
dim(sample)<-c(50,50,8)
aa<-stateeigen$values
for(i in 1:50){
statei<-state[-i,]
pp.i<-eigen(var(statei))$values
    for(k  in  1:50){
    statek<-state[-k,]
    pp.k<-eigen(var(statek))$values
    stateik<-state[-c(i,k),]
    pp.i.k<-eigen(var(stateik))$values
    sample[i,k,]<-49*49*(aa-pp.i-pp.k+pp.i.k)
}
    }
```