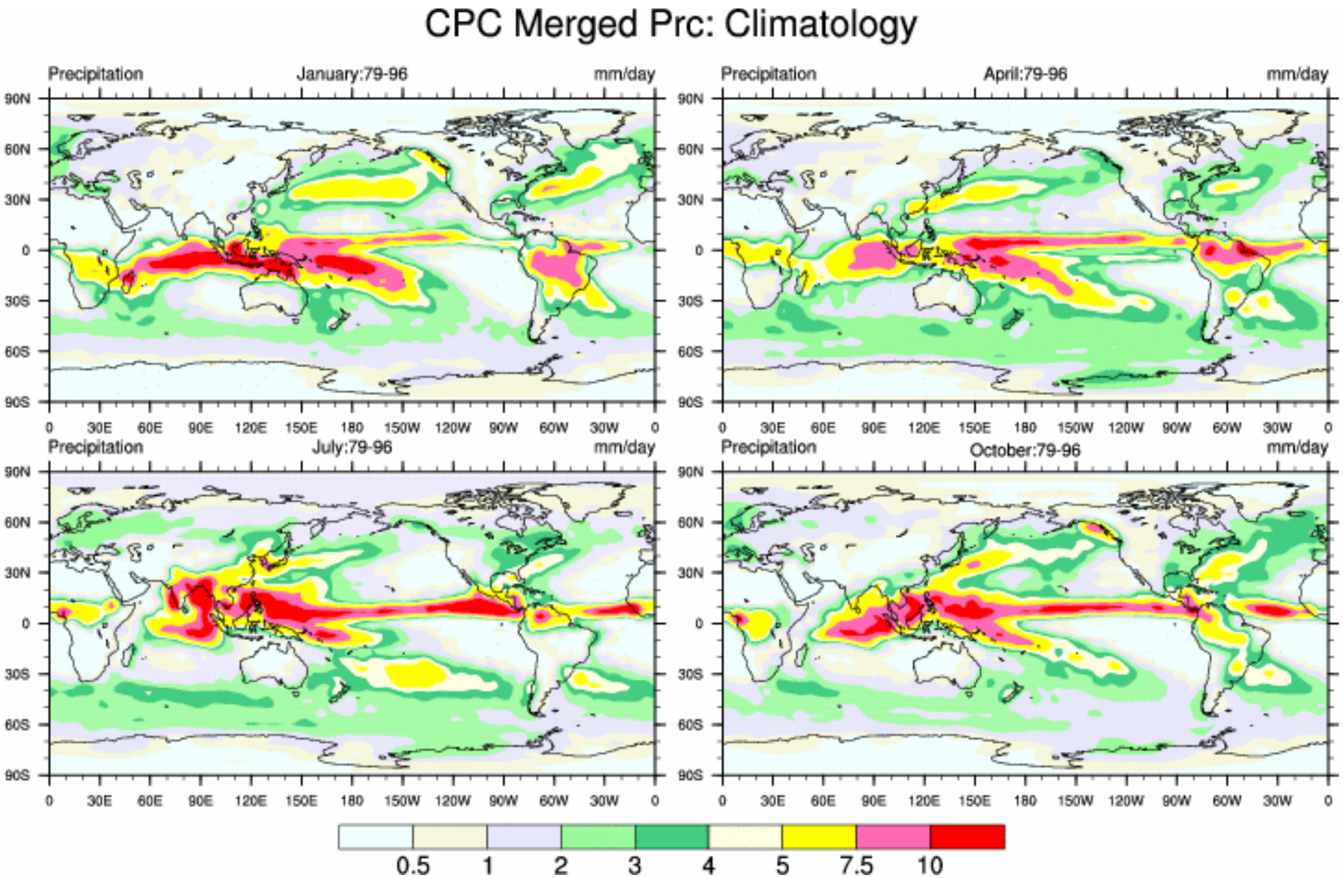
NCL Data Processing



Dennis Shea

National Center for Atmospheric Research

NCAR is sponsored by the National Science Foundation

Empirical Orthogonal Functions (EOFs)

- **EOFs** <=> **Principal Components (PCs)**
- widely used **statistical** technique
- **eigenvectors** of covariance matrix between grid pts (stations)
- **not** based on physical principals
- used to **explore** data

- let **x** => f(**T**ime, **S**pace)
  - example: slp(time,lat,lon), **T**=>time, **S**=lat,lon

- partitioned/decomposed into **orthogonal** patterns/modes
  - efficient representation of '**system variance**'
  - **linear combinations** that compress the data
  - 1st linear combination explains largest variance
- **spatial** patterns **& time series** of each pattern's amplitude
- **may/may-not** have explainable physical info

EOF: functions

NCL has two functions:

eofunc_Wrap: calculates **orthogonal** patterns/modes

eofunc_ts_Wrap: calculates pattern/mode **amplitudes**

- **eofunc_Wrap**
  - expects '**time**' dimension to be **rightmost** dimension
    - may have to reorder using **named dimensions**
    - **x** should be **weighted** to reflect spatial extent
    - **user** specifies number of EOFs (rarely more than 4)

EOF: eofunc Calculation Details

Examines the **S**patial & **T**emporal sizes (**S**,**T**) of input **x**

- may do a linear transformation to yield smallest **COV(x)**
  - generally, **T** << **S** ; hence, **T**x**T** in sym. storage mode
- if linear transformation performed; reverse transform

anomaly covariance matrix created (or **correlation** matrix)

- covariance between the i^{th} and j^{th} locations over time (N)
- $\mathrm{cov}(\mathbf{xa})_{i,j} = [\ \Sigma(\mathbf{x}_{n,i} - \mathbf{X}_i)\ (\mathbf{x}_{n,j} - \mathbf{X}_j)\ ]/(N-1)$
  - $\mathbf{X}_i$, $\mathbf{X}_j$ are temporal means of **x** at each location
  - **xa** is the **a**nomaly covariance matrix

EOFs (patterns/modes): LAPACK's "**dspevx**"

- user specifies number of EOFs to return (K)
- returns eigenvalues; % variance explained

ts: amplitude time series:

- for each EOF_k: $ts_{k,n} = \Sigma(EOF_{i,j,k} * \mathbf{ax}_{i,j,n})$

EOF: eofunc returned info

- **EOFs** (spatial)
- **% variance explained** by each EOF
- **eigenvalues** of the covariance matrix
    - if applicable, eigenvalues of transformed matrix also

EOFS: Simple Example (1)

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
                                                          ; rectilinear
f  = addfile("erai_1989-2009.mon.msl_psl.nc","r")         ; open file
p  = f->SLP(::12, {0:90}, :)                              ; (21,61,240)
                                                          ; spatial weighting
w   = sqrt(cos(0.01745329*p&latitude) )                   ; weights(61)
wp = p*conform(p, w, 1)                                   ; wp(21,61,240)
copy_VarCoords(p, wp)

x        = wp(latitude|:,longitude|:,time|:)              ; reorder wgt data
neof     = 4                                              ; user specify
eof      = eofunc_Wrap(x, neof, False)
eof_ts = eofunc_ts_Wrap (x, eof, False)

printVarSummary( eof )                                  ; examine EOF variables
printVarSummary( eof_ts )
```

EOFS: Simple Example (1)

"**printVarSummary**" output

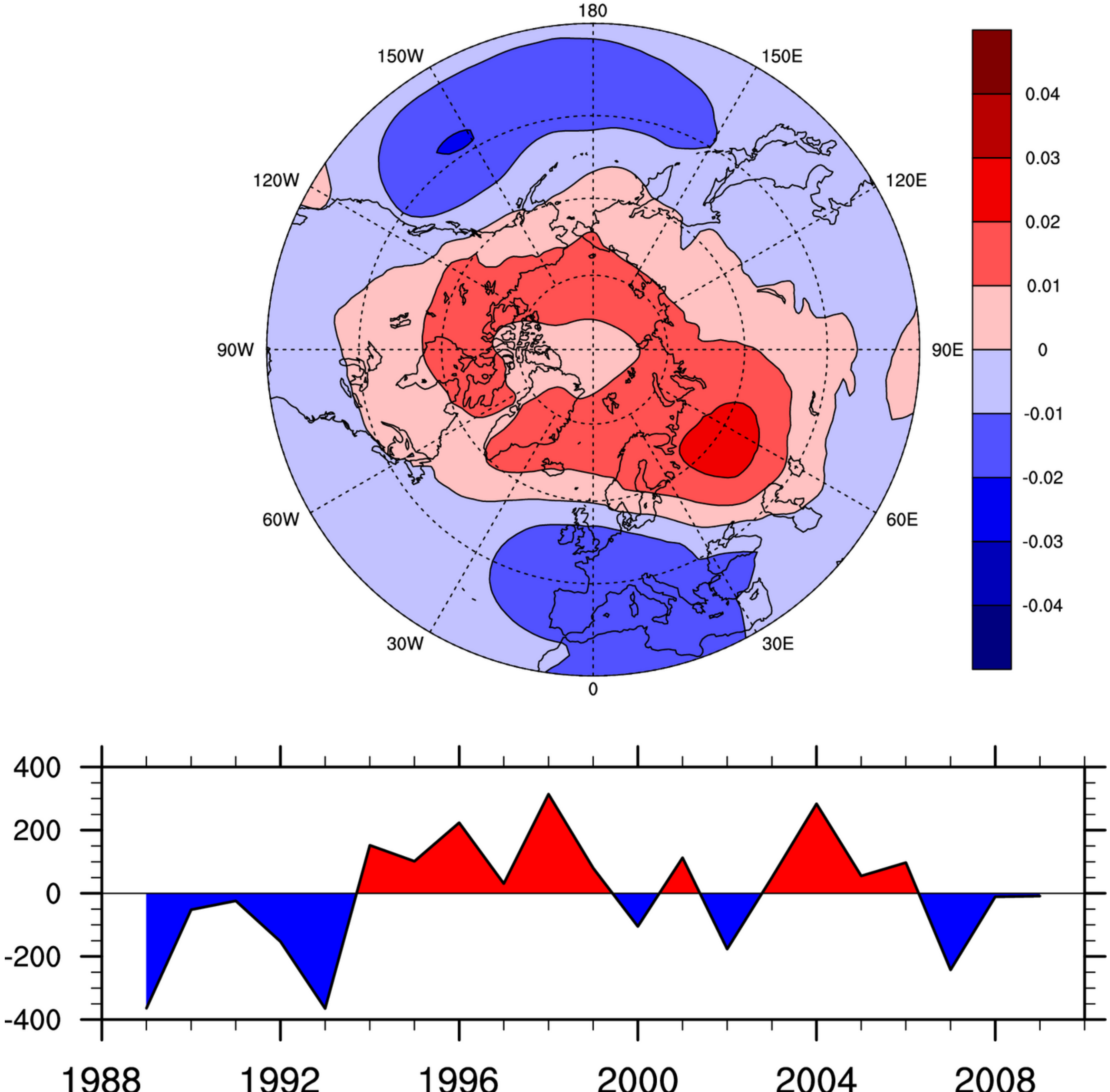
```
Variable: eof
Number of Dimensions: 3
Dimensions and sizes: [evn | 4] x [latitude | 61] x [longitude | 240]
Coordinates:
            evn: [1..4]
            latitude: [ 0..90]
            longitude: [ 0..358.5]
Number Of Attributes: 6
  eval_transpose :    ( 47.2223, 32.42917, 21.44406, 15.27389 )
  eval :       ( 34519.5, 23705.72, 15675.61, 11165.21 )
  pcvar :      ( 26.83549, 18.42885, 12.18624, 8.679848 )
  matrix :     covariance
  method :     transpose
  _FillValue : 1e+20

Variable: eof_ts
Number of Dimensions: 2
Dimensions and sizes: [evn | 4] x [time | 21]
Coordinates:
            evn: [1..4]
            time: [780168..955488]
Number Of Attributes: 3
  ts_mean :    ( 3548.64, 18262.12, 20889.75,10387.08 )
  matrix :     covariance
  _FillValue : 1e+20
```

EOF: write a NetCDF file

```
; Create netCDF: no define mode [simple approach]
system("/bin/rm -f EOF.nc")                ; rm any pre-existing file
fout          = addfile("EOF.nc", "c")     ; new netCDF file
fout@title    = "EOFs of SLP 1989-2009"
fout->EOF     = eof
fout->EOF_TS  = eof_ts
```

Graphics: http://www.ncl.ucar.edu/Applications/Scripts/eof_2.ncl

SLP: 1989-2009: EOF 1: % Variance=26.8



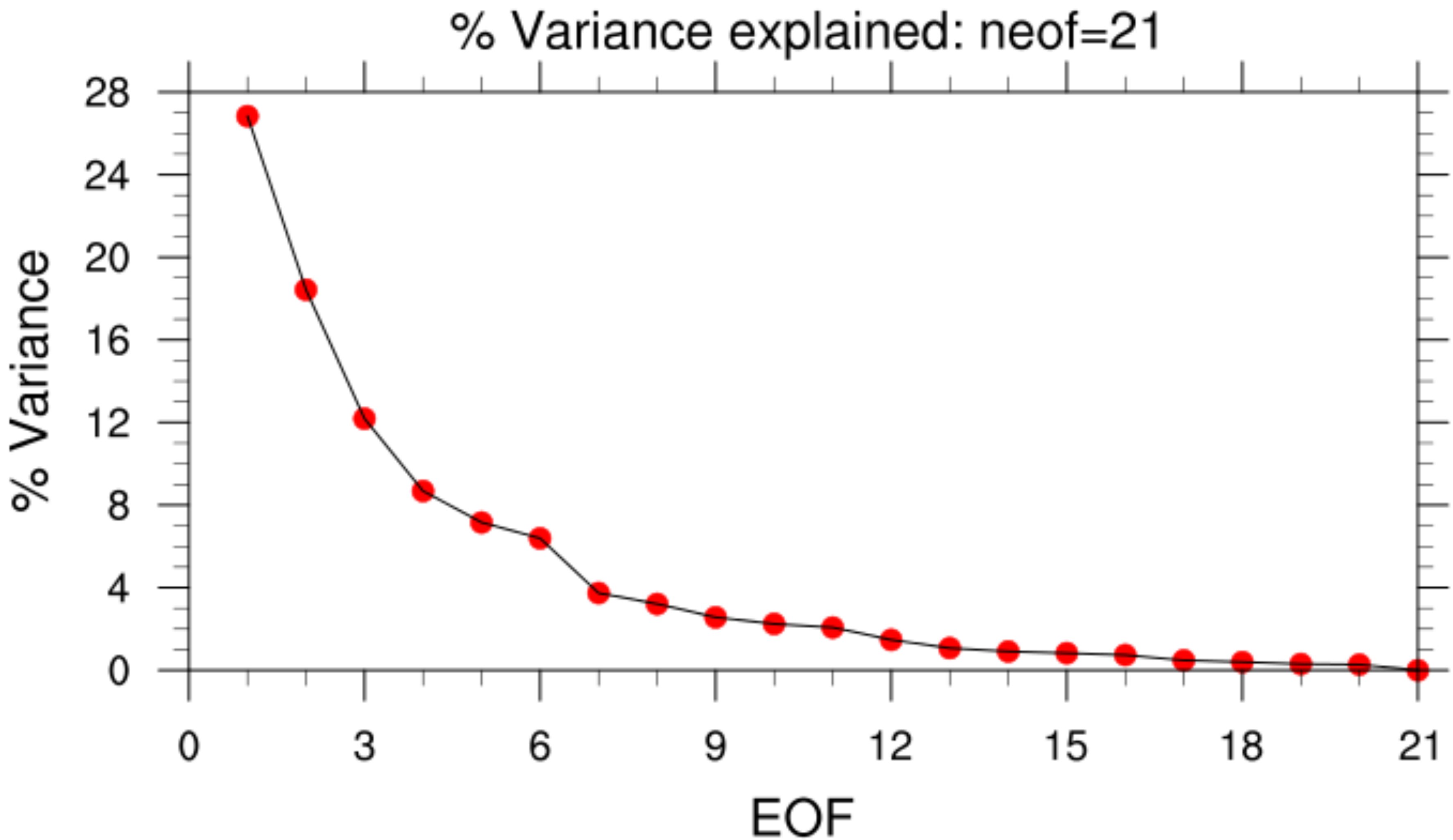
EOF: significance

- **successive eigenvalues should be distinct**
  - if not, the eigenvalues and associated patterns are noise
  - 1 from 2, 2 from 1 and 3, 3 from 2 and 4, etc

  - North et. al (*MWR*, July 1982: eq 24-26) provide formula
    - http://dx.doi.org/10.1175/1520-0493(1982)110<0699:SEITEO>2.0.CO;2
  - Quadrelli et. Al (*JClimate*, Sept, 2005) more information
    - http://dx.doi.org/10.1175/JCLI3500.1

- **NOTE:** patterns are domain dependent

EOF: Sample % Variance Distribution



Shape is ‘red’

EOF: North (1982): eigenvalue separation: function

North et al (1982): eqn 22: this is an ‘objective approximation’

$$\delta\lambda \approx \lambda\left(\sqrt{\frac{2}{N}}\right)$$

```
undef ("eval_north")
function eval_north( eval[*]:numeric, ntim[1]:integer, prinfo[1]:logical)
local neval, dlam, low, high, sig, n
begin
  neval   = dimsizes(eval)
  dlam    = eval * sqrt(2.0/ntim)          ; eq 22 of North et al. (1982): Mon. Wea. Rev
  low     = eval  - dlam
  high    = eval + dlam

  sig     = new(neval, logical)
  sig     = False                          ; default is not significantly separated
```

EOF: function eigenvalue separation

```
                                    ; take care of 1st and last special cases
  if (eval(0).gt.high(1)) then              ; 1st eigenvalue (index 0)
     sig(0) = True
  end if
  if (eval(neval-1).lt.low(neval-2)) then   ; last eigenvalue (index ‘neval-1’)
     sig(neval-1) = True
  end if

  do n=1,neval-2                            ; loop over all other eigenvalues
    if (eval(n).lt.low(n-1) .and. eval(n).gt.high(n+1)) then
       sig(n) = True
    end if
  end do

  if (prinfo) then
     print(dlam+"   "+low+"   "+eval+"   "+high+"  "+sig)
  end if
  sig@long_name = ”eval significantly separated"
  return(sig)
end
```

EOF: eigenvalue separation: script output

North et al (1982) test:

$$\delta\lambda \approx \lambda\left(\sqrt{\frac{2}{N}}\right)$$

```
prinfo = True
sig    = eval_north(eof@eval, ntim, prinfo)
```

eval index	**dlam**	low	**eval**	high	**sig**
(0)	10652.9	23866.6	34519.5	45172.4	True
(1)	7315.8	16390	23705.7	31021.5	True
(2)	4837.6	10838	15675.6	20513.2	True
(3)	3445.7	7719.6	11165.2	14610.9	False
(4)	2841.2	6365.3	9206.5	12047.6	False

[snip]

EOF: Rotation via Varimax

- rotates EOFs via Kaiser **varimax** criterion
    - rotated EOFs will be orthogonal
    - time series will be correlated (not orthogonal)
- how many EOFs should be used? No objective method!

- **my opinion**
    - you should (? **must** ?) know what you are doing
    - use when no significant EOFs were derived
        - rotation may reduce noise and yield interpretable info
        - **however, if some are distinct and some are not then performing a rotation will mix the results**

```
eof_vmax = eofunc_varimax_Wrap(eof, 1)
eofunc_varimax_reorder(eof_vmax)
```

https://www.ncl.ucar.edu/Applications/eof.shtml #5

EOF: Principal Oscillation Pattern (POP) Analysis

- uses EOFs and much more!
- **http://www.ncl.ucar.edu/Applications/prn_osc_pat.shtml**

Gehne, M. (2014): Irregularity and decadal variation in ENSO:
a simplified model based on Principal Oscillation Patterns.
Climate Dynamics: Dec 2014, Volume 43, 12, pp 3327-3350
http://dx.doi.org/10.1007/s00382-014-2108-6

----

von Storch, H. et al (1995): Principal Oscillation Patterns: A Review.
J. Climate, **8**, 377–400.
http://dx.doi.org/10.1175/1520-0442(1995)008<0377:POPAR>2.0.CO;2

Compositing

```
 t1  = (/ 15, 37, 95, 88,90 /)          ; cd_calendar, ind, get1Dindex
 t2  = (/  1, 22, 31, 97, 100, 120/)

 f   = addfile("01-50.nc",  "r")
T1 = f->T(t1,:,:,:)                        ; T(time,lev,lat,lon)
T2 = f->T(t2,:,:,:)
                                           ; composite averages
T1avg = dim_avg_n_Wrap(T1, 0) ; (lev,lat,lon)
T2avg = dim_avg_n_Wrap(T2, 0)

 Tdiff  = T2avg                           ; trick to transfer meta data
 Tdiff  = T2avg - T1avg                   ; difference
 Tdiff@long_name = T2@long_name + ": composite difference"
```

Also use **coordinate subscripting**: let "time" have units *yyyymm*

```
  t1    = (/ 190401, 191301, 192001, ……, 200301/)
 T1    = f->T({t1},:,:,:))
```

Compositing: temporal

Compositing: combining data from **different** periods that satisfy some **common criteria**

```
                                        ; Climate Prediction Center
fnam  = "ElNino_LaNina.txt"             ; contains seasonal SST anomalies
nrow  = numAsciiRow(fnam)
data  = readAsciiTable(fnam, 13, "float", 2)   ; ncol=13,  nskip=2
year  = data(:,0)
sea     = data(:,1)                            ; DJF seasonal values
nyren = ind(sea .gt. 0.5)                      ; indices for El Nino >  0.5
nyrla  = ind(sea .lt.-0.5)                     ;                La Nina < -0.5
YYYY01_en = year(nyren)*100+1                  ; January of El Nino year
YYYY01_la  = year(nyrla)*100+1                 ; January of La Nina years
```

Compositing: temporal

```
f       = addfile(“air.sig995.mon.mean.nc","r")  ; near surface temperatures
YYYYMM     = cd_calendar(f->time, -1)          ; all times on ‘f’
ien      = get1Dindex(YYYYMM, YYYY01_en)  ; indices of YYYYMM
ila      = get1Dindex(YYYYMM, YYYY01_la)

ten = short2flt(f->air(ien,:,:))               ; [time | 21] x [lat | 91] x [lon | 180]
tla  = short2flt(f->air(ila,:,:))              ; [time | 19] x [lat | 91] x [lon | 180]

ten_avg = dim_avg_n_Wrap(ten,0)   ; [lat | 91] x [lon | 180]
tla_avg  = dim_avg_n_Wrap(tla,0)

tdif    = ten_avg - tla_avg                    ; temp range (El Nino – La Nina)
copy_VarCoords(ten_avg, tdif)
```

Composite: Result