Tutorial 4: NCL part 2: Graphics

The NCAR Command Language (NCL) is a programming language and a powerful graphic lab

The NCAR Command Language (NCL) is an open source, free, interpreted programming language, specifically designed for the access, analysis, and visualization of data.
http://www.ncl.ucar.edu/

NCL has many features common to modern programming languages, including types, variables, operators, expressions, conditional statements, loops, and functions and procedures. An NCL Mini Reference Manual describing basic language features may be downloaded at:
http://www.ncl.ucar.edu/Document/Manuals/

This Mini Graphics Manual includes topics on setting up your NCL environment, high level graphical interfaces, color, vectors, contours, workstations, page maximization, maps and more.

For actual script examples and sample plots, please visit one of the following sites:
http://www.ncl.ucar.edu/Applications/
http://www.ncl.ucar.edu/Document/Manuals/Getting_Started/More

All links:

NCL website: http://www.ncl.ucar.edu/

General documentation: http://www.ncl.ucar.edu/Document/

Reference Manual (online): http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/

Language Manual (pdf): http://www.ncl.ucar.edu/Document/Manuals/language_man.pdf

Graphics Manual (pdf): http://www.ncl.ucar.edu/Document/Manuals/graphics_man.pdf

Most of the basic examples: http://www.ncl.ucar.edu/Applications/

Generic script snapshot:

In general, a script has the following characteristics: (1) load the libraries containing the high-level graphical interfaces with the **load** command. By convention, this occurs before the **begin** statement. (2) Read in the data. (3) Conduct data processing (optional). (4) Open a workstation. (5) Choose a color table (optional). (6) Create a resource variable to which various graphical options (resources) may be assigned as attributes (if plot modifications are desired). (7) Invoke the appropriate graphical interface passing in the workstation, data, and resources.

```
;*****************************************
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/shea_util.ncl"

;*****************************************

begin

;*****************************************
in = addfile("myfile.nc","r"); pointer to file
t = in->T ; read in data
;*****************************************
; create plot
;*****************************************
wks = gsn_open_wks("ps","ce") ; open ps file in a file named ce.ps
; choose colormap
gsn_define_colormap(wks,"BlAqGrYeOrRe")
res = True ; resource varb
res@cnFillOn = True ; turn on color
res@cnLinesOn = False ; no cn lines
res@cnLevelSpacingF = 0.5 ; cn spacing
res@gsnSpreadColors = True ; full colors
res@lbAutoLabelStride = True ; nice lb labels
plot = gsn_csm_contour_map_ce(wks,t,res)

end
```

The default behavior of the graphical interfaces is to draw the plot and advance the frame. These terms will be described in detail later. This default can be changed.

gsn and gsc_csm Interfaces:

gsn functions

The generic interfaces (gsn) are functions and procedures that create basic x-y, contour, streamline, and vector plots. Generally, default settings are used, but the user may readily change these

settings. A list of these interfaces is listed in Appendix B. Example plots and a tutorial on how to use these interfaces is at:

http://www.ncl.ucar.edu/Document/Manuals/Getting_Started/

The examples on this site contain a line-by-line description of various graphical options.exit ; This stops the program here, really usefull to debug

gsn_csm functions

These high-level interfaces emulate the graphical style of figures appearing in the *J. of Climate* (June, 1998) special issue focusing on the Climate System Model (CSM). While the gsn_csm interfaces were designed for a specific purpose, many users prefer them over the generic interfaces. The reason is that they automatically perform tasks like adding color label bars, which must be explicitly added when using the generic gsn interfaces. They also will use a variable's long_name and units attributes (if available) to label a plot. The long_name will be placed in the upper left corner and the units in the upper right corner. Other features include the addition of lat/lon labels of the form "30N/120E" on cylindrical equidistant and polar projection plots, and special labels on pressure height plots. Appendix B contains a list of these interfaces. You can view many examples produced by these interfaces at: http://www.ncl.ucar.edu/Applications/

If you use these functions you will need to load the related libraries at the beginning of the script:

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
```

The gsn_csm plot interfaces may access information required by the CCSM netCDF convention. For example, if the data has a "units" or "long_name" attribute, it will be used to automatically label the plot. (figure 1a).

Units attribute of coordinate variable:

The latitude and longitude coordinate variables should have (but not limited to) attributes in one of the following forms:

"degrees_north" "degrees_east"

"degrees-north" "degrees-east"

"degree_north" "degree_east"

"degrees north" "degrees east"

"degrees_N" "degrees_E"

"Degrees_north" "Degrees_east"

If the coordinate variable does not conform to one of the above names, you will receive an annoying error message in the form of:

(0) is_valid_lat_ycoord: Warning: The units attribute of the Y

coordinate array is not set to one of the allowable units values

(i.e. 'degrees_north'). Your latitude labels may not be correct.
As with non-conforming named dimensions, it is simple to add the appropriate attribute to the coordinate variable:

*x***&**lat**@**units = "degrees_north"
The **&** symbol is used to access the coordinate variable, and the **@** symbol is used to access the attribute. These symbols are described in greater detail in the Mini-Reference Manual (see front cover for download location).

1st step: Open a workstation

Opening a workstation is required prior to the creation of one or more plots. The workstation is simply the location where the graphical instructions are sent (e.g. to a window or postscript file). The workstation is given a name. This name becomes part of the output filename or title of X11 window. Users may have many workstations opened at the same time, although most only open one. Only one colormap can be associated with each workstation.

There are six types of workstations: ncgm (NCAR computer graphics metafile), ps (postscript), eps (encapsulated postscript, contains a bounding box), epsi (encapsulated postscript with a bitmap preview), pdf, and X11 window. Examples:

wks = **gsn_open_wks**("pdf","my_figoutput_file")
wks = **gsn_open_wks**("ps","my_figoutput_file")

The text to the left of the equal sign is a variable and can therefore be arbitrarily named.

2nd step: Setting resources

The graphic programming part in NCL includes lots of resources (depending on the type of plots you want to produce). The attributes of these resources (defined by the @) will control several aspects of the plots. They are mainly Boolean resources (True or False will activate/mute them) but can also define the contour (strides) or others.

Note that *res* is a user-defined variable. It is wise to create separate variables for resources to be passed to different types of interfaces (e.g. *con_res*, *vec_res*, *xy_res*). Vector resources sent to a contour routine will cause warning messages. The following code snippet creates the variable *res* which is of type logical. Attributes associated with *res* are then assigned.

res = True ; first define the resource to be true, the name does not matter

*res***@tiMainString** = "my title" ; here you setup the main title of the plot

*res***@cnFillOn** = True ; produces colored shading

The resource variable is always the last argument in the graphical interface calling sequence.

plot = **gsn_csm_contour**(*wks,data,res*) ; data is the 2D matrix to plot

plot2 = **gsn_xy**(*wks,data,res*) ; data here is a time serie

The variable to the left of the equal sign is of type graphic. The name is arbitrary

Drawing and advancing the frame:

res=True

res@gsnDraw=True ; this will draw the plot

res@gsnFrame=True ; and then advance the frame

by default both resources are defined by True (so you don't need to mention them)

However you can turn them off:

res@gsnDraw=False

res@gsnFrame=False

In this case nothing is plotted

If you then want to plot and advance the frame you have to add these commands at the end (after calling the gsn procedures):

plot = **gsn_csm_contour**(*wks,data,res*) ; plot is the name of your graphical object, data is the 2D ;matrix to plot and res is the associated ressource

draw(plot) ; draw the plot

frame(wks); advance the frame

Ressource types:

am	Annotation manager	**pm**	plot manager	**vf**	vector field
cn	contour	**pr**	primitive	**vc**	vectors
ca	coordinate arrays	**sf**	scalar field	**vp**	viewport
gs	graphical style	**ti**	title	**wk**	workstation
lb	labelbar	**tm**	tickmarks	**ws**	workspace
lg	legend	**tx**	text	**xy**	xy plot
mp	maps	**tr**	transform		

The first two letters of the resource identify what type of resources it is (see former table).
Obviously, if you want to plot a time serie (controlled by xy resources) the maps resources will not be ignored.

In the following, find few examples (based) to illustrate how to use the resources and the various gsn procedures (to produce maps, contours, vectors, 1D plots etc etc).
The most common resources are displayed in the Appendix.
Note that all resource details are available online at:
http://www.ncl.ucar.edu/Document/Graphics/Resources/

Example1: Plot a color contour map (single plot)

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/shea_util.ncl"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Read Netcdf file and display informations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

begin

;;;;;;;;;;;;;;;;;;;;;;;;;
; Read the file
;;;;;;;;;;;;;;;;;;;;;;;;;
; First define a string which defines the file location
fileinobs="/home/caminade/Obs/CRUTS2.1/pr_1m_196101_200012_CRUTS2.1.nc"
print("Reading  "+fileinobs+"")  ;This  will  print  on  screen:  Reading
;/home/caminade/Obs/CMAP/pr_1m_197901_200707_CMAP.nc
f=addfile(fileinobs,"r") ; use the addfile function to point out the file
in read mode
fld = f->pr ; Read the variable pr in the netcdf file and assign it to the
;variable fld
fld = fld/30. ; mm/month to mm/day

; Then to print information about this variable:
fld@_FillValue=-9999. ; change the default missing value
printVarSummary(fld) ; VERY usefull function, this will print in the linux
;terminal various informations

;;;;;;;;;;;;;;;;;;;;;;;;;
; Read dimensions
;;;;;;;;;;;;;;;;;;;;;;;;;

dime=dimsizes(fld) ; Assigne the matrix dimension to fld
dtime=dime(0) ; first element of the dime array is the time dimension
dlat=dime(1) ; 2nd element of the dime array is the latitude dimension
dlon=dime(2) ; 3rd element of the dime array is the longitude dimension
delete (dime) ; delete the dime variable in memory
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Compute seasonal and annual mean with the related functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; compute seasonal mean
season=(/"DJF", "MAM","JJA","SON"/) ; define typical seasons, type string
fldseas=   month_to_seasonN(fld(time|:,lat|:,lon|:),season)   ;   use   the
month_to_seasonN function , time must be a multiple of 12
printVarSummary(fldseas) ; print informations

; compute annual mean
fldannual=    month_to_annual(fld(time|:,lat|:,lon|:),0)    ;    use    the
month_to_annual  function  ,  time  must  be  a  multiple  of  12,  0  means
;cumulative, 1 average
printVarSummary(fldannual) ; print informations

; compute the min and max for each season with a do loop and using the min
; and max functions

do nseas=0,dimsizes(season)-1

      fld_max=max(fldseas(nseas,:,:,:))
      fld_min=min(fldseas(nseas,:,:,:))
      print("Global   rainfall   for   "+season(nseas)+"   max="+fld_max+"
min="+fld_min+"")

end do


; compute annual and seasonal mean climatology
fldannualclim=dim_avg_Wrap(fldannual(lat|:,lon|:,year|:))
fldseasclim=dim_avg_Wrap(fldseas(season|:,lat|:,lon|:,time|:))

printVarSummary(fldannualclim)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Start the plot
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


  print("Plotting")

  figout="figout_single_map" ; define the name of the figure to be produced

  type = "ps" ; properties of the workstation
  type@wkOrientation = "portrait"
  wks = gsn_open_wks(type,figout) ; define the workstation

  gsn_define_colormap(wks,"prcp_3") ; define the color palette


  res=True
  res@gsnFrame=False    ; do not draw by default then you need to mention
;draw(plot) later
  res@gsnDraw=False     ; do not advance the frame then you need to mention
;frame(wks) later
  res@gsnMaximize =False  ; This ressource maximize the size of the plot in
;the figure
```

```
  res@gsnAddCyclic=False  ; This ressource needs to be setup as True if the
;dataset you work on is not global (like a RCM for example)

  res@gsnSpreadColors       = True   ; This divides the color palette in
;equally space segments (need to be setup to False if you define the
contour explicitely
  res@gsnLeftString     = ""  ; This divides the string on the top left of
;the plot by default the unit attribute of the variable here setup do blank
  res@gsnRightString    = "" ; same for the right (variable name attribute
;I think)
  res@gsnCenterString   = "" ; same for the center
  res@gsnZonalMean    = True ; this one by default is setup to False, add a
;zonal average on the right hand side of the plot

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; ContourPlot resources

 res@cnFillOn=True ; activate the color shading
 res@cnLevelSelectionMode = "ManualLevels" ; setup the contours as Manual

 res@cnMinLevelValF = 0. ; select the minimum value of the contours
 res@cnMaxLevelValF = 100. ; select the maximum value of the contours
 res@cnLevelSpacingF = 10. ; select the stride of the contours

 res@cnRasterModeOn=False   ; If True shading (plot the grid boxes) False
;filling (interpolation)
 res@cnLinesOn=False ; enable/disable the contour lines
; res@cnLineLabelsOn = False ; enable/disable the contour lines labels (the
values)
 res@cnLineLabelBackgroundColor = "white" ; if labels True then write the
;value on a white square
 res@cnInfoLabelString ="" ; If you plot the lines, display the values and
;the stride in the lower left corner of the plot

; MapPlot resources

 res@mpOutlineBoundarySets = "National" ; Add the national boundaries

 res@mpMinLonF                    = -40. ; setup the domain to plot lonmin,
;lonmax, latmin, latmax here Africa for example
 res@mpMaxLonF                = 80.
 res@mpMinLatF                = -40.
 res@mpMaxLatF                = 40.

 res@mpFillOn=True ; Apply fill for the maps
 res@mpFillColors     = (/-1,-1,-1,-1/) ; setup everything to transparent
;otherwise you can plot the ocean, land river with a specific colo
 res@mpPerimOn = False ; add the perimeter
 res@mpGeophysicalLineThicknessF = 2 ; thickness of the continet line


 res@lbLabelBarOn =True ; setup the labelbar
 res@lbTitleString="mm/day" ; label of the labelbar
 res@lbTitleFontHeightF=0.018 ; Font Height of the laber bar Title
 res@lbLabelFontHeightF=0.016 ; Font Height of the labels of the labelbar
 res@lbLabelStride=2 ; stride, display labelbar labels every 2 values
 res@lbTitlePosition="Bottom" ; drop the label bar at the bottom
 res@pmLabelBarOrthogonalPosF = -0.005 ; shift the label bar southward

 res@tiMainFontHeightF  = .025 ; setup the Main Font of the title
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

res@tiMainString       ="Annual Rainfall Climatology CRU2: 1961-2000"
plot=gsn_csm_contour_map_ce(wks,fldannualclim,res) ; plot the fiels (1st
;arg workstation, 2nd arg variable to plot must be a 2D matrix, res the
;related ressources`

draw(plot) ; draw the plot
frame(wks) ; advance the frame

end
```

Example available on geog2 at:
/home/caminade/Dave/NCL_tutorial/examples/ plot_single_map.ncl

Example2: Plot a color contour map (panel plot)

The Main program core is the same than the former one. The graphics part is modified (modifications are highlighted as bold):

```
  print("Plotting")

  figout="figout_panelplot_map" ; define the name of the figure to be
;produced

  type = "ps" ; properties of the workstation
  type@wkOrientation = "portrait"
  wks = gsn_open_wks(type,figout) ; define the workstation

  gsn_define_colormap(wks,"prcp_1") ; define the color palette
  plot = new(4,graphic) ; define the plot graphic object with 4 viewports


  res=True
  res@gsnFrame=False
  res@gsnDraw=False
  res@gsnMaximize =False
  res@gsnAddCyclic=False
  res@gsnSpreadColors     = True
  res@gsnLeftString     = ""
  res@gsnRightString    = ""
  res@gsnCenterString   = ""
  res@gsnZonalMean    = False

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; ContourPlot resources

 res@cnFillOn=True ; activate the color shading
 res@cnLevelSelectionMode = "ManualLevels" ; setup the contours as Manual

 res@cnMinLevelValF = 1. ; select the minimum value of the contours
 res@cnMaxLevelValF = 14. ; select the maximum value of the contours
 res@cnLevelSpacingF = 1. ; select the stride of the contours
```

```
 res@cnRasterModeOn=False
 res@cnLinesOn=False ; enable/disable the contour lines
 res@cnLineLabelsOn = False
 res@cnLineLabelBackgroundColor = "white"
 res@cnInfoLabelString =""

; MapPlot resources

 res@mpOutlineBoundarySets = "National" ; Add the national boundaries

 res@mpMinLonF                = -40.
 res@mpMaxLonF                = 80.
 res@mpMinLatF                = -40.
 res@mpMaxLatF                = 40.

 res@mpFillOn=True ; Apply fill for the maps
 res@mpFillColors    = (/-1,-1,-1,-1/)
 res@mpPerimOn = False ; add the perimeter
 res@mpGeophysicalLineThicknessF = 2

 res@lbLabelBarOn =False ; setup the labelbar
 res@tiMainFontHeightF  = .025 ; setup the Main Font of the title



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Plot for all seasons

do nseas=0,dimsizes(season)-1
      res@tiMainString       = "CRU2 clim "+season(nseas)+" 1961-2000"
      plot(nseas)=gsn_csm_contour_map_ce(wks,fldseasclim(nseas,:,:),res)
end do

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Advance the panel plot

  resP                  = True                    ; modify the panel plot
  resP@gsnMaximize      = False                  ; use full page
  resP@gsnPanelLabelBar=True  ; label of the labelbar in the panel plot
  resP@lbTitleString="mm/day" ; Font Height of the laber bar Title
  resP@lbTitleFontHeightF=0.018  ;  Font  Height  of  the  labels  Title  of  the
;labelbar
  resP@lbLabelFontHeightF=0.016 ; Font Height of the labels of the labelbar
  resP@lbLabelStride=2 ; stride, display labelbar labels every 2 values
  resP@lbTitlePosition="Bottom" ; drop the label bar at the bottom
  resP@pmLabelBarOrthogonalPosF = -0.005 ; shift the label bar southward

  gsn_panel(wks,plot,(/2,2/),resP)               ; now draw as one plot in
;a panel plot 2 lines per 2 column
  ; Note that gsnDraw and gsnFrame for the panel plot are defined to True
;by default

end
```

Example available on geog2 at:
/home/caminade/Dave/NCL_tutorial/examples/ plot_panelplot_map.ncl

Example3: Plot a 1D index

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/shea_util.ncl"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Read Netcdf file and plot time serie
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

begin

;;;;;;;;;;;;;;;;;;;;;;;;;
; Read the file
;;;;;;;;;;;;;;;;;;;;;;;;;
; First define a string which defines the file location
fileinobs="/home/caminade/Obs/CRUTS2.1/tas_1m_196101_200012_CRUTS2.1.nc"
print("Reading  "+fileinobs+"")  ;This  will  print  on  screen:  Reading
;/home/caminade/Obs/CMAP/pr_1m_197901_200707_CMAP.nc
f=addfile(fileinobs,"r") ; use the addfile function to point out the file
in read mode
fld = f->tas ; Read the variable pr in the netcdf file and assign it to the
;variable fld

; Then to print information about this variable:
fld@_FillValue=-9999. ; change the default missing value
printVarSummary(fld) ; VERY usefull function, this will print in the linux
;terminal various informations

;;;;;;;;;;;;;;;;;;;;;;;;;
; Read dimensions
;;;;;;;;;;;;;;;;;;;;;;;;;

dime=dimsizes(fld) ; Assigne the matrix dimension to fld
dtime=dime(0) ; first element of the dime array is the time dimension
dlat=dime(1) ; 2nd element of the dime array is the latitude dimension
dlon=dime(2) ; 3rd element of the dime array is the longitude dimension
delete (dime) ; delete the dime variable from memory

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Compute seasonal and annual mean with the related functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; compute seasonal mean
season=(/"DJF", "MAM","JJA","SON"/) ; define typical seasons, type string
fldseas=   month_to_seasonN(fld(time|:,lat|:,lon|:),season)   ;   use   the
month_to_seasonN function , time must be a multiple of 12
printVarSummary(fldseas) ; print informations

; compute annual mean
fldannual=    month_to_annual(fld(time|:,lat|:,lon|:),0)    ;    use    the
month_to_annual  function  ,  time  must  be  a  multiple  of  12,  0  means
;cumulative, 1 average
printVarSummary(fldannual) ; print informations

; compute annual and seasonal mean climatology
fldannualclim=dim_avg_Wrap(fldannual(lat|:,lon|:,year|:))
```

```
fldseasclim=dim_avg_Wrap(fldseas(season|:,lat|:,lon|:,time|:))

printVarSummary(fldseas)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; compute the index
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define a domain range
latmin=10.
latmax=20.
lonmin=-16
lonmax=20.

if  isMonotonic(fldseas&lat).eq.-1  then  ;  if  the  latitude  is  sorted  by
;descending order invert it to ascending
fldseas=fldseas(:,:,::-1,:)
end if

if any(fldseas&lon.gt.181) then ; if any longitude value is defined between
;0 and 360 flip the lon between -180 to 180
fldseas=lonFlip(fldseas)
end if

fldseas_small=fldseas(:,:,{latmin:latmax},{lonmin:lonmax})   ;   subset   the
;domain to compute the average
rad    = 4.0*atan(1.0)/180.0 ; transformation degree radian
clat   = cos(fldseas_small&lat*rad) ; compute the cosine of the latitude
index=wgt_areaave(fldseas_small,clat,1.0,0) ; perform the weighted average
;(coslat)
index!0="nseas" ; name the dimension 1 to nseas
index!1="year"; name the dimension 2 to year

time=ispan(1961,2000,1) ; the time vector


printVarSummary(fldseas_small)
printVarSummary(index)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Start the plot
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


  print("Plotting")

  type = "ps"
  type@wkOrientation = "portrait"
  wks = gsn_open_wks(type,"serie_1D")

  res                     = True
  res@gsnFrame            = False
  res@gsnDraw             = False


  res@trYMinF=18.0 ; Ymin Value
  res@trYMaxF=35.  ; Ymax Value
  res@trXMinF=1960 ; Xmin Value
  res@trXMaxF=2005 ; Xmax Value

  res@tiXAxisString = "Years" ; X axis labels
  res@tiYAxisString = "Temp (C)"; Y axis labels
```

```
  res@xyMonoDashPattern=True
  res@xyLineColors      = (/"blue","green","red","orange"/) ; colors of each
;lines here 4 seasons as index is 4 seasons x 40 years
  res@xyLineThicknesses = (/4.0,4.0,4.0,4.0/) ; thickness of each line

  res@pmLegendDisplayMode    = "Always"     ; activate the legend
  res@pmLegendSide           = "Top"        ; reference to place the legend
  res@pmLegendParallelPosF   = 0.15         ; move units right with respect
;to the reference
  res@pmLegendOrthogonalPosF = -1.1     ; move units down with respect to
;the reference

  res@pmLegendWidthF         = 0.12   ; width of the legend box
  res@pmLegendHeightF        = 0.14   ; Height of the legend box
  res@lgLabelFontHeightF     = .016   ; Font Height of the legend
  res@lgPerimOn                 = True    ; add a perimeter box around the
;legend

  res@xyExplicitLegendLabels = season ; labels of each lines here related
;to the season variable
  res@tiMainString                  =   "Sahel   lat=["+latmin+"-"+latmax+"]
;lon=["+lonmin+"-"+lonmax+"] "    ; Main title

  plot = gsn_csm_xy (wks,time,index,res) ; plot the time serie note that
;the rightmost dim of index must be the same than time


  draw(plot)
  frame(wks)


end


Example available on geog2 at:
/home/caminade/Dave/NCL_tutorial/examples/ plot_index1D.ncl
```

Example4: Plot whatever you want

There are several plot types available in NCL, including styles (plume, box and whiskers, vector maps etc etc). You will find basic example per topics (XY plots, tickmarks, legends, contour effects etc etc) At this address (topic plot types and plot techniques):

http://www.ncl.ucar.edu/Applications/

I strongly advise to bookmark this page in your favourite browser.

Now try examples and plot your own figures!

Appendix (detailed here)

Appendix A: Common Resources

These are a list of some common resources by topic. They are by no means exhaustive. The term dynamic means that the value is determined by NCL. For an exhaustive list of resources, see:

http://www.ncl.ucar.edu/Document/Graphics/Resources/

Axis - http://www.ncl.ucar.edu/Document/Graphics/Resources/tr.shtml

Name	Function	Default	Example
`trYReverse` `trXReverse`	reverse x or y axis	False	True
`trYMinF` `trXMinF`	set minimum of x or y axis	0.0	3
`trYMaxF` `trXMaxF`	set maximum of x or y axis	1.0	900
`trYLog` `trXLog`	turn on/off log axis	False	True

Contour - http://www.ncl.ucar.edu/Document/Graphics/Resources/cn.shtml

Name	Function	Default	Example
`cnFillOn`	turn on/off color filled contours	False	True
`cnLinesOn`	turn on/off contour lines	True	False
`cnFillMode`	set type of contour fill	"AreaFill"	"RasterFill"
`cnLevelSelectionMode`	control contour levels	"AutomaticLevels"	"ExplicitLevels" "ManualLevels"
`cnMinLevelValF` `cnMaxLevelValF`	set minimum or maximum contour level	dynamic	5 35
`cnLevelSpacingF`	set contour spacing	dynamic	2
`cnLevels`	set contour elvels when cnLevelSelectionMode is "ExplicitLevels"	dynamic	(/3,5,7,9,10,45/)
`cnLineThicknessF`	set thickness of contour lines	1.0	2.0
`cnFillPatterns`	set pattern fills	"SolidFill"	(/1,3,-1/) (-1 is transparent
`cnInfoLabelOn`	turn on/off the contour info label	True	False

Labelbars - http://www.ncl.ucar.edu/Document/Graphics/Resources/lb.shtml

Name	Function	Default	Example
`cnFillOn`	turn contour fill on/off	False	True
`cnFillMode`	set contour fill mode	"AreaFill"	"RasterFill"
`cnLabelBarEndStyle`	set style for end labels	"IncludeOuterBoxes"	"ExcludeOuterBoxes"
`gsnSpreadColors`	span full range of colormap	False	True
`gsnSpreadColorStart`	begin colormap at particular index	2	46
`gsnSpreadColorEnd`	end colormap at particular index	*ncolors*-1	89
`lbLabelBarOn`	turn on/off the labelbar	True for gsn_csm interfaces	False
`lbOrientation`	set orientation of labelbar	horizontal for gsn_csm interfaces	"vertical"
`lbLabelAutoStride`	automatically pick nice labelbar label stride	False	True
`lbTitleOn`	turn on/off a labelbar title	False	True
`lbTitleString`	set labelbar title	Null	"m/s"
`lbLabelAlignment`	st where the labelbar label is oriented wrt to the color boxes	"ExternalEdges"	"BoxCenters"
`pmLabelBarOrthogonalPosF`	moves the labelbar orthogonally to its position. For a horizontal labelbar, this is up and down.	N/A	-0.03
`pmLabelBarParallalPosF`	moves the labelbar perpendicularly to its position. For a vertical labelbar, this is left and right.	N/A	-0.01
`pmLabelBarWidthF`	set the width of the labelbar	set for the user in the gsn_csm interfaces	
`pmLabelBarHeightF`	set the height of the labelbar	set for the user in the gsn_csm interfaces	

GSN - http://www.ncl.ucar.edu/Document/Graphics/Resources/gsn.shtml

Name	Function	Default	Example
`gsnAddCyclic`	turn on/off the addition of a cyclic point to the longitude coordinate values	True for data that has 1D coordinate variables	False
`gsnCenterString`	see figure 1a	N/A	"string here"
`gsnDraw`	draw the plot	True	False
`gsnFrame`	advanced the frame (page)	True	False
`gsnLeftString`	see figure 1a	long_name (if exists) in gsn_csm interfaces	"Salinity"
`gsnMaximize`	maximizes plot and rotates to landscape if necessary	False	True
`gsnPanelFigureStrings`	add a series of strings to the upper left corner of each plot in a panel	N/A	(/"a","b","c"/)
`gsnPanelLabelBar`	turn on/off a common labelbar in a panel plot	False	True
`gsnRightString`	see figure 1a	units (if exists) in gsn_csm interfaces	"ppm"
`gsnScalarContour`	force vector/scalar gsn_csm interfaces to draw vectors over the scalar field	False	True
`gsnSpreadColors`	span full range of colormap	False	True
`gsnSpreadColorStart`	begin colormap at particular index	2	46
`gsnSpreadColorEnd`	end colormap at particular index	*ncolors*-1	89
`gsnXYBarChart`	changes an x-y line into a bar chart	False	True
`gsnXRefLine`	add a vertical reference line to a plot	None	1.0
`gsnXRefLineColor`	change color of X reference line	foreground color	"green"
`gsnYRefLine`	add a horizontal reference line to a plot	None	0.0
`gsnYRefLineColor`	change color of Y reference line	foreground color	"blue"

Legends - http://www.ncl.ucar.edu/Document/Graphics/Resources/lg.shtml

Name	Function	Default	Example
pmLegendWidthF	set width of a legend	dynamic	0.6
pmLegendHeightF	set height of a legend	dynamic	0.3
lgTitleOn	turn on legend title	False	True
lgTitleString	set title string	N/A	"Profiles"
lgOrientation	set orientation of the legend	"horizontal"	"vertical"
lgPerimOn	turn the legend perimeter on/off	True	False
xyExplicitLegendLabels	change default legend labels	N/A	(/"a","b"/)
pmLegendOrthgonalPosF	adjust the legend orthogonally	N/A	-0.03
pmLegendParallelPosF	adjust the legend perpendicularly	N/A	0.2

XY curves - http://www.ncl.ucar.edu/Document/Graphics/Resources/xy.shtml

Name	Function	Default	Example
xyDashPatterns	set line pattern	solid	(/0,2/) (/"solid","dash"/)
xyLineThicknesses	set line thicknesses	1.0	(/2.0,3.0,4.0/)
xyLineColors	set line colors	foreground color	(/"red","blue"/)
xyMarkLineModes	set whether lines contain markers, lines, or both markers and lines	"Lines"	"Lines" "Markers" "MarkLines"
xyMarkers	set marker styles	asterisk	5
xyMarkerColor	set marker colors	foreground color	"green"
xyMarkerSizeF	set marker size	0.01	0.03

Maps - http://www.ncl.ucar.edu/Document/Graphics/Resources/mp.shtml

The second through fifth resources are to be used when zooming in on a cylindrical equidistant or polar stereographic projection. They are the limits set when using **mpLimitMode** = "LatLon". This resource is set for the user by the high-level plot interfaces. Other projections, such as lambert conformal, require a different limit mode (**mpLimitMode** = "Corners").

Name	Function	Default	Example
mpLimitMode	determine how a map is zoomed in	depends on projection	"LatLon" "Corners"
mpMinLatF	set minimum latitude for map zoom	dynamic	30.
mpMaxLafF	set maximum latitude for map zoom	dynamic	60.
mpMinLonF	set minimum longitude for map zoom	dynamic	-70.
mpMaxLonF	set maximum longitude for map zoom	dynamic	89.
mpFillOn	turn on/off map fill	True for gsn_csm interfaces	False
mpCenterLonF	set center longitude of projection	0	180.
mpDataBaseVersion	set map database resolution	"LowRes"	"MediumRes" "HighRes" (must be downloaded)
mpLandFillColor	set color of land areas	"gray" for gsn_csm interfaces	"brown"
mpOceanFillColor	set color of ocean areas	"transparent"	"SkyBlue"
mpInlandWaterFillColor	set color of inland water areas	"transparent"	"blue"
mpOutlineOn	turn on/off the map outlines	True	False
mpOutlineBoundarySets	set various continental outlines on and off	"Geophysical"	"GeosphysicalAndUSStates" "National"
mpGeophysicalLineThicknessF	set line thickness of map outlines	1.0	2.0
mpGeophysicalLinColor	set color of map outlines	foreground	"red"
mpUSStateLineColor	set color of US state boundaries	foreground	"blue"

Polygons, polylines, polymarkers -
http://www.ncl.ucar.edu/Document/Graphics/Resources/gs.shtml

Name	Function	Default	Example
`gsFillColor`	set fill color for inside of polygon	transparent	"red"
`gsEdgeColor`	set color of the outline of a polygon	none	"black"
`gsEdgesOn`	turn on/off polygon edge	False	True
`gsLineColor`	set polyline color	foreground color	"orange"
`gsLineThicknessF`	set polyline thickness	1.0	2.5
`gsMarkerIndex`	set marker style	asterisk (0)	5
`gsMarkerColor`	set marker color	foreground color	"purple"
`gsMarkerSizeF`	set marker size	0.007	0.014

Appendix B: High-level Graphical Interfaces

gsn generic interfaces

gsn_xy
gsn_y
gsn_contour
gsn_contour_map
gsn_vector
gsn_vector_scalar
gsn_vector_map
gsn_vector_scalar_map
gsn_streamline
gsn_streamline_map
gsn_map

As an example, the following line of code will contour the two-dimensional array *data*:

```
plot = gsn_contour(wks,data,res)
```

gsn_csm interfaces

In the list below, an **_ce** stands for cylindrical equidistant projection. An **_hov** stands for hovmuller d
All other interface names are self-explanatory. As with the gsn generic interfaces, the **gsn_csm** inte
are functions and return a graphical object. Note: many of the interfaces listed below could be place
more than one category.

Contour:
gsn_contour_shade (nice customization of contour filling)
gsn_csm_contour
gsn_csm_contour_map (choose your projection)
gsn_csm_contour_map_ce
gsn_csm_contour_map_polar
gsn_csm_contour_map_overlay (overlay additional contours)

```
plot = gsn_csm_contour(wks,data,res)
```

Streamline:
gsn_csm_streamline
gsn_csm_streamline_map (choose your projection)
gsn_csm_streamline_map_ce
gsn_csm_streamline_map_polar
gsn_csm_streamline_contour_map
gsn_csm_streamline_contour_map_ce
gsn_csm_streamline_contour_map_polar

```
plot = gsn_csm_streamline(wks,u,v,res)
```

Vector:
gsn_csm_vector
gsn_csm_vector_map
gsn_csm_vector_map_ce
gsn_csm_vector_scalar_map
gsn_csm_vector_scalar_map_ce
gsn_csm_vector_scalar_map_polar

```
plot = gsn_csm_vector(wks,u,v,res)
```

Pressure/Height:
gsn_csm_pres_hgt
gsn_csm_pres_hgt_streamline
gsn_csm_pres_hgt_vector

Misc:
gsn_csm_lat_time
gsn_csm_time_lat
gsn_csm_hov
gsn_csm_xy
gsn_csm_y

gsn special interfaces

Polylines: These interfaces add a polyline to an existing plot:

gsn_polyline	(plot coordinates)
gsn_polyline_ndc	(page coordinates)
gsn_add_polyline	(can be paneled, plot coords)

Polymarkers: These interfaces add polymarkers to an existing plot:

gsn_polymarker	(plot coordinates)
gsn_polymarker_ndc	(page coordinates)
gsn_add_polymarker	(can be paneled, plot coords)

Polygons: These interfaces add a polygon to an existing plot:

gsn_polygon	(plot coordinates)
gsn_polygon_ndc	(page coordinates)
gsn_add_polygon	(can be paneled, plot coords)

Text: These interfaces add text to an existing plot:

gsn_text	(plot coordinates)
gsn_text_ndc	(page coordinates)
gsn_add_text	(can be paneled, plot coords)
gsn_create_text	(no coords, use in conjunction with gsn_add_annotation)

Colormaps: These interfaces are used for manipulating color maps. You can view the built-in color maps at:

http://www.ncl.ucar.edu/Document/Graphics/color_table_gallery.shtml

gsn_define_colormap	gsn_merge_colormaps	gsn_draw_colormap
gsn_retrieve_colormap	gsn_draw_named_colors	gsn_reverse_colormap
hlsrgb	hsvrgb	rgbhls
rgbhsv	rgbyiq	yiqrgb

Miscellaneous: These interfaces perform a variety of functions:

gsn_add_annotation	gsn_panel
gsn_attach_plots	gsn_table
gsn_create_labelbar	
gsn_create_legend	
gsn_histogram	
gsn_labelbar_ndc	
gsn_legend_ndc	
gsn_open_wks	

Appendix D: Common Error Messages

Message:

```
(0) check_for_y_lat_coord: Warning: Data either does not contain a
valid latitude coordinate array or doesn't contain one at all
(0) check_for_lon_coord: Warning: Data either does not contain a valid
longitude coordinate array or doesn't contain one at all
```

Solution:

Need to rename x and y dimensions to one of those listed in section 2.6.

Message:

```
(0) is_valid_lat_ycoord: Warning: The units attribute of the Y
coordinate array is not set to one of the allowable units values (i.e.
'degrees_north'). Your latitude labels may not be correct.
(0) is_valid_lat_xcoord: Warning: The units attribute of the X
coordinate array is not set to one of the allowable units values (i.e.
'degrees_east'). Your longitude labels may not be correct.
```

Solution:

Need to add or change the units attributes for the latitude/longitude coordinate arrays (see section 2.6)

Message:

```
(0) gsn_add_cyclic: Warning: The range of your longitude data is not
360. You may want to set gsnAddCyclic to False to avoid a warning
message from the Spline function.
```

Solution:

A cyclic point is added to the data in all gsn_csm high-level map interfaces. If this is inappropriate (e.g. a regional plot), then it is necessary to set **`gsnAddCyclic`** `= False`

Message:

```
(0) warning:_NhlCreateSplineCoordApprox: Attempt to create spline
approximation for Y axis failed: consider adjusting trYTensionF value
warning:IrTransInitialize: error creating spline approximation for
trYCoordPoints; defaulting to linear
```

Solution:

Chances are this has occurred because of an error in the longitude coordinate variable. It is either incorrect or there is a gap in the data.

Message:

```
fatal:ContourPlotDraw: Workspace reallocation would exceed maximum size
16777216 fatal:ContourPlotDraw: draw error fatal:PlotManagerDraw: error
in plot draw fatal:_NhlPlotManagerDraw: Draw error
```

Solution:

The plot of your data is too large for NCL's default 16MB size. You must increase the size by setting:

```
setvalues NhlGetWorkspaceObjectId()
      "wsMaximumSize": 33554432
end setvalues
```