Subject: SVDC for EOF analysis of time series data
Posted by ivitseva on Wed, 02 May 2012 13:08:32 GMT
View Forum Message <> Reply to Message

Dear All,

I've spent the last two weeks with finding my way in scripting PCA/EOF analysis on time series of SPI (Standardised Precipitation Index) using the SVDC routine. I DO NEED to use Singular Value Decompostion for PCA/EOF rather than the traditional eigenql routine because once I've managed the SVD I wanna move on to CCA and the Multi Taper Method (which also use SVD).

I've scripted the method in two ways: one applies svdc on the original data matrix of anomalies being a [time,space] matrix, the other applies svdc on the covariance matrix of the anomalies, where the covariance matrix is a T-mode matrix of [time,time]. For the latter I've alse used David Fanning's EOF solution under: http://www.idlcoyote.com/code_tips/eof_analysis.html. Thanks David! In both methods I present the eigenvectors as the EOF spatial pattern (this I call EV) and I also use another way of presenting the EOF, which is correlating (or regressing) the Principal Components with the original data. See e.g. http://www.ess.uci.edu/~yu/class/ess210b/lecture.5.EOF.all.p df.
The good news is that I get identical results with these two scripts, the bad news is that I get very different result from ENVI's PCA transform (on the covariance matrix as well).
Probably I'm doing something wrong, somewhere I must use a reform function wrongly or...I don't know. It drives me crazy, ANY help is greately appreciated and thank you in advance!!!

I'm attaching here the two scripts and I've copied the input SPI time series (a small subset of 200*200) and the two scripts onto our FTP site for any good merciful soul who has time to relive me from my pain:
ftp://xftp.jrc.it/outgoing/eof/eof.rar
login:anonymous
password: (no passwprd is needed)

And here are the scripts:

SVD on the covariance matrix

```
pro eof_svdc2

STARTTIME = SYSTIME(1)

infile=dialog_pickfile(filter='*.bil') ; select the file as input for
the EOF analysis
```

```
strpos=strpos(infile,'\',/reverse_search)

outev=dialog_pickfile(filter='*.bil',/write,title='Write the EV
file')
outeof=dialog_pickfile(filter='*.bil',/write,title='Write the EOF
file')

ns=200L ; number of lines
nl=200L ; number of samples
nb=144

im=fltarr(ns,nb,nl,/nozero)
openr,lun,infile,/get_lun
readu, lun, im

;reorganize the SPI into a two dimensional array of spatial
measurements vs. time
inarr=reform(transpose(im,[0,2,1]),ns*nl,nb)
free_lun,lun

;center the time series
for j=0L,ns*nl-1 do inarr[j,*]-=mean(inarr[j,*])

;ignore NaN values
nonan=where(finite(total(inarr,2)),nw)
modarr=inarr[nonan,*]

;calculate the covariance matrix
covmatrix=(1/float(nb-1))*(Double(modarr)## transpose(modarr))

;calculate eigenvectors and eigenvalues
print,'starting svdc'
svdc,covmatrix,W,U,V ;w[time],U[time,time],V[time,time]

print,'calculating the EOFs from the eigenvectors (U)'
s=size(modarr,/dimensions)
EV=fltarr(s[1],s[0])
for j=0,s[1]-1 do begin
t=transpose(modarr)##U[j,*]
EV[j,*]=t/sqrt(Total(t^2)) ;this standardises the EV with its
euclidean norm
endfor

print,'calculating the PCs'
PC=fltarr(s[1],s[1])
for j=0,s[1]-1 do PC[j,*]=modarr##EV[j,*]

print,'calculating the EOFs from the PCs'
```

```
inarr=reform(inarr,ns,nl,nb)
EOF=fltarr(ns,nl,10,/nozero) ;I only calculate the first 10 EOFs to
save time
for p=0,10-1 do begin
for i=0,ns-1 do begin
for j=0,nl-1 do begin
EOF[i,j,p]=correlate(PC[p,*],reform(inarr(i,j,*)))
endfor
endfor
endfor

EVout=fltarr(nb,ns*nl,/nozero)
EVout[*,nonan]=EV[*,*]
EVout=transpose(reform(EVout,nb,ns,nl),[1,0,2])
EOF=transpose(EOF,[0,2,1])

print,'Finished EOF transforms'
percvar=W/total(W)*100.0
print,'variance explained:'+percvar

openw,lun,outev,/get_lun
print,'write the EOF file'
writeu,lun,float(EVout)
free_lun,lun

print,'write the EV header'
; WRITE HEADER OF THE OUTPUT
HEADER_OUT=STRMID(outev,0,STRLEN(outev)-3)+'hdr'
OPENW, 3, HEADER_OUT
printf,3,'ENVI'
printf,3,'description = {'
printf,3,' EV}'
printf,3,'samples ='+strcompress(ns)
printf,3,'lines ='+strcompress(nl)
printf,3,'bands ='+strcompress(nb)
printf,3,'header offset = 0'
printf,3,'file type = ENVI Standard'
printf,3,'data type = 4
printf,3,'interleave = bil'
printf,3,'byte order = 0'
printf,3,'map info = {Geographic Lat/Lon, 1.0000, 1.0000, 6.54163552,
53.02878674, 8.3333300000e-002, 8.3333300000e-002, WGS-84,
units=Degrees}'
CLOSE, 3

openw,lun,outeof,/get_lun
print,'write the EOFPC file'
writeu,lun,float(EOF)
```

```
free_lun,lun

print,'write the EOF header'
; WRITE HEADER OF THE OUTPUT
HEADER_OUT=STRMID(outeof,0,STRLEN(outeof)-3)+'hdr'
OPENW, 3, HEADER_OUT
printf,3,'ENVI'
printf,3,'description = {'
printf,3,' EOF}'
printf,3,'samples ='+strcompress(ns)
printf,3,'lines ='+strcompress(nl)
;printf,3,'bands ='+strcompress(nb)
printf,3,'bands=10'
printf,3,'header offset = 0'
printf,3,'file type = ENVI Standard'
printf,3,'data type = 4
printf,3,'interleave = bil'
printf,3,'byte order = 0'
printf,3,'map info = {Geographic Lat/Lon, 1.0000, 1.0000, 6.54163552,
53.02878674, 8.3333300000e-002, 8.3333300000e-002, WGS-84,
units=Degrees}'
CLOSE, 3

; Evaluation of processing time
ELAPSED_TIME = FIX(SYSTIME(1) - STARTTIME)
MINUTES = ELAPSED_TIME / 60
SECS=ELAPSED_TIME MOD 60
PRINT, 'PROCESSING TOOK :'+STRCOMPRESS(MINUTES)+' MINUTES
AND'+STRCOMPRESS(SECS)+' SECONDS'
PRINT, 'Finished the process'
end

 ------------------------------------------------------------ ----------
SVD on the matrix of anomalies

pro eof_svdc3

STARTTIME = SYSTIME(1)

infile=dialog_pickfile(filter='*.bil') ; select the file as input for
the EOF analysis
strpos=strpos(infile,'\',/reverse_search)
outEV=dialog_pickfile(filter='*.bil',/write,title='Write the EV
file')
outEOF=dialog_pickfile(filter='*.bil',/write, title='Write the EOF
file')

ns=200L ; number of lines
```

```
nl=200L ; number of samples
nb=144
im=fltarr(ns,nb,nl,/nozero)
openr,lun,infile,/get_lun
readu, lun, im

;reorganize the SPI into a two dimensional array of time vs. spatial
measurements
inarr=reform(transpose(im,[1,0,2]),nb,ns*nl)
free_lun,lun

;ignore NaN values
nonan=where(finite(total(inarr,1)),nw)
modarr=inarr[*,nonan]

;center the data
for j=0L,nw-1 do modarr[*,j]-=mean(modarr[*,j]);[time,space]

;calculate the SVD: svdc,input,e_value,e_vector, pc
print,'SVDC decomposition'
svdc,modarr,W,U,V

print,'Calculate the EOFs from the eigenvectors'
EV=fltarr(nb,ns*nl,/nozero)
EV[*,nonan]=U[*,*]
EV=transpose(reform(EV,nb,ns,nl),[1,0,2])

print,'Calculate the PCs'
PC=fltarr(nb,nb)
for p=0,nb-1 do begin
PC[p,*]=reform(V[p,*]/STDEV(V[p,*]))
;the division normalises PCs setting the standard deviation to unity
and mean to 0
endfor

print,'Calculate the EOFs from the PCs'
inarr=transpose(reform(inarr,nb,ns,nl),[1,2,0])
EOF=fltarr(ns,nl,10,/nozero)
for p=0,9 do begin
for i=0,ns-1 do begin
for j=0,nl-1 do begin
 EOFPC[i,j,p]=correlate(reform(PC[p,*]),reform(inarr[i,j,*]))
endfor
endfor
endfor

EOF=transpose(EOF,[0,2,1])
```

```
openw,lun,outEV,/get_lun
print,'write the output file'
writeu,lun,float(EV)
free_lun,lun

print,'write the header'
HEADER_OUT=STRMID(outEV,0,STRLEN(outEV)-3)+'hdr'
OPENW, 3, HEADER_OUT
printf,3,'ENVI'
printf,3,'description = {'
printf,3,' EV}'
printf,3,'samples ='+strcompress(ns)
printf,3,'lines ='+strcompress(nl)
printf,3,'bands ='+strcompress(nb)
printf,3,'header offset = 0'
printf,3,'file type = ENVI Standard'
printf,3,'data type = 4
printf,3,'interleave = bil'
;printf,3,'sensor type = AVHRR'
printf,3,'byte order = 0'
printf,3,'map info = {Geographic Lat/Lon, 1.0000, 1.0000, 6.54163552,
53.02878674, 8.3333300000e-002, 8.3333300000e-002, WGS-84,
units=Degrees}'
CLOSE, 3

openw,lun,outEOF,/get_lun
print,'write the output file'
writeu,lun,float(EOF)
free_lun,lun

print,'write the header'
HEADER_OUT=STRMID(outEOF,0,STRLEN(outEOF)-3)+'hdr'
OPENW, 3, HEADER_OUT
printf,3,'ENVI'
printf,3,'description = {'
printf,3,' EOF}'
printf,3,'samples ='+strcompress(ns)
printf,3,'lines ='+strcompress(nl)
printf,3,'bands = 10'
printf,3,'header offset = 0'
printf,3,'file type = ENVI Standard'
printf,3,'data type = 4
printf,3,'interleave = bil'
;printf,3,'sensor type = AVHRR'
printf,3,'byte order = 0'
printf,3,'map info = {Geographic Lat/Lon, 1.0000, 1.0000, 6.54163552,
53.02878674, 8.3333300000e-002, 8.3333300000e-002, WGS-84,
units=Degrees}'
```

```
CLOSE, 3

; Evaluation of processing time
ELAPSED_TIME = FIX(SYSTIME(1) - STARTTIME)
MINUTES = ELAPSED_TIME / 60
SECS=ELAPSED_TIME MOD 60
PRINT, 'PROCESSING TOOK :'+STRCOMPRESS(MINUTES)+' MINUTES
AND'+STRCOMPRESS(SECS)+' SECONDS'
PRINT, 'Finished the process'
end
```
