# REDUCED RANK APPROXIMATION WITH WEIGHTS FROM COVARIANCE STRUCTURES

JAN DE LEEUW

ABSTRACT. We present theory and implementation of an algorithm for reduced rank or low-rank approximation of a rectangular data matrix, using row and column weight matrices that defined by covariance structures. The parameters of the weight matrices are estimated along with the reduced rank approximation by minimizing the log-likelihood loss function for the matrix variate normal.

## 1. FRAMEWORK

1.1. **Coding.** Supposed $\mathbb{F} = (\mathbb{D}|\mathbb{O})$ is a data-frame, with $n$ rows and $m+1$ columns. The $m$ columns in $\mathbb{D}$ are assume to be factors, with factor $j$ having $k_j$ levels. The single column $\mathbb{O}$ is a numerical variable. We think of $\mathbb{D}$ as the *design* and of $\mathbb{O}$ as the outcome. Typical variables in $\mathbb{D}$ are, for example, spatial locations, time points, replications, and indicators for variables.

We could observe, for example, ozone level, PM-10 level, temperature, and wind-speed for the 365 days of the year 2007, at 10 different observation stations. The data frame $\mathbb{F}$ will have $4 \times 365 \times 10 = 14,600$ rows and $3 + 1 = 4$ columns. The first three columns code for each observation where it was made, on which day, and what cross-sectional variable it measures. Note that $\mathbb{O}$ has the measurements on all four cross-sectional variables. Also note the design is balanced, unless there are missing data.

*Date*: Monday 22nd December, 2008 — 9h 42min — Typeset in LUCIDA BRIGHT.

The particular coding of the variables that we have chosen is not the only possible one. The 10 observation stations may be on a $4\times4$ rectangular latitude-longitude grid, which can be coded as two factors with four levels each, instead of one factor with ten levels. The 365 days may be coded as seven weekdays in 52 weeks, which again requires two factors instead of one. Clearly using such alternative codings in our example may make the design unbalanced. Not all weekdays will occur equally often in the year, and not all 16 grid points are occupied by observation stations.

There is a different, although isomorphic, way of coding the data, which may in some cases be more natural. We can use the design part of the frame to define an array $X$ of rank $m$ and dimension $k_1 \times \cdots \times k_m$, in which the measurements from $\mathbb{O}$ occupy the cells of the array. In a balanced design there is one observation in each cell, in an unbalanced design some cells will be empty. By using replications as a factor defining a dimension of the array we guarantee there is never more than one observation in each cell.

### 1.2. **Data Reduction.**

1.3. **Earlier Work.** In classical multivariate analysis the design has only two factors, the individuals and the cross-sectional variables. The array has rank two, in order words it is a matrix of observations by variables. It is usually assumed that individuals are replications, which means that the covariance matrix of rows is the identity, while the covariance matrix of the columns is modeled by some regression or structural equation model.

## 2. Specialization

In this paper we discuss a special case of the general framework in which we have two factors and a balanced design. Thus data can be collected in a matrix, and there are no missing values. We also

use a specific bilinear model to describe the means, and we use covariance structures to describe the row and column covariances. Thus one could think, for example, of one year of hourly ozone measurements at a single station, where the design has the factors days-of-the-year (365 levels) and hours-of-the-day (24 levels).

The data are an $n \times m$ matrix $X$. We want a reduced rank approximation to $X$, with weight matrices for both rows and columns. But, unlike in the classical case [Eckart and Young, 1936; Gabriel and Zamir, 1979; De Leeuw, 1984], the weight matrices are functions of unknown parameters, which have to be estimated along with the reduced rank approximation.

The loss function we use is the deviance[1] of a matrix variate normal distribution [Gupta and Nagar, 2000]

$$\mathcal{D}(\theta,\xi,Y) = m\log\mathbf{det}(\Sigma(\theta)) + n\log\mathbf{det}(\Omega(\xi)) + \\ + \mathbf{tr}\,\Sigma(\theta)^{-1}(X-Y)\Omega(\xi)^{-1}(X'-Y').$$

The constraints on the parameters are $\mathbf{rank}(Y) \leq p$, while $\Sigma \in \mathcal{S}(\theta)$ and $\Omega \in \mathcal{O}(\xi)$ are covariance structures [2].

## 3. Algorithm

To solve problems of this type, we use a *block relaxation method* [De Leeuw, 1994]. The algorithm is quite general, because it can also be applied to different types of constraints on $Y$, and to different parametric models for $\Sigma$ and $\Omega$.

[1] The *deviance* is minus two times the log-likelihood, except for irrelevant constants.

[2] A *covariance structure* $\mathcal{S}$ of order $n$ is the image of a function $\Sigma$ from an open subset of parameter space $\mathbb{R}^p$ to the cone of positive semi-definite matrices of order $n$.

In the unrestricted case, with no constraints on the structure of the means and covariances (and with a sufficient number of replications), the same algorithm has been proposed by Mardia and Goodall, Dutilleul, Lu and Zimmerman. It has been reported (Dutilleul) that in this case, which is of course inherently simpler than the one we consider here, the block relaxation algorithm (somewhat inelegantly described as the "flip-flop algorithm") outperforms general purpose optimization methods such as Newton-Raphson.

In our application the update algorithm from iteration $k$ to iteration $k+1$ is

$$\begin{aligned}
Y_{(k+1)} &= \underset{\mathbf{rank}(Y)\leq p}{\mathbf{argmin}}\ \mathcal{D}(\Sigma_{(k)},\Omega_{(k)},Y),\\
\Sigma_{(k+1)} &= \underset{\Sigma\in\mathcal{S}(\theta)}{\mathbf{argmin}}\ \mathcal{D}(\Sigma,\Omega_{(k)},Y_{(k+1)}),\\
\Omega_{(k+1)} &= \underset{\Omega\in\mathcal{O}(\xi)}{\mathbf{argmin}}\ \mathcal{D}(\Sigma_{(k+1)},\Omega,Y_{(k+1)}).
\end{aligned}$$

The corresponding function values are

$$\begin{aligned}
\mathcal{D}^1_{(k+1)} &= \min_{\mathbf{rank}(Y)\leq p}\mathcal{D}(\Sigma_{(k)},\Omega_{(k)},Y) = \mathcal{D}(\Sigma_{(k)},\Omega_{(k)},Y_{(k+1)}),\\
\mathcal{D}^2_{(k+1)} &= \min_{\Sigma\in\mathcal{S}(\theta)}\mathcal{D}(\Sigma,\Omega_{(k)},Y_{(k+1)}) = \mathcal{D}(\Sigma_{(k+1)},\Omega_{(k)},Y_{(k+1)}),\\
\mathcal{D}^3_{(k+1)} &= \min_{\Omega\in\mathcal{O}(\xi)}\mathcal{D}(\Sigma_{(k+1)},\Omega,Y_{(k+1)}) = \mathcal{D}(\Sigma_{(k+1)},\Omega_{(k+1)},Y_{(k+1)}).
\end{aligned}$$

Obviously

$$\cdots\leq\mathcal{D}^3_{(k+1)}\leq\mathcal{D}^2_{(k+1)}\leq\mathcal{D}^1_{(k+1)}\leq\mathcal{D}^3_{(k)}\leq\mathcal{D}^2_{(k)}\leq\mathcal{D}^1_{(k)}\leq\cdots$$

and if the sequence of function values is bounded from below it converges.

## 4. Existence of the MLE

Whether the sequence is bounded depends on both $X$ and $p$. In the case that $\mathbf{rank}(X)\leq p$, for example, the algorithm immediately gives $Y=X$ and the loss function can be made arbitrarily small by

letting $\Sigma$ and/or $\Omega$ approach any singular matrix. In a related example, we know from the fixed factor analysis literature [Anderson and Rubin, 1956] that fixing $\Sigma = I$ and letting $\Omega$ vary over all diagonal matrices always leads to $\inf_{Y,\Omega} \mathcal{D}(Y, \Omega) = -\infty$, no matter what $X$ and $p$ are.

Another example: suppose $\Sigma$ is an equi-correlation structure. Thus all diagonal elements are equal to $\sigma^2$ and all off-diagonal elements are equal to $\sigma^2\rho$. This means that $\Sigma$ has one eigenvalue equal to $\sigma^2(1 + (n-1)\rho$, corresponding to the normalized constant eigenvector with elements $n^{-\frac{1}{2}}$, and $n-1$ eigenvalues equal to $\sigma^2(1-\rho)$, corresponding to $n-1$ eigenvectors orthogonal to the constant one. Thus

$$\mathcal{D} = n \log \sigma^2 + \log(1 + (n-1)\rho) + (n-1)\log(1-\rho) + \\ + \frac{1}{\sigma^2}\left\{\frac{1}{1+(n-1)\rho}\alpha + \frac{1}{1-\rho}\beta\right\},$$

where $\alpha = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}$ and $\beta = \sum_{i=1}^{n} s_{ii} - \alpha$. If $\alpha = 0$ then we make $\mathcal{D}$ arbitrary close to $-\infty$ by letting $\rho \to -\frac{1}{n-1}$. And if $\beta = 0$ we have $\mathcal{D} \to -\infty$ if $\rho \to 1$.

The same thing happens if we require $\Sigma$ to be Toeplitz (also known in psychometrics as a simplex).

It is clear that these degenerate solutions happen if there is a vector in the null-space of $S$ which is also in the null-space of $\Sigma$. By construction, $S$ will always be singular.

Suppose that all matrices $\Sigma(\theta)$ commute pairwise. Then they have an eigenvector in common. Say this is $u$. Thus $\Sigma(\theta)u = \lambda(\theta)u$. Now define $Y = uu'X$. Then obviously $u'(X - Y) = 0$ and thus $u'(X-Y)\Omega^{-1}(X-Y)' = 0$, and thus we can let $\mathcal{D} \to -\infty$ if we let $\lambda(\theta) \to 0$.

We can prevent degeneracy from happening by making sure that $\Sigma$ is always positive definite.

## 5. SUBPROBLEMS

5.1. **Optimum Y.** We must find

$$Y^{(k+1)} = \underset{\mathbf{rank}(Y)\leq p}{\mathbf{argmin}}\ \mathbf{tr}\ \Sigma_{(k)}^{-1}(X-Y)\Omega_{(k)}^{-1}(X'-Y').$$

This is a regular weighted least squares reduced rank approximation problem, with Kronecker product structure for the weights. It can be solved by using the singular value decomposition.

Suppose $\Sigma_{(k)} = P_{(k)}'P_{(k)}$, where $P_{(k)}$ is upper-triangular. Note[3] that $\Sigma_{(k)}^{-1} = P_{(k)}^{-1}P_{(k)}^{-T}$. If $\Sigma_{(k)} \in \mathcal{A}_n$ then an explicit formula for the Cholesky factors $P_{(k)}$, and their inverses, is given in Appendix A. In the same way $\Omega_{(k)} = Q_{(k)}'Q_{(k)}$ and $\Omega_{(k)}^{-1} = Q_{(k)}^{-1}Q_{(k)}^{-T}$.

Define $Z_{(k)} = P_{(k)}^{-T}XQ_{(k)}^{-1}$. Then

$$\min_{\mathbf{rank}(Y)\leq p}\ \mathbf{tr}\ \Sigma_{(k)}^{-1}(X-Y)\Omega_{(k)}^{-1}(X'-Y') = \min_{\mathbf{rank}(U)\leq p}\ \mathbf{tr}\ (Z_{(k)}-U)(Z_{(k)}-U)'.$$

If

$$U_{(k)} = \underset{\mathbf{rank}(U)\leq p}{\mathbf{argmin}}\ \mathbf{tr}\ (Z_{(k)}-U)(Z_{(k)}-U)'$$

then we have

$$Y_{(k+1)} = P_{(k)}'U_{(k)}Q_{(k)}.$$

We find $U_{(k)}$ by the truncated singular value decomposition of $Z_{(k)}$.

5.2. **Optimum $\Sigma$.** Define the $n\times n$ matrix $S^{(k+1)} = \frac{1}{m}(X-Y_{(k+1)})\Omega_{(k)}^{-1}(X-Y_{(k+1)})'$, using the formula for the inverse in appendix A. Then

$$\underset{\Sigma\in\mathcal{A}_n(1)}{\mathbf{argmin}}\ \mathcal{D}(\Sigma,\Omega_{(k)},Y_{(k+1)}) = \underset{\Sigma\in\mathcal{A}_n(1)}{\mathbf{argmin}}\ \log\mathbf{det}(\Sigma) + \mathbf{tr}\ \Sigma^{-1}S_{(k+1)}.$$

This means fitting the AR(1) covariances to the "observed covariance matrix" $S_{(k+1)}$ by maximum likelihood. Appendix A shows that this can be done very simply and efficiently by finding the roots of a quadratic.

[3] $A^{-T}$ is the inverse of the transpose, or the transpose of the inverse, of $A$.

5.3. **Optimum $\Omega$.** Finding the optimal $\Omega$ is, of course, exactly analogous to finding the optimal $\Sigma$. Instead of the $n \times n$ matrix $S_{(k+1)}$ we define the $m \times m$ matrix $W_{(k+1)} = \frac{1}{n}(X - Y_{(k+1)})' \Sigma^{-1}_{(k+1)} (X - Y_{(k+1)})$. Otherwise the formulas are basically the same.

## 6. Generalizations

### 6.1. **More General Weights.**

6.1.1. *ARMA.* The R code in Appendix C.1 is written in such a way that it is easy to plug in different weight matrices. In fact, with a little bit of tweaking it is easy to replace the reduced rank approximation by other linear or non-linear models for the mean of the matrix variate normal.

As a first step, we could implement `AR(p)` or `ARMA(p,q)` covariance structures. This is, however, quite a bit more demanding, because the covariance matrices $\Sigma$ and $\Omega$ become considerably more complicated. Using existing time series code in R is complicated by the fact that this code is written for single series, not for covariance matrices. Also, it uses the general `optim()` routine to compute maximum likelihood estimates, which is convenient but computationally probably suboptimal.

We will look into using exact formulas for the `ARMA` covariance structure, such as the ones given in Van der Leeuw [1997], and we will look into developing special purpose algorithms to minimize the resulting loss functions.

6.1.2. *Cross Sectional.*

6.1.3. *Spatial.*

### 6.2. **Multidimensional Arrays.**

## REFERENCES

T.W. Anderson and H. Rubin. Statistical Inference in Factor Analysis. In J. Neyman, editor, *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 5*, pages 111–150. University of California Press, 1956.

J. De Leeuw. Fixed-rank Approximation with Singular Weight Matrices. *Computational Statistics Quarterly*, 1:3–12, 1984.

J. De Leeuw. Block relaxation algorithms in statistics. In H.-H. Bock, W. Lenski, and M. M. Richter, editors, *Information Systems and Data Analysis*, pages 308–324. Springer, 1994.

C. Eckart and G. Young. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika*, 1:211–218, 1936.

K.R. Gabriel and S. Zamir. Lower Rank Approximation of Matrices by Least Squares with Any Choice of Weights. *Technometrics*, 21: 489–498, 1979.

A.K. Gupta and D.K. Nagar. *Matrix Variate Distributions*. Chapman & Hall/CRC, Boca Raton, Florida, 2000.

L. Guttman. A Generalized Simplex for Factor Analysis. *Psychometrika*, 20:173–192, 1955.

B. Martos. *Nonlinear Programming Theory and Methods*. North-Holland Publishing Company, Amsterdam, Netherlands, 1975.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer, New York, N.Y., second edition, 2006.

J. Van der Leeuw. *Maximum Likelihood Estimation of Exact ARMA Models*. PhD thesis, Tilburg University, Tilburg, The Netherlands, 1997.

## APPENDIX A. PROPERTIES OF AR(1) COVARIANCE MATRICES

A weakly stationary AR(1) process[4] of length $n$ with autocorrelation $\phi$ and error variance $\sigma^2$ is generated by the recursion $\underline{y}_i = \phi\underline{y}_{i-1} + \underline{\epsilon}_i$ for $i = 2, \cdots, n$. Here $\phi^2 < 1$ and the $\underline{\epsilon}_i$ have mean zero and variance $\sigma^2$. Also the $\underline{\epsilon}_i$ are uncorrelated with each other and with $\underline{y}_1$. From stationarity means and variances must be the same for all $i$. Thus $\mathbf{E}(\underline{y}_1) = \mathbf{E}(\underline{y}_2) = \phi\mathbf{E}(\underline{y}_1)$ and $\mathbf{E}(\underline{y}_1) = 0$. In the same way $\mathbf{E}(\underline{y}_1^2) = \mathbf{E}(\underline{y}_2^2) = \phi^2\mathbf{E}(\underline{y}_1^2) + \sigma^2$ and thus $\mathbf{E}(\underline{y}_1^2) = \frac{\sigma^2}{1-\phi^2}$. It follows that the $\underline{y}_i$ have a covariance matrix $V$ with elements

$$v_{ij} = \frac{\sigma^2}{1-\phi^2}\phi^{|i-j|}.$$

A.1. **Inverse.** The inverse $V^{-1}$ exists if and only if $\sigma^2 > 0$ and $\phi^2 \neq 1$. It is tri-diagonal with elements

$$v^{ij} = \frac{1}{\sigma^2}\begin{cases} 1 & \text{if } i = j = 1 \text{ or } i = j = n, \\ -\phi & \text{if } |i-j| = 1, \\ 1+\phi^2 & \text{if } 1 < i = j < n. \end{cases}$$

A.2. **Cholesky.** If $V^{-1} = C'C$ is the Cholesky decomposition of $V^{-1}$, with $C$ upper-triangular, then

$$c_{ij} = \frac{1}{\sigma}\begin{cases} 1 & \text{if } 1 \leq i = j < n, \\ -\phi & \text{if } j - i = 1, \\ \sqrt{1-\phi^2} & \text{if } i = j = n. \end{cases}$$

[4]Our results are slightly different from those in the time series literature [Shumway and Stoffer, 2006, p. 125-127] because we deal with the endpoint differently. This is related to the fact that in time series analysis the finite observed series is embedded in an infinite series, and then statements are made about that infinite series. For this reason it makes more sense, perhaps, to call our covariances structure a *regular simplex* [Guttman, 1955] instead of an AR(1) covariance matrix.

It follows that $\mathbf{det}(C) = \frac{1}{\sigma^n}\sqrt{1-\phi^2}$, and thus $\mathbf{det}(V) = \frac{(\sigma^2)^n}{1-\phi^2}$. Note that $V$ is positive definite if and only if $\sigma^2 > 0$ and $\phi^2 < 1$.

We can also compute the Cholesky decomposition $V = D'D$, with $D$ upper triangular. This gives

$$d_{ij} = \frac{\sigma}{\sqrt{1-\phi^2}}\begin{cases}\phi^{j-i} & \text{if } i = 1,\\ \sqrt{1-\phi^2}\phi^{j-i} & \text{if } j \geq i > 1.\end{cases}$$

A.3. **Deviance.** If $S$ is any matrix of order $n$ then the deviance is

$$\mathcal{D}(\phi,\sigma^2) = \log\mathbf{det}(V)+\mathbf{tr}\,V^{-1}S = n\log\sigma^2-\log(1-\phi^2)+\frac{\alpha\phi^2-2\beta\phi+\gamma}{\sigma^2},$$

where

$$\begin{aligned}\alpha &= \sum_{i=2}^{n-1} s_{ii},\\ \beta &= \sum_{i=1}^{n-1} s_{i,i+1},\\ \gamma &= \sum_{i=1}^{n} s_{ii}.\end{aligned}$$

Note that if $S$ is positive semi-definite then $\gamma \geq \alpha \geq 0$. If $\phi^2 < 1$ and $S \neq 0$ we have $\alpha\phi^2 - 2\beta\phi + \gamma > 0$ and

$$\mathcal{D}(\phi,\star) \triangleq \min_{\sigma^2}\mathcal{D}(\phi,\sigma^2) = \log\frac{\alpha\phi^2-2\beta\phi+\gamma}{1-\phi^2} + n - n\log n,$$

with the minimum attained at

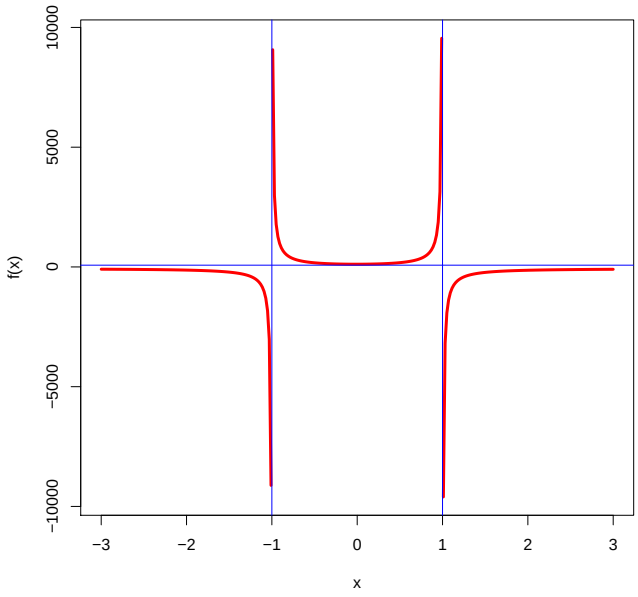
$$\sigma^2(\phi) = \frac{\alpha\phi^2-2\beta\phi+\gamma}{n}.$$

To find the maximum likelihood estimate of $\phi$ we must minimize

$$f(\phi) = \frac{\alpha\phi^2-2\beta\phi+\gamma}{1-\phi^2}.$$

A typical plot of $f$ is in Figure 1. It follows from the positivity of $f$ on the open interval $\mathcal{I} = (-1,+1)$, and the two vertical asymptotes, that $f$ has at least one minimum on $\mathcal{I}$. Actually, because $f$ is the ratio of a positive convex quadratic and a positive concave

FIGURE 1. Estimating Autocorrelation




function, it is explicitly quasi-convex on $\mathcal{I}$ and has a unique minimum [Martos, 1975, Theorem 51].

At this minimum the derivative must vanish. The derivative vanishes if and only if $\beta\phi^2 - (\alpha + \gamma)\phi + \beta = 0$. If $\beta = 0$ the only root is $\hat{\phi} = 0$. Otherwise the quadratic equation has two real roots, of which only one is in $\mathcal{I}$. Note that the product of the two roots of the quadratic is 1, which implies that $f$ has a local maximum at $1/\hat{\phi}$, which is outside $\mathcal{I}$. The root in $\mathcal{I}$ is the maximum likelihood estimate $\hat{\phi}$ of $\phi$, which can then be used to compute the maximum likelihood estimate $\hat{\sigma}^2 = \sigma^2(\hat{\phi})$.

In some cases we may want to consider the situation in which $\sigma^2$ is known and we only estimate $\phi$. Because $1 - \phi^2$ is concave, and the logarithm is an increasing concave function, it follows that $-\log(1 - \phi^2)$, and consequently $\mathcal{D}(\phi, \sigma^2)$, is convex in $\phi$ on $\mathcal{I}$. Again the vertical asymptotes guarantee a unique minimum, which

is attained at the unique root of the cubic $\sigma^2\phi+(\alpha\phi-\beta)(1-\phi^2)=0$ that lies in $\mathcal{I}$.

## Appendix B. Covariances with Linear Structure

Consider the problem of minimizing

$$\mathcal{D}(\theta)=\log\mathbf{det}(\Sigma(\theta))+\mathbf{tr}\,\Sigma(\theta)^{-1}S,$$

where

$$\Sigma(\theta)=\sum_{s=1}^r\theta_sA_s,$$

and the $A_s$ are known matrices.

We use the basic results

$$\begin{aligned}\log\mathbf{det}(\Sigma(\theta)+\epsilon A_s)&=\log\mathbf{det}(\Sigma(\theta))+\epsilon\,\mathbf{tr}\,\Sigma(\theta)^{-1}A_s+o(\epsilon),\\ \mathbf{tr}\,[\Sigma(\theta)+\epsilon A_s]^{-1}S&=\mathbf{tr}\,\Sigma(\theta)^{-1}S-\epsilon\,\mathbf{tr}\,\Sigma(\theta)^{-1}A_s\Sigma(\theta)^{-1}S+o(\epsilon).\end{aligned}$$

Thus the partials are

$$\partial_s\mathcal{D}(\theta)=\mathbf{tr}\,\Sigma(\theta)^{-1}A_s-\mathbf{tr}\,\Sigma(\theta)^{-1}A_s\Sigma(\theta)^{-1}S,$$

and

$$\begin{aligned}\partial^2_{st}\mathcal{D}(\theta)=-\mathbf{tr}\,\Sigma(\theta)^{-1}A_s\Sigma(\theta)^{-1}A_t+\\ +2\mathbf{tr}\,\Sigma(\theta)^{-1}A_t\Sigma(\theta)^{-1}A_s\Sigma(\theta)^{-1}S.\end{aligned}$$

These formulas are all that is needed to apply the Newton-Raphson method. We can apply Fisher Scoring by observing that if $\Sigma(\theta)\approx S$ then

$$\partial^2_{st}\mathcal{D}(\theta)\approx\mathbf{tr}\,\Sigma(\theta)^{-1}A_t\Sigma(\theta)^{-1}A_s,$$

which provides a convenient positive semi-definite approximation to the Hessian.

If the covariance structure is specified instead as

$$\Sigma(\xi)^{-1}=\sum_{s=1}^r\xi_sA_s,$$

then things simplify considerably. We find

$$\partial_s \mathcal{D}(\xi) = -\mathbf{tr}\ A_s \Sigma(\xi) + \mathbf{tr}\ A_s S,$$

and

$$\partial^2_{st} \mathcal{D}(\xi) = \mathbf{tr}\ \Sigma(\xi) A_s \Sigma(\xi) A_t,$$

which shows that $\mathcal{D}$ in this case is convex in $\xi$.

## Appendix C. Code

C.1. **Core Routines.** We give the code for the main function `wPCA()`, as well as for the subroutine `redRank()` that compute an unweighted reduced rank approximation, and the subroutine `ar_ml()` that uses the method of Appendix A to compute the maximum likelihood of the AR(1) parameters from a covariance matrix.

```
source("../../lincov/lincov.R")

wPCA<-function(x,gR,gC,mlFuncR=linCov,mlFuncC=linCov,
    ndim=2,oeps=1e-6,ieps=1e-6,otmax=100,itmax=100,
    overbose=TRUE,iverbose=TRUE){
n<-nrow(x); m<-ncol(x); pR<-dim(gR)[3]; pC<-dim(gC)[3]
thR<-rep(0,pR); thC<-rep(0,pC); thR[1]<-1; thC[1]<-1
sig<-gR[,,1]; ome<-gC[,,1]; ofunc<-Inf; itel<-1
repeat {
    schol<-chol(sig); ochol<-chol(ome)
    schiv<-solve(schol); ochiv<-solve(ochol)
    xx<-crossprod(schiv,x)%*%ochiv
    rr<-redRank(xx,ndim)
    a<-crossprod(schol,rr$a)
    b<-crossprod(ochol,rr$b)
    y<-tcrossprod(a,b)
    res<-x-y; rss<-rr$rss
    funL<-rss+m*log(det(sig))+n*log(det(ome))
    s<-res%*%solve(ome,t(res))/m
    vR<-mlFuncR(s,gR,thR,eps=ieps,itmax=itmax,verbose=
        iverbose)
    thR<-vR$th; sig<-vR$sig; funR<-m*(vR$f)+n*log(det(
        ome))
    w<-crossprod(res,solve(sig,res))/n
    vC<-mlFuncC(w,gC,thC,eps=ieps,itmax=itmax,verbose=
        iverbose)
```

```
22      thC<-vC$th; ome<-vC$sig; funC<-n*(vC$f)+m*log(det(
           sig))
23      if (overbose)
24          cat("Iteration ",formatC(itel,digits=4),
25              "lssloss ",formatC(funL,digits=6,width=10)
                   ,
26              "llrloss ",formatC(funR,digits=6,width=10)
                   ,
27              "llcloss ",formatC(funC,digits=6,width=10)
                   ,
28              "\n")
29      nfunc<-funC
30      if ((abs(ofunc - nfunc) < oeps) || (itel == otmax)
           ) break()
31      itel<-itel+1; ofunc<-nfunc
32      }
33  return(list(a=a,b=b,y=y,sig=sig,ome=ome,thC=thC,thR=
       thR,dev=nfunc))
34  }
35
36  redRank<-function(x,p=2){
37  s<-svd(x,nv=p,nu=p); u<-as.matrix(s$u); v<-as.matrix(s
       $v); d<-s$d[1:p]
38  return(list(a=u,b=t(d*t(v)),rss=sum(s$d[-(1:p)]^2)))
39  }
```

C.2. **Utilities.** Because the Cholesky factor, inverses, and determinants of an AR(1) covariance matrix are simple functions of the two parameters $\sigma^2$ and $\phi$ we can use special routines that bypass the usual R routines chol(),det() and solve(). This save a huge number of multiplications, and we have to worry less about pivoting and rounding errors. For greater efficiency, these AR utilities should be translated to C.

```
ar_cov<-function(sig,phi,n)
    return((sig/(1-phi^2))*phi^abs(outer(1:n,1:n,"-"))
       )

ar_log_det<-function(sig,phi,nr)
return(nr*log(sig)-log(1-phi^2))

ar_ml<-function(a) {
nr<-nrow(a); phi<-0
gam<-sum(diag(a))
alp<-gam-(a[1,1]+a[nr,nr])
bet<-sum(a[outer(1:nr,1:nr,function(x,y) x-y==1)])
dis<-sqrt((alp+gam)^2-bet^2)
if (!(bet == 0))
    phi<-((alp+gam)-dis)/(2*bet)
g<-(alp*phi^2)+gam-(2*bet*phi)
sig<-g/nr
dev<-(nr*log(g)+nr)-(log(1-phi^2)+nr*log(nr))
return(list(sig=sig,phi=phi,dev=dev))
}


ar_inv_mat<-function(x,sig,phi) {
n<-nrow(x); z<-x
z[1,]<-x[1,]-phi*x[2,]
for (i in 2:(n-1))
    z[i,]<-(1+phi^2)*x[i,]-phi*(x[i-1,]+x[i+1,])
z[n,]<-x[n,]-phi*x[n-1,]
return(z/sig)
}

ar_mat_inv<-function(x,sig,phi) {
m<-ncol(x); z<-x
z[,1]<-x[,1]-phi*x[,2]
```

```
for (i in 2:(m-1))
    z[,i]<-(1+phi^2)*x[,i]-phi*(x[,i-1]+x[,i+1])
z[,m]<-x[,m]-phi*x[,m-1]
return(z/sig)
}

ar_mat_chol_inv_up<-function(x,sig,phi) {
m<-ncol(x); z<-x
z[,1]<-x[,1]
for (i in 2:(m-1))
    z[,i]<-x[,i]-phi*x[,i-1]
z[,m]<-sqrt(1-phi^2)*x[,m]-phi*x[,m-1]
return(z/sqrt(sig))
}

ar_chol_inv_lw_mat<-function(x,sig,phi) {
n<-nrow(x); z<-x
z[1,]<-x[1,]
for (i in 2:(n-1))
    z[i,]<-x[i,]-phi*x[i-1,]
z[n,]<-sqrt(1-phi^2)*x[n,]-phi*x[n-1,]
return(z/sqrt(sig))
}

ar_chol_lw_mat<-function(x,sig,phi) {
n<-nrow(x)
return(crossprod(ar_chol(sig,phi,n),x))
}

ar_mat_chol_up<-function(x,sig,phi) {
m<-ncol(x)
return(x%*%ar_chol(sig,phi,m))
}
```

```
ar_chol<-function(sig,phi,n) {
a<-phi^abs(outer(1:n,1:n,"-"))
a[1,]<-a[1,]/sqrt(1-phi^2)
return(sqrt(sig)*rm_low_diag(a))
}

ar_inv_chol<-function(sig,phi,n) {
a<-diag(n); a[n,n]<-sqrt(1-phi^2)
a[cbind(1:(n-1),2:n)]<--phi
return(a/sqrt(sig))
}

rm_low_diag<-function(a) {
n<-nrow(a); nn<-1:n
return(a*ifelse(outer(nn,nn,"<="),1,0))
}
```

DEPARTMENT OF STATISTICS, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90095-1554

*E-mail address*, Jan de Leeuw: deleeuw@stat.ucla.edu

*URL*, Jan de Leeuw: http://gifi.stat.ucla.edu