A Computer Program for Tests of Equality of Covariance Matrices

by

Regina Becker and K. C. S. Pillai
Purdue University

Technical Report #85-8

Department of Statistics
Purdue University

April 1985

PURDUE UNIVERSITY DEPARTMENT OF STATISTICS

Enclosed is a technical report containing documentation and a listing of a computer program for performing the computations described in the paper, "A Computer Program for Tests of Equality of Covariance Matrices," by Regina Becker and K. C. S. Pillai. Dr. Pillai passed away on June 5, 1985. He was a distinguished scholar and a leader in the field of multivariate analysis.

The research and programming for this project was supported by the Purdue University Department of Statistics. Accordingly, this work is freely distributed to anyone who requests it.

If, however, you have funds available to support this type of research, we would be very grateful to receive a charitable contribution of $50.00 or more. The enclosed form can be used for this purpose. The money so collected will be used to start an annual memorial lecture on Multivariate Analysis and Its Applications here at Purdue.

Purdue University
Mathematical Sciences Building
West Lafayette, Indiana 47907

Contribution to the

Purdue Foundation*

Date ______________________________

Amount ____________________________

From ____

____

____

____

*Please make check payable to Purdue Foundation and put the Pillai Memorial Lecture Fund in the corner.

Mail to: Department of Statistics
Purdue University
West Lafayette, Indiana 47907

I. Introduction

This document is a supplement to the paper, "Percentage points of the largest characteristic root of the multivariate beta matrix", by K. C. S. Pillai and Bernhard N. Flury in Communications in Statistics, Theory and Methods (1984) 13(18), 2199-2237. A program is presented here which is an algorithm for the test of equality of the covariance matrices of two p-variate normal populations based on Roy's largest root test statistic, illustrated by a numerical example. The program runs on the Purdue University Computing Center's CDC 6000 system. Some minor modifications may be needed for other computers.

Seven subroutines from the IMSL package are needed to run the program, namely, 1) becovm, 2) linv2p, 3) ludecp, 4) vmulff, 5) vmulfs, 6) eigrs, and 7) algama. Usage of the subroutines is described in the sequel.

Users who do not have access to the IMSL package could substitute their own subroutines. It is possible that IMSL could allow a user to obtain copies of these seven routines only for use in this program. For information, contact IMSL, NBC Building, 7500 Bellaire Blvd., Houston, Texas 77036; (713) 772-1927.

The test mentioned above may be described in more detail. Let $\underset{\sim}{\mathbf{S}}_h$ (pxp) be the sum product (SP) matrix computed from a random sample of size n_h from $N_p(\underset{\sim}{\mu}_h, \underset{\sim}{\Sigma}_h)$, i.e. p-variate normal population with mean vector $\underset{\sim}{\mu}_h$ and covariance matrix $\underset{\sim}{\Sigma}_h$ which is symmetric positive definite, h = 1,2. The two random samples are independently selected and $p < n_1, n_2$. Then a.e. the p characteristic roots of $\underset{\sim}{\mathbf{S}}_1 \underset{\sim}{\mathbf{S}}_2^{-1}$ are all positive. Further, let $\gamma_1, \ldots, \gamma_p$ be the characteristic roots of $\underset{\sim}{\Sigma}_1 \underset{\sim}{\Sigma}_2^{-1}$. Now the test of the hypothesis H_0: $\gamma_i = 1$, $i = 1,\ldots,p$, versus H_a: $\gamma_i \geq 1$, $i = 1,\ldots,p$, $\sum_{i=1}^{p} \gamma_i > p$ may be carried out at level α as follows:

Reject H_O if the largest characteristic root of $\underset{\sim}{S}_1(\underset{\sim}{S}_1+\underset{\sim}{S}_2)^{-1} = \theta_{p,m,n} > \theta_{1-\alpha,p,m,n}$, otherwise do not reject. Here $m = \frac{1}{2}(n_1-p-2)$, $n = \frac{1}{2}(n_2-p-2)$ and the upper $(1-\alpha)$ percentile may be found from the reference above (Pillai and Flury or references therein).

If one prefers to compute the P-value the program gives the method to do so, however the P-value may be used with confidence only if it does not exceed ten percent in view of the fact that it is computed from an approximation to the cdf of the largest root at the upper end (see Pillai and Flury (1984)).

While Section II describes the input, Section III gives the input for an example and Section IV the output for the example. The data consists of measurements on four variables (1) height (inches), (2) weight (pounds), (3) chest (inches) and (4) waist (inches) of male reserve officers in civilian status of the Armed Forces of the Philippines, hailing from two regions of the Philippine Islands but all within the age interval 29 to 31. The numbers of officers selected at random were 20 and 24 respectively from the first and second regions. (Normality assumption was found to be justified in view of earlier tests. Also for the original data see S. R. Ventura (1957). On the extreme roots of a matrix in multivariate analysis and associated tests. Unpublished thesis. The Statistical Center, University of the Philippines, Manila.)

Section V gives the program listing, including the usages of the seven subroutines listed above.

II. Input

Program: eqcov

Order of control cards:

a. jobcard
b. pfiles(get,datafile,x=tape2)
c. pfiles(get,eqcov,id=dnt)
d. mnf(i=eqcov,l=trash)
e. #eor
f. parameter card
g. format card
h. (input)
j. #s

Preparation of cards specific to this program:

parameter card	1-5	nv = number of variables (must be between 2 and 20 unless program modified)
	6-10	n1 = sample size of data set 1 (le 300 unless modified)
	11-15	n2 = sample size of data set 2 (le 300 unless modified)
	16-20	itape = 5 if information is on input itape = 2 if information is on tape
	21-25	ntype = 0 if data is used 1 if covariance matrices used 2 if cross product matrices used

Format card

Columns 1-80 may be used to describe the data format. Use the usual fortran format statement omitting the word format.

Data or matrix input

a. If data is input, enter each case on a new line. The format statement should describe one case. Enter n1+n2 cases, each case having nv variables. Enter cases for data set 1 followed by cases for data set 2.

b. If matrices are input, enter the lower triangular portion or the complete matrix for data set 1 followed by the lower triangular portion or the complete matrix for data set 2.

III. Input for Example

```
   4   20   24    5    2
(4f10.0)
  61.9880
 157.8800 4043.8000
   4.6200  295.7000   73.5500
   1.0900  479.1500   50.2250   81.6375
  72.0163
 195.5875 3045.9583
  31.3625  281.2083   92.4583
   2.6062  388.6875   36.8125   93.6563
```

IV. Output for Example

```
***test for equality of covariance matrices based on the largest root test***


number of variables is  4
sample size for data set 1 is  20
sample size for data set 2 is  24

data set 1 cross product matrix:

   61.9880
  157.8800 4043.8000
    4.6200  295.7000   73.5500
    1.0900  479.1500   50.2250   81.6375

data set 2 cross product matrix:

   72.0163
  195.5875 3045.9583
   31.3625  281.2083   92.4583
    2.6062  388.6875   36.8125   93.6563


eigenvalues:

     .3736     .6583    1.2219    1.5195


the largest root is      .6031


P-value is  .6662


***note: the P-value is approximate.
   if the P-value is .10 or below it can be used with confidence.
   otherwise, the value of the largest root may be compared to the tables.
```

V. Program Listing

```
      program main(input,output,tape5=input,tape6=output,tape2)

c*********************************************************************
c*                                                                   *
c*       This program is an algorithm for a test for equality        *
c*       of covariance matrices of two normal populations.           *
c*       It is based on the procedure described in:                  *
c*                                                                   *
c*       Pillai, K.C.S. and Flury, Bernhard N. (1984),               *
c*       "Percentage Points of the Largest Characteristic            *
c*       Root of the Multivariate Beta Matrix,"                      *
c*       Communications in Statistics- Theory and Methods,           *
c*       13(18), 2199-2237.                                          *
c*                                                                   *
c*                  Department of Statistics                         *
c*                     Purdue University                             *
c*                  West Lafayette, IN 47907                         *
c*                                                                   *
c*   programmer - Regina Becker                                      *
c*   Department of Statistics                                        *
c*   Purdue University                                               *
c*   January, 1985                                                   *
c*                                                                   *
c*********************************************************************
c
c
c      ***input information***
c
c  variables:  nv = number of variables (must be less
c                   than or equal to 20)
c
c              n1 = sample size of data set 1 (must be less than
c                   or equal to 300)
c
c              n2 = sample size of data set 2 (must be less than
c                   or equal to 300)
c
c              itape = the informational input source
c                      5 if information is on input
c                      2 if information is on tape
c
c              ntype = the informational input code
c                      0 if data is input
c                      1 if covariance matrices input
c                      2 if cross product matrices input
c
c
c  order of input:
c
c   1. parameter card      nv,n1,n2,itape,ntype
c                          use (5I5) format
c
c   2. format card         Columns 1-80 may be used to describe the
c                          informational input format.  Use the usual
c                          fortran type format statement, omitting
```

```
c                          the word format.
c
c  3. data or matrix input
c
c     a. If data is input, enter each case on a new line.  The format
c        statement should describe one case.  Enter n1+n2 cases, each
c        case having nv variables.  Enter cases for data set 1
c        followed by cases for data set 2.
c
c     b. If matrices are input, enter the lower triangular portion or
c        the complete matrix for each data set.  For example, if there
c        are 3 variables the covariance matrix input should be:
c
c               s11
c               s21 s22
c               s31 s32 s33
c               s41 s42 s43 s44
c               r11
c               r21 r22
c               r31 r32 r33
c               r41 r42 r43 r44
c
c       note: the s matrix is covariance matrix of data set 1
c             the r matrix is covariance matrix of data set 2
c
c*************************************************************************
c
c       The following subroutines are called by the program:
c
c       1. becovm - from the IMSL library, this subroutine calculates
c                   the cross product matrix of each data set if data
c                   is input.
c
c       2. linv2p - from the IMSL library, this subroutine finds the
c                   inverse of a matrix
c
c       3. ludecp - from the IMSL library, this subroutine performs a
c                   triangular decomposition of a matrix x such that
c                   x = l * l-transpose. l is a lower triangular
c                   matrix
c
c       4. vmulff - from the IMSL library, this subroutine performs
c                   multiplication of two matrices stored in full
c                   storage mode
c
c       5. vmulfs - from the IMSL library, this subroutine performs
c                   multiplication of two matrices a and b where a
c                   is stored in full storage mode and b is stored in
c                   symmetric storage mode
c
c       6. eigrs - from the IMSL library, computes the eigenvalues
c                  of a real symmetric matrix
c
c       7. algama - from the IMSL library, this function computes
c                   the natural log of the gamma function
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       subroutine becovm
c
```

```
c       usage: call becovm(x,ix,nbr,temp,xm,sp1,ier)
c
c               x     input matrix of dimension nbr(3) by nv for which cross
c                     product matrix is desired.  nbr(3)=sample size
c
c               ix    input, row dimension of x as specified in dimension
c                     statement in calling program. ix = n1size
c
c               nbr   input vector of length 6.
c                     nbr(1)  = nv
c                     nbr(2)  = number of observations per variable
c                     nbr(3)  = nbr(2)
c                     nbr(4)  = 1
c                     nbr(5)  = 1
c                     nbr(6)  = 1
c
c               temp input vector of length nv
c
c               xm    output vector of length nv containing variable means
c
c               sp1   output matrix of dimension nv by nv stored in symmetric
c                     storage mode requiring nv * (nv+1)/2 locations.  sp1
c                     contains the cross products matrix
c
c               ier   error parameter
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       subroutine linv2p
c
c       usage: call linv2p(sp2,nv,sp2inv,idgt,d1,d2,wkarea,ier)
c
c               sp2   nv by nv positive definite symmetric matrix to be
c                     inverted.  sp2 is stored in symmetric storage mode
c
c               nv    order of sp2
c
c               sp2inv  output vector of length nv(nv+1)/2 containing
c                         the inverse of sp2.  storage is symmetric mode.
c
c               idgt the approximate number of digits in the answer which
c                    were unchanged after improvement
c
c               d1,  components of the determinant of sp2
c               d2   determinant(sp2)=d1*2.**d2 (output)
c
c               wkarea  work area of dimension nsize (250)
c
c               ier   error parameter
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       subroutine ludecp
c
c       usage: call ludecp(sp2inv,ul,nv,d1,d2,ier)
c
c               sp2inv  input vector of length nv(nv+1)/2 containing the
c                         nv by nv positive definite symmetric matrix stored
c                         in symmetric storage mode
c
```

```
c              ul   output vector of length nv(nv+1)/2 containing the
c                   decomposed matrix l such that sp2inv=l*l-transpose.
c                   l is stored in symmetric storage mode.  The diagonal
c                   of l contains the reciprocals of the actual diagonal
c                   elements.
c
c              nv   order of sp2inv
c
c              d1,  components of the determinant of sp2inv.
c              d2   determinant(sp2inv) = d1*2.**d2 (output)
c
c              ier  error parameter
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c        subroutine vmulff
c
c        usage: call vmulff(c,l,r,c1,c2,ia,ib,lt,ic,ier)
c
c              c    r by c1 matrix stored in full storage mode
c
c              l    c1 by c2 matrix stored in full storage mode
c
c              r    number of rows in c = nv
c
c              c1   number of columns in c = nv
c
c              c2   number of columns in l = nv
c
c              ia   row dimension of matrix c as specified in the dimension
c                   statement in the calling program = maxnv
c
c              ib   row dimension of matrix l as specified in the dimension
c                   statement in the calling program = maxnv
c
c              lt   r by c2 matrix containing the product lt=c*l
c
c              ic   row dimension of matrix lt as specified in the dimension
c                   statement in the calling program = maxnv
c
c              ier  error parameter
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c        subroutine vmulfs
c
c        usage: call vmulfs(lt,sp1,l,m,ia,c,ic)
c
c              lt   l by m matrix stored in full storage mode
c
c              sp1  m by m symmetric matrix stored in symmetric storage mode
c
c              l    number of rows in lt = nv
c
c              m    number of columns in lt = nv
c
c              ia   row dimension of matrix lt as specified in dimension
c                   statement in the calling program = maxnv
c
c              c    l by m matrix containing the product c=lt*sp1
```

```
c
c               ic   row dimension of c as specified in the dimension
c                    statement in the calling program = maxnv
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c        subroutine eigrs
c
c        usage: call eigrs(ul,nv,jobn,d,c,ic,wkarea,ier)
c
c               ul   input real symmetric matrix of order nv whose
c                    eigenvalues are to be computed
c
c               nv   order of the matrix ul
c
c               jobn = 0  is an order to compute eigenvalues only
c
c               d    output vector of length nv containing eigenvalues of ul
c
c               c    output nv by nv matrix of eigenvectors
c
c               ic   input row dimension of c as specified in dimension
c                    statement in calling program = nv
c
c               wkarea  work area of length at least nv
c
c               ier  error parameter
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c        subroutine algama
c
c        usage: call algama(x)
c
c               x       input argument
c
c               algama  output value of the log base e of the absolute
c                       value of gamma(x)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       a special note:  a parameter statement has been used to set maximum
c                        sizes for these variables:
c
c                        n1   maximum is n1size = 300
c
c                        n2   maximum is n2size = 300
c
c                        nv   maximum is nvsize = 20
c
c                        ind  maximum is nvsize*(nvsize+1)/2 = 210
c
c                        wkarea maximum is nsize.  nsize must be at least
c                              nv(nv+1)/2 + n
c
c
c       These sizes may be increased if storage space on your computer allows
c       for larger array storage.  In that sense, variable limits can be
c       controlled by the programmer.
c
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      parameter(n1size=300,n2size=300,nvsize=20,ind=210,nsize=250)
      real pval,x(n1size,nvsize),temp(nvsize),sp1(ind),sp2(ind),
     *wkarea(nsize),sp2inv(ind),lt(nvsize,nvsize),l(nvsize,nvsize),
     *t(nvsize),ul(ind),c(nvsize,nvsize),h(nvsize),m,n
      real xm(nvsize),d(nvsize)

      integer df1,df2,fmt(80),nbr(6)
      double precision sum,f(nvsize),ff

c        read in nv=number of variables, n1=sample size of data set1
c        n2=sample size of data set2, itape (5 if data is input,
c        2 if data is on pfiles), ntype(0 if data, 1 if covariance matrix,
c        2 if cross product matrix)

         read(5,10) nv,n1,n2,itape,ntype
10       format(3i5,2i5)
         maxnv=20

         write(6,410)
410      format(1x,'***test for equality of covariance matrices based on',
     *   ' the largest root test***',//)

         write(6,412) nv,n1,n2
412      format(1x,'number of variables is ',i2/x,'sample size for data'
     * ' set 1 is ',i3/x,'sample size for data set 2 is ',i3/)
c        check that nv le maxnv

         if(nv.gt.maxnv)then
         write(6,8)
8        format(1x,'***number of variables must be less than 21'/)
         goto 900
         endif

         if(ntype.lt.0.or.ntype.gt.2)then
         write(6,12)
12       format(1x,'***check input type specification'/
     *   1x,'ntype=0 if data , 1 if covariance matrix, 2 if cross product
     *   matrix')
         goto 900
         endif
         if(itape.ne.5.and.itape.ne.2)then
         write(6,9)
9        format(1x,'***incorrect input medium specified'/
     *   1x,'use itape=5 if data is from input'/
     *   1x,'use itape=2 if data is from outside file'/)
         goto 900
         endif

         df1=n1-1
         df2=n2-1
         min=df1
         if(df2.lt.df1)min=df2
         if(nv.gt.df1.or.nv.gt.df2)then
         write(6,11) min
11       format(1x,'***number of variables must be less than the'/
     *   1x,'smaller degrees of freedom which is ',i4//)
         goto 900
         endif
```

```
      pie=4.*atan(1.0)
c     read data format and data

      read(5,20) (fmt(i),i=1,80)
 20   format(80a1)

      if(ntype.eq.0)then
      do 30 i=1,n1
      read(itape,fmt) (x(i,j),j=1,nv)
 30   continue


c     call becovm to get variance-covariance matrix

      nbr(1)=nv
      nbr(2)=n1
      nbr(3)=n1
      nbr(4)=1
      nbr(5)=1
      nbr(6)=1
      ix=n1size

      call becovm(x,ix,nbr,temp,xm,sp1,ier)

c     sp1 has cross product matrtix since nbr(6)=1
c     this is stored in symmetric storage mode

      do 40 i=1,n2
      read(itape,fmt) (x(i,j),j=1,nv)
40    continue

      nbr(2)=n2
      nbr(3)=n2
      ix=n2size
      call becovm(x,ix,nbr,temp,xm,sp2,ier)
      else
      do 43 i=1,nv
      ii=(i-1)*i/2
      read(itape,fmt) (sp1(ii+j),j=1,i)
43    continue
      index=nv*(nv+1)/2
      do 45 i=1,nv
      ii=(i-1)*i/2
      read(itape,fmt) (sp2(ii+j),j=1,i)
45    continue
      if(ntype.eq.2)go to 49
      do 47 i=1,index
      sp1(i)=sp1(i) * df1
      sp2(i)=sp2(i) * df2
 47   continue
49    endif

      write(6,400)
400   format(1x,'data set 1 cross product matrix:'/)

      do 401 i=1,nv
      ii=(i-1)*i/2
      write(6,402) (sp1(ii+j),j=1,i)
401   continue
```

```
      write(6,403)
403   format(/x,'data set 2 cross product matrix:'/)

      do 404 i=1,nv
      ii=(i-1)*i/2
      write(6,402)  (sp2(ii+j),j=1,i)
402   format(1x,12f10.4)
404   continue


c     find sp2 inverse

      call linv2p(sp2,nv,sp2inv,idgt,d1,d2,wkarea,ier)

c     sp2 inverse = sp2inv. decompose this to l*l transpose

      call ludecp(sp2inv,ul,nv,d1,d2,ier)

c     ul has matrix l such that l*l-transpose = sp2inv
c     diagonal of l has reciprocals of actual diagonal elements

      ul(1) = 1./ul(1)
      jj=1
      do 50 i=2,nv
      jj=jj+i
      ul(jj)=1./ul(jj)
50    continue


c     write l in full storage mode

      do 60 i=1,nv
      inc=(i-1)  * i / 2
      do 60 j=1,i
      l(i,j)=ul(inc+j)
      if(i.ne.j) l(j,i)=0.0
60    continue

c     multiply l-transpose * sp1 * l

      do 70 i=1,nv
      do 70 j=1,nv
      lt(j,i)=l(i,j)
70    continue

      call vmulfs(lt,sp1,nv,nv,maxnv,c,maxnv)
      call vmulff(c,l,nv,nv,nv,maxnv,maxnv,lt,maxnv,ier)

c     this results in l-transpose*sp1*l into lt. need characteristic
c     roots of lt.  convert this to ssm, put in ul(not used now).

      call vcvtfs(lt,nv,maxnv,ul)

c     get eigenvalues of ul

      call eigrs(ul,nv,0,d,c,nv,wkarea,ier)

      write(6,406)
406   format(//x,'eigenvalues:'/)
```

```
      write(6,666)  (d(i),i=1,nv)
666   format(1x,12f10.4//)
c     d contains eigenvalues, smallest to largest
c     arrange theta(i)  (called t(i))  largest to smallest
      do 80 i=1,nv
      t(nv-i+1)=d(i)/(1.+d(i))
80    continue


      m=float(df1-nv-1)/2.
      n=float(df2-nv-1)/2.

      rnv=float(nv)
      nvt=nv/2
      if(nv.eq.2*nvt)then
      hh=1.0
      else
      xx=t(1)
      call mdbeta(xx,m+1.,n+1.,hh,ier)
      endif

c     compute h(1) to h(nv)

      h(1)=1.0
      do 100 i=2,nv-1
      h(i)=float(nv-i+1)  *  (2.*m  +  float(nv-i+2))  *  h(i-1)/
     *       (float(i-1)*(2.*m  +  2.*n  +float(2*nv-i+2)))
100   continue
c     compute f
      f(1)=(m+n)/(m+n+rnv)
      do 120 i=2,nv-1
      f(i)=((m+n)*h(i)-(m+float(nv-i+1))*f(i-1))/
     *       (m+n+float(nv-i+1))
120   continue

      sum=0.0
      do 130 i=1,nv-1
      sum=sum+(-1.0)**i  *  f(i)  *  t(1)**(nv-i)
130   continue

      z1=.5*(2.*m+2.*n+2.*rnv+1.)
      z2=.5*(2.*m+rnv+1.)
      z3=.5*(2.*n+rnv+1.)
      z4=.5*rnv
      z5=.5*(2.*m+2.*n+rnv+2.)

      cst=(.5*alog(pie)+algama(m+n+rnv+1.)+algama(z1))-
     *     (alog(m+n)+algama(z2)+algama(z3)+algama(z4)+algama(z5))

c     find antilog of cst

      cst=exp(cst)
      ff=hh + cst *(t(1)**m)*((1.-t(1))**(n+1.))  *sum

      write(6,408)  t(1)
408   format(//x,'the largest root is ', f10.4)
      p=1.-ff
      write(6,407)  p
407   format(//x,'p-value is ',f6.4///)
```

```
      write(6,409)
409   format(1x,'***note: the p-value is approximate.')
      write(6,414)
414   format(4x,'if the p-value is .10 or below it can be used',
     *  ' with confidence.')
      write(6,415)
415   format(4x,'otherwise, the value of the largest root may be',
     *  ' compared to the tables.'//)

900   stop
      end
#eor
```