Subject: Empirical Orthogonal Functions
Posted by Sereno A. Barr-Kumara on Tue, 22 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Does any one have routines to do Empirical Orthogonal Function analysis
of arrays. I have looked at Peter Rileys IDL pro searchable
web page and other pages as well. I see that there are routines to
do principal components, but not EOF analysis.

Thanks

regards

barr-kum

Sereno A. Barr-Kumarakulasinghe	sbarrkum@ic.sunysb.edu
Marine Sciences Research Center	barr@kafula.msrc.sunysb.edu
State University of New York	
Stony Brook, NY 11794-5000	
http://pro.msrc.sunysb.edu/~sbarrkum/barr.html

Subject: Re: Empirical Orthogonal Functions
Posted by Sereno A. Barr-Kumara on Mon, 28 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

I recieved a lot of information on Empirical Orthagonal Function (EOF)
analysis. The individuals who replied were, B}rd Krane
<bard.krane@fys.uio.no>, "David S. Myers"
<dmyers@terra.MSRC.Sunysb.EDU>, Dennis Shea
<shea@cgd.ucar.edu> and Christian Marquardt
<marq@kassandra.met.fu-berlin.de>.
I have tried to summarise the content of the replys, and references and
what I understand of them. Also attached are IDL and FORTRAN 90
code for finding EOF's and the references. Note IDL 4.0 has a routine
called svdc which does SVD. The IDL code I have written is for IDL
3.6.1a

There are two similar methods for analysing the relationship between
variables. One is called a) SVD (Singular-Value Decomposition) or also
called Canonical Covariance or coinertia analysis and the other b) is
Canonical Corelation Analysis (CCA's). EOF analysis is the same as
Principal Component Analysis and is a special case of SVD (Newman &
Sardesmukh).

What does EOF do:
Each EOF finds the main (orthagonal) spatial patterns which describe the series. EOF's answer the question what are the most common patterns of x (Newman & Sardesmukh). Or in more mathematical terms, "EOF analysis is just obtaining the eigenvalues and eigenvactors of a covariance (real symmetric) matrix. The eigenvalues can be used to determine the % variance explained by each component and the eigenvectors are the patterns associatedwith each eigenvalue" (Marquardt).

The references mostly contain information and caveats about the use of SVD's and CCA to find information about coupled modes of two geophysical fields of interest. The caveats appear to be very IMPORTANT, ignorig them can result in erroneous interpretation of the physics. The caveat was that SVD could only be used sucessfully if the the two geophysical fields 1) were transformed orthagonal to each other or 2) the covariance matrix of either x or y is an identity martix. (Newman & Sardesmukh)

==========

References

C.S. Bretherton, C. Smith and J.M. Wallace, An intercomparison of methods for finding coupled patterns in climate data, J. Clim., 5, pp. 541-560, 1992 (this is the 'birth' of SVD's)

M. Newman and P.D. Sardeshmukh, A caveat concerning singular value decomposition, J. Clim. 8, pp. 352-360, 1995

S. Cherry, Singular value decomposition analysis and canonical correlation analysis, J. Clim., 9, pp. 2003-2009, 1996

==========

Thank You

barr-kum

Sereno A. Barr-Kumarakulasinghe	sbarrkum@ic.sunysb.edu
Marine Sciences Research Center	barr@kafula.msrc.sunysb.edu
State University of New York	
Stony Brook, NY 11794-5000	
http://pro.msrc.sunysb.edu/~sbarrkum/barr.html

IDL EOF Code:

```
function eofb, data

; USAGE: Result=eof(DATA)
; Result is a structure containing
; the following result.new, result.pct, result.co
; Performs an eof analysis of DATA, which is an NxM matrix
; where N=number of time series and M=number of points per series.
; NEW will be the modal time set of time series and PCT are the
; corresponding percentages of the variance that each mode 'explains'
; CO(i,j) is the correlation coefficient of the ith data series with
; the jth eof mode
; Original Matlab Code by: David Myers, ITPA, SUNY Stony Brook

; get sizes
n=(size(data))(1);
m=(size(data))(2);

;make covariance matrix

covar=data#transpose(data)
vectors=covar

tred2, vectors, values, offdiag   ;reduce matrix to tridiagonal form
tqli, values, offdiag, vectors

; sort eigenvalues and get percentages
val =values(sort(values))
tot=0
vec=fltarr(n,n)

for i=0,n-1 do begin
   vec(0:*,i)=vectors(0:*,(sort(values))(i));
   tot=tot + val(i);
endfor
pct=val/tot;

; make new data record

new=transpose(vec)#data;

; make correlation matrix

co=fltarr(n,m)

for i=0,n-1 do begin
   for j=0,m-1 do begin
```

```
    co(i,j)=correlate((transpose(data))(i,0:*), (transpose(new))(j,0:*));
  endfor
endfor

return, {new:new, pct:pct, co:co}

end
```

Fortran Code for SVD from Dennis J. Shea, Climate & Global Dynamics Div., NCAR

calls an lapack routine. Note: this uses autopmatic arrays so use f90. If u do not have an f90 compiler expand the argument calling list and pass them thru as arguments.

```
c -------------------------------------------------------
      subroutine drveof (x,nrow,ncol,nobs,msta,xmsg,neval,eval,evec
     *                  ,pcvar,trace,iopt,jopt,ier)

c driver to calculate :
c .   the principal components (eignvalues and corresponding
eigenvectors)
c .       of the data array x. x may contain missing observations.
c .       if it has msg data, this will calculate a var-cov matrix
c .       but it may not be positive definite.

c .   NOTE: the cov/cor matrix uses symmetric storage mode
c .         so only about half the memory is used.

c subroutine arguments are similar to historical routine
c .   "drvprc" but with no "work,lwork" arguments.

c .       USES LAPACK/BLAS ROUTINES for optimal performance
c .       USES AUTOMATIC ARRAYS SO USE WITH F90 (or Cray
f77)

c nomenclature :
c .   x         - matrix containing the data.  it contains n observations
c .               for each of m stations or grid pts.
c .   nrow,ncol - exact row (observation/time) and column (station/grid
pt)
c .               dimensions of x in the calling routine.
```

```
c .   nobs      - actual number of observations [time]       (nobs <= nrow)
c .   msta      - actual number of stations/grid pts [space] (msta <= ncol)
c .   xmsg       - missing code (if no obs are missing set to some
c .              number which will not be encountered)
c .   neval      - no. of eigenvalues and eigen vectors to be computed.
c .              neval <= msta
c .   eval       - vector containing the eigenvalues in DESCENDING order.
c .              eval must be at least neval in length.
c .   evec       - an array which will contains the eigenvector info.
c .              this must be dimensioned at least (ncol,neval) in the
c .              calling routine. There is some normalization done
c .              but I am not sure waht it is.
c .   pcvar      - contains the % variance associated with each eigenvalue.
c .              this must be dimensioned at least neval in length.
c .   trace      - trace of the variance - covariance matrix.in the
c .              case of a var-covar matrix , the trace should
c .              equal the sum of all possible eigenvalues.
c .   iopt       - set to zero (not used); kept for compatibility with old
routine
c .   jopt       - =  0 : use var-covar matrix in prncmp
c .                1 : use correlation matrix in prncmp
c .   ier        - error code

c NOTE: upon return if one want to get loadings,
c       something like this should be tried.
c
c           to calculate the coef or amplitude vector
c           associated with the n th eigenvector
c           mult the anomaly matrix (data) times the
c           transpose of the eigenvector
c               meval = neval
c               do n=1,meval
c                 do nyr=1,nyrs
c                    evyran(nyr,n) = 0.0
c                   do m=1,msta
c                     if (datam(nyr,m).ne.xmsg) then
c                      evyran(nyr,n) = evyran(nyr,n) + evec(m,n)*data(nyr,m)
c                     endif
c                   enddo
c                 enddo
c               enddo


      real x(1:nrow,1:ncol)
     *   , eval(neval) , evec(1:ncol,neval)
     *   , pcvar(neval)

      real    cssm(msta*(msta+1)/2)   , work(8*msta)     ! automatic arrays
```

```
      integer iwork(5*msta), ifail(msta)

      character*16 label
      double precision temp

      data ipr/6/
      data iprflg /1/

c calculate covariance or correlation matrix in symmetric storage mode

      ier = 0
      if (jopt.eq.0) then
         call vcmssm (x,nrow,ncol,nobs,msta,xmsg,cssm,lssm,ier)
      else
         call crmssm (x,nrow,ncol,nobs,msta,xmsg,cssm,lssm,ier)
      endif
      if (ier.ne.0) then
         write (ipr,'(//" sub drveof: ier,jopt= ",2i3
     1               ," returned from vcmssm/crmssm")') ier,jopt
         if (ier.gt.0) return          ! fatal: input dimension error
      endif

c activate if print of cov/cor matrix [work] is desired
c . must change format in "sub ssmiox" for nice looking output

      if (iprflg.eq.2) then
         if (jopt.eq.0) then
            label(1:15) = 'covar matrix:  '
         else
            label(1:15) = 'correl matrix: '
         endif
         write (ipr,'(//,a15,"sym storage mode")') label(1:15)
         call ssmiox (cssm,msta)
      endif

c calculate the trace  before it is destroyed by sspevx

      na = 0
      temp = dble( 0. )
      do nn=1,msta
        na = na+nn
        temp = temp + dble( cssm(na) )
      enddo
      if (temp.eq.dble(0.)) then
         ier = -88
         write (ipr,'(//" sub drveof: ier,jopt= ",2i3
     *               ," trace=0.0")') ier,jopt
         return
```

```
      endif
      trace = real(temp)

c calculate the specified number of eigenvalues and the corresponding
c .   eigenvectors.

      meval = min(neval,msta)               ! neval <= msta

      tol  = 10.*epsmach (ipr)
      vlow = 0.0                            ! not used
      vup  = 0.0                            ! not used
      ilow = max(msta-meval+1,1)
      iup  = msta
      call sspevx ('V','I','U',msta,cssm,vlow,vup,ilow,iup,tol
     *           ,mevout,eval,evec,ncol,work,iwork,ifail,info)

      if (info.ne.0) then
         ier = ier+info
         write (ipr,"(//,' sub drveof: sspevx error: info='
     *                 ,   i9)") info
         write (ipr,"(   ' sub drveof: sspevx error: ifail='
     *                 ,   20i5)") (ifail(i),i=1,min(20,msta))
      endif

c reverse the order of things from "sspevx" so
c .   largest eigenvalues/vectors are first.

      do n=1,meval                  ! eigenvalues
        work(n) = eval(n)
      enddo
      do n=1,meval
        eval(n) = work(neval+1-n)
      enddo

      do n=1,msta                   ! eigenvectors
        do nn=1,neval
          work(nn) = evec(n,nn)
        enddo
        do nn=1,neval
          evec(n,nn) = work(neval+1-nn)
        enddo
      enddo

      do n=1,meval                  ! % variance explained
        pcvar(n) = (eval(n)/trace)*100.
      enddo

      return
```

```
      end
c  ------------------------------------------------------------ -------
      subroutine vcmssm (x,nrow,ncol,nrt,ncs,xmsg,vcm,lvcm,ier)

c this routine will calculate the variance-couariance matrix (vcm)
c .   of the array x containing missing data. obviously if x does contain
c .   missing data then vcm is only an approximation.

c note : symmetric storage mode is utilized for vcm to save space.

c input:
c     x        - input data array  ( unchanged on output)
c     nrow,ncol- exact dimensions of x in calling routine
c     nrt,ncs  - dimension of sub-matrix which contains the data
c                (nrt <= nrow : ncs <= ncol)
c     xmsg     - missing data code (if none set to some no. not
encountered)
c output:
c     vcm      - var-cov matrix
c     lvcm     - length of vcm
c     ier      - error code (if ier=-1 then vcm contains missing entry)

      dimension x(1:nrow,1:ncol) , vcm(*)

      ier  = 0
      if (nrow.lt.1 .or. ncol.lt.1) ier = ier+1
      if (nrt .lt.1 .or. ncs .lt.1) ier = ier+10
      if (ier.ne.0) return

      lvcm = ncs*(ncs+1)/2

c calculate the var-cov between columns (stations)

      nn = 0
      do 30 nca=1,ncs
      do 30 ncb=1,nca
      xn    = 0.
      xca   = 0.
      xcb   = 0.
      xcacb = 0.
      nn    = nn + 1
      do 10 i=1,nrt
      if (x(i,nca).ne.xmsg .and. x(i,ncb).ne.xmsg) then
         xn    = xn + 1.
         xca   = xca + x(i,nca)
         xcb   = xcb + x(i,ncb)
         xcacb = xcacb + ( x(i,nca) )*( x(i,ncb) )
      endif
```

```
   10 continue
      if (xn.ge.2.) then
         vcm(nn) = ( xcacb-(xca*xcb)/xn )/(xn-1.)
      else
         ier     = -1
         vcm(nn) = xmsg
      endif
   30 continue

      return
      end
c  ------------------------------------------------------------ -------
      subroutine crmssm (x,nrow,ncol,nrt,ncs,xmsg,crm,lcrm,ier)

c this routine will calculate the correlation matrix   (crm)
c .   of the array x containing missing data. obviously if x does contain
c .   missing data then crm is only an approximation.

c note : symmetric storage mode is utilized for crm to save space.

c input:
c     x       - input data array  ( unchanged on output)
c     nrow,ncol- exact dimensions of x in calling routine
c     nrt,ncs  - dimension of sub-matrix which contains the data
c                (nrt <= nrow : ncs <= ncol)
c     xmsg     - missing data code (if none set to some no. not
encountered)
c output:
c     crm      - correlation matrix
c     lcrm     - length of crm
c     ier      - error code (if ier=-1 then crm contains missing entry)

      real x(1:nrow,1:ncol) , crm(*)

      ier  = 0
      if (nrow.lt.1 .or. ncol.lt.1) ier = ier+1
      if (nrt .lt.1 .or. ncs .lt.1) ier = ier+10
      if (ier.ne.0) return

      lcrm = ncs*(ncs+1)/2

c calculate the var-cov between columns (stations)

      nn   = 0
      do 30 nca=1,ncs
      do 30 ncb=1,nca
      xn    = 0.
      xca   = 0.
```

```
      xcb   = 0.
      xca2  = 0.
      xcb2  = 0.
      xcacb = 0.
      nn    = nn + 1
      do 10 i=1,nrt
      if (x(i,nca).ne.xmsg .and. x(i,ncb).ne.xmsg) then
         xn    = xn + 1.
         xca   = xca + x(i,nca)
         xcb   = xcb + x(i,ncb)
         xca2  = xca2 + x(i,nca)*x(i,nca)
         xcb2  = xcb2 + x(i,ncb)*x(i,ncb)
         xcacb = xcacb + x(i,nca)*x(i,ncb)
      endif
   10 continue
      if (xn.ge.2.) then
         xvara   =  ( xca2-((xca*xca)/(xn)) )/(xn-1.)
         xvarb   =  ( xcb2-((xcb*xcb)/(xn)) )/(xn-1.)
         if (xvara.gt.0. .and. xvarb.gt.0.) then
            crm(nn) = ( xcacb-((xca*xcb)/(xn)) )/(xn-1.)
            crm(nn) = crm(nn)/(sqrt(xvara)*sqrt(xvarb))
         else
            ier     = -1
            crm(nn) = xmsg
         endif
      else
         ier     = -1
         crm(nn) = xmsg
      endif
   30 continue

      return
      end
c  ------------------------------------------------------------ ------------------
      subroutine ssmiox (x,ncs)

c output a symmetric storage mode matrix [x]

c note: format statement have to be changed

c .   x(1)
c .   x(2)  x(3)
c .   x(4)  x(5)  x(6)
c .   etc.

      real x(1:*)

      data ipr/6/
```

```
      ncend=0
      do 10 nc=1,ncs
      ncstrt = ncend+1
      ncend  = ncstrt+nc-1
      write(ipr,'(2x,i5,10(1x,f12.3),/, (7x,(10(1x,f12.3))))')
     1           nc,(x(n),n=ncstrt,ncend)
   10 continue

      return
      end
c  ------------------------------------------------------------ -
      real function epsmach (ipr)

      integer ipr    ! =1 print else no print
      real    eps

      eps = 1.0
    1 eps = 0.5*eps
      if ((1.0+eps) .gt. 1.0) go to 1
      epsmach = 2.0*eps
      if (ipr.eq.1) write (*,"('EPSMACH: eps=',e15.7)") epsmach

      return
      end

 ============================================================ ======
```