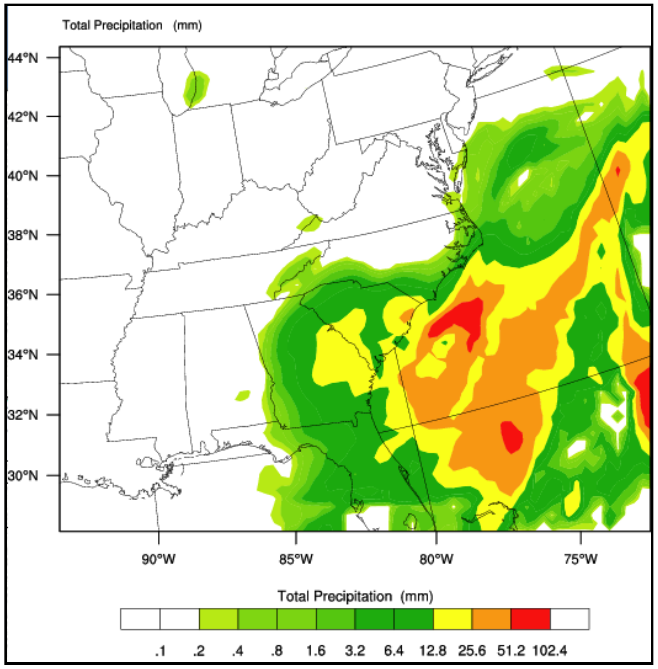
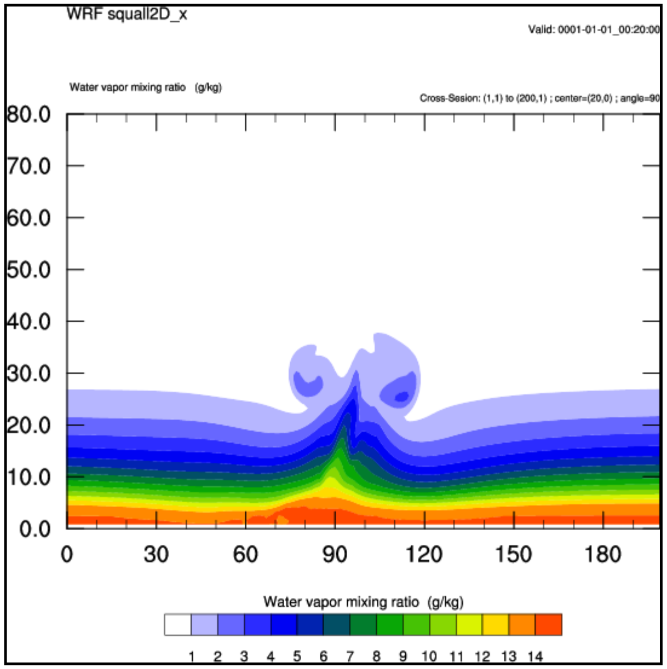
Post-processing WRF-ARW data with the NCAR Command Language

Mary Haley and Cindy Bruyère

12th Annual WRF Users' Event, June 20-24, 2011





The National Center for Atmospheric Research is sponsored by the National Science Foundation.

Goals

- Introduce you to NCL and WRF-NCL
- *Get you familiar with WRF-NCL scripts*
  - *Opening and examining a data file*
  - *Reading and querying variables*
  - *Plotting variables*
- Sneak in tips and information for existing users

Topics

- Overview
- What’s new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

NCL

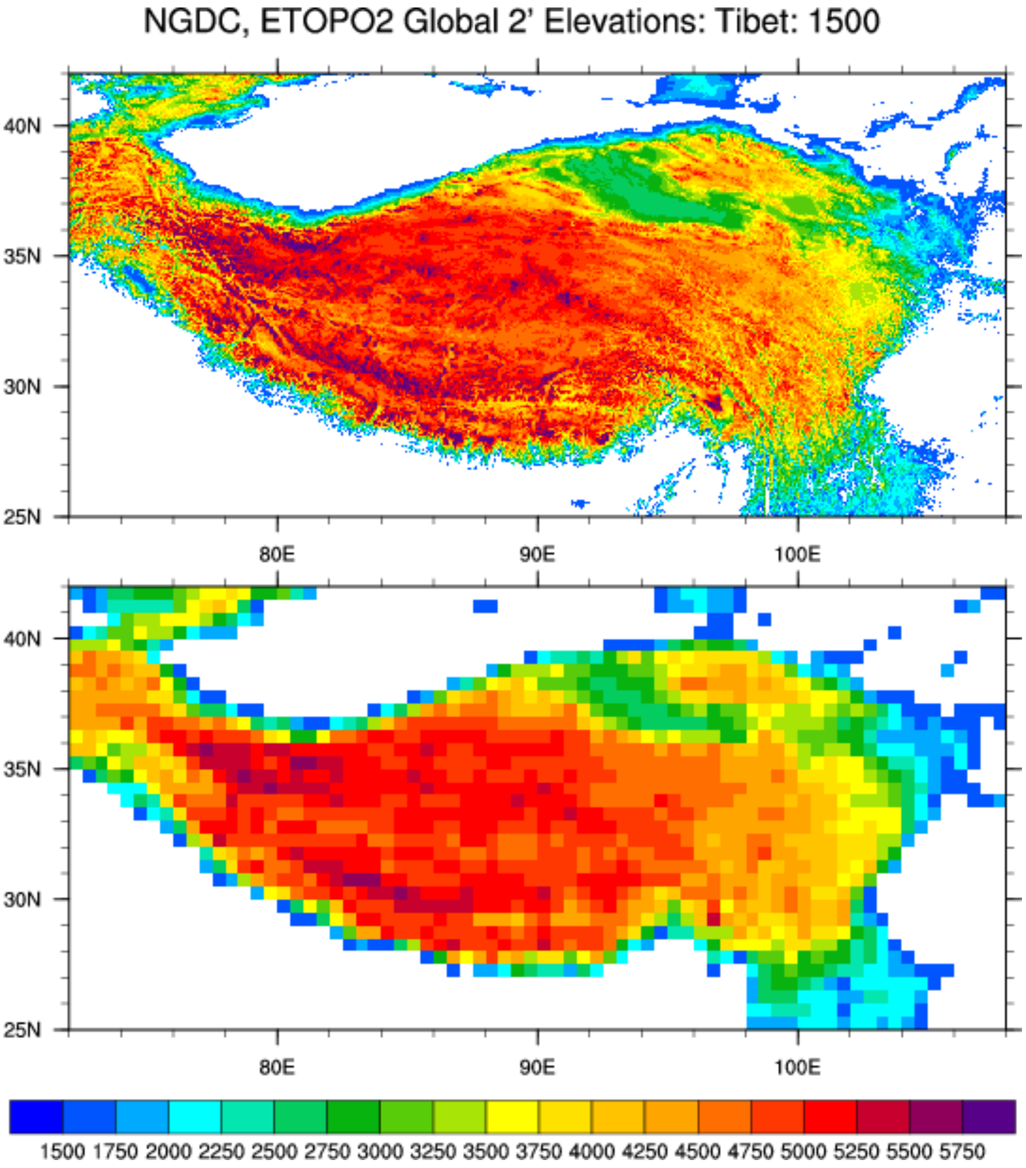
NCAR Command Language

A scripting (interpreted) language tailored for the analysis and visualization of geoscientific data

- Developed in NCAR/CISL in close collaboration with CGD & MMM scientists
- UNIX binaries and source available, **free**
- Extensive NCL website, hundreds of examples
- Hands-on workshops
- Email lists for consulting

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

CGD CISL



What is NCL?

- A scripting language similar to Python or IDL
- Tailored to climate and atmospheric sciences
- Has variable types, “if-then-endif”, “do” loops, arithmetic operators
- F90-like array arithmetic that will ignore missing values
- Can call your own Fortran 77/90 or C routines
- Simple, robust file input/output
- Hundreds of data analysis routines
- Publication-quality graphics that are highly customizable

NCL: File input and output

- Data model based on netCDF model (metadata describes data)
- One function reads all supported data formats:
  - NetCDF, GRIB 1 and 2, HDF4, HDF-EOS2, HDF-EOS5, shapefiles, (new: HDF5)
  - Writes NetCDF and HDF4 (compressed NetCDF too)
- OPeNDAP-enabled client available
- ASCII, binary (read and write)

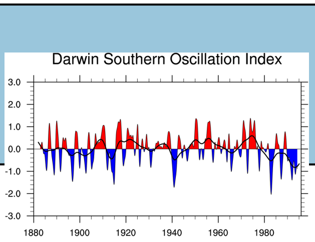
http://www.ncl.ucar.edu/Applications/list_io.shtml

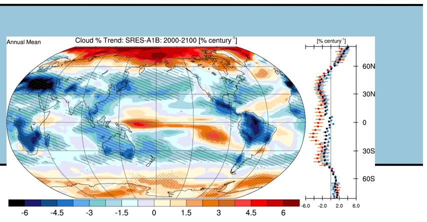
NCL: Data analysis

- Array-based math
- Hundreds of functions
  - WRF-ARW specific functions
  - Spherical harmonics
  - Scalar and vector regridding
  - Vertical interpolation
  - EOFs
- Many tailored to geosciences
- Most automatically handle missing data
- Can call C and Fortran routines - WRAPIT

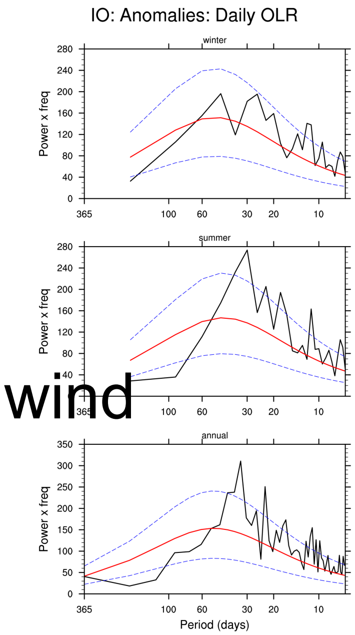
http://www.ncl.ucar.edu/Applications/list_dataP.shtml

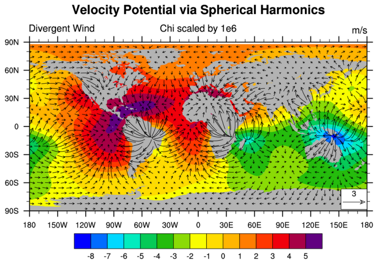
NCL: Visualization

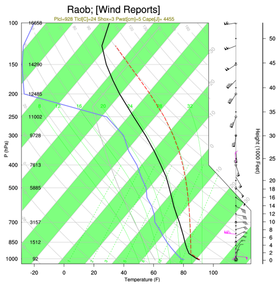


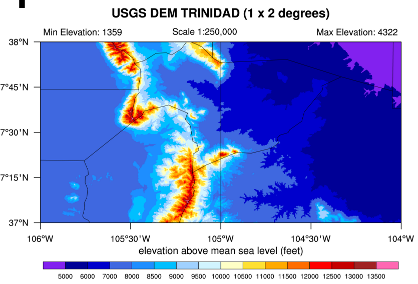


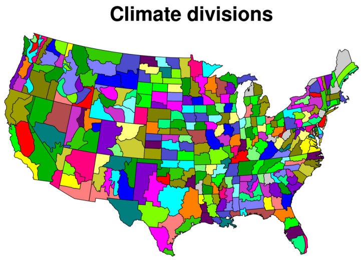
- High-quality and customizable visualizations
- Contours, XY, vectors, streamlines
- Maps with common map projections
- Handles data on regular and irregular grids, triangular meshes
- Specialized scripts for meteograms, skew-T, wind roses, histograms, cross section, panels
- wrf_xxxx functions: simplifies visualization for WRF-ARW data
- Over 1,400 visualization “options”







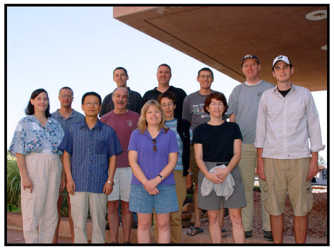


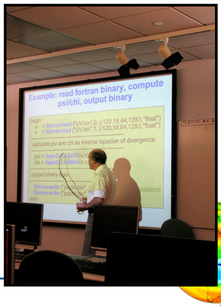


http://www.ncl.ucar.edu/gallery.shtml

NCL Training Workshops

- First training workshop in 2000, 52 so far, 720+ attendees
  - 3-4 local workshops a year
  - One free annual workshop at a UCAR member university
  - One invited international workshop
- Lectures taught by a scientist and a software engineer
- Includes special lecture on various data formats used in geosciences
- Four hands-on labs sessions; students encouraged to bring their own datasets




Example: read fortran binary, compute psi/chi, output binary

Funding to attend NCL Workshop

August 16-19, 2011

University of Wyoming

Laramie, Wyoming

Bryan Shader – host

University of Wyoming

NWSC
NCAR-Wyoming
Supercomputing Center

CISL

NESL

UW and NCAR/CISL will provide travel funds for students and staff from EPSCoR states or minority-serving institutions to attend.

Deadline to apply: July 8, 2011

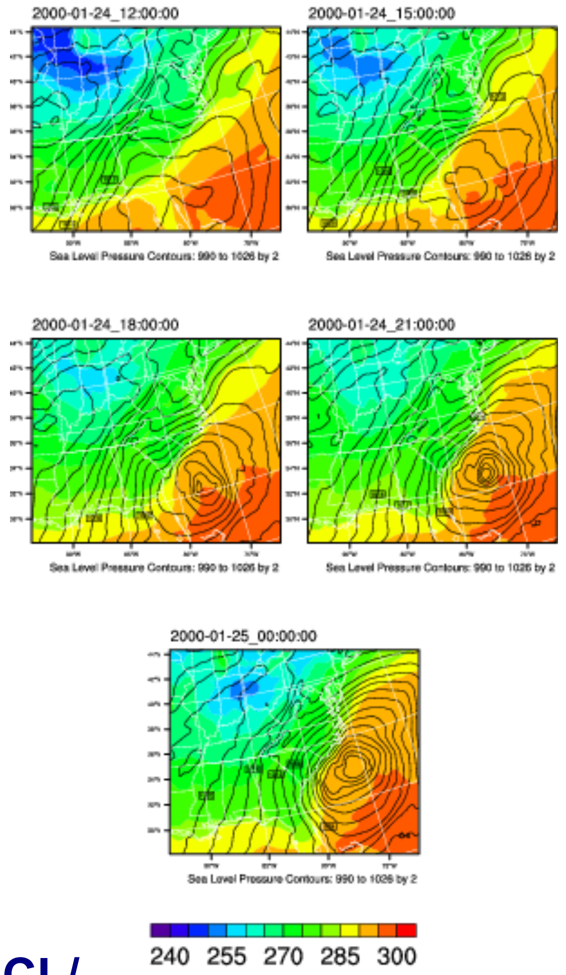
http://www.ncl.ucar.edu/Training/Workshops/

WRF-NCL

A suite of analysis and visualization functions tailored for WRF-ARW model data

- **Included with NCL**
- Developed by scientists in MMM
- Maintained by Cindy Bruyère/MMM and myself
- Functions for calculating basic diagnostics
- Functions for specialized visualizations – precipitation, surface, vorticity, meteograms, helicity, squall, dBZ, etc.
- Website with lots of analysis and visualization examples
- Workshops and tutorials
- Email list for consulting, wrfhelp@ucar.edu

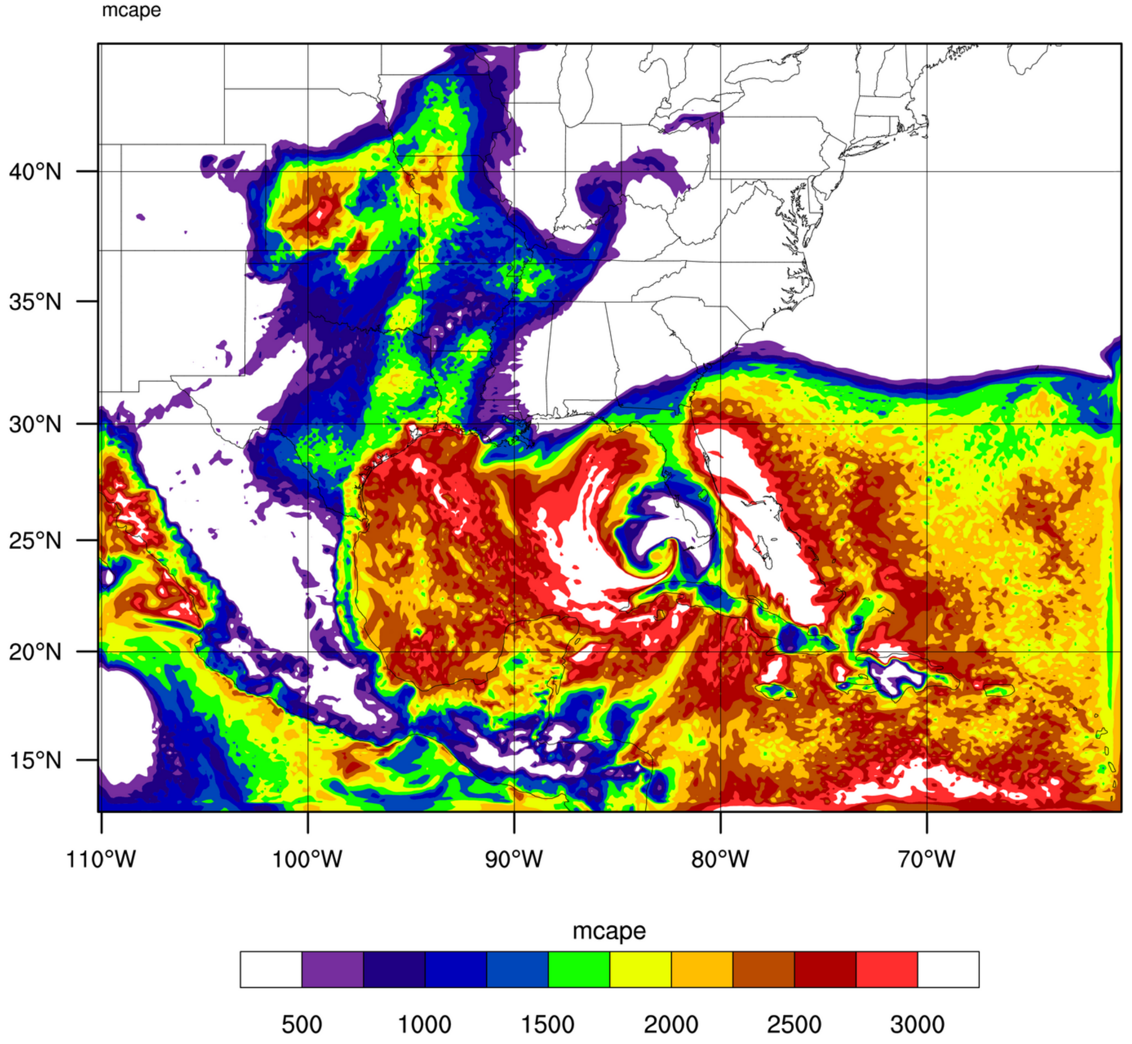
http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/


TEMP at 2 M (K)

2000-01-24_12:00:00
Sea Level Pressure Contours: 990 to 1026 by 2
2000-01-24_15:00:00
Sea Level Pressure Contours: 990 to 1026 by 2
2000-01-24_18:00:00
Sea Level Pressure Contours: 990 to 1026 by 2
2000-01-24_21:00:00
Sea Level Pressure Contours: 990 to 1026 by 2
2000-01-25_00:00:00
Sea Level Pressure Contours: 990 to 1026 by 2
240
255
270
285
300

REAL-TIME WRF

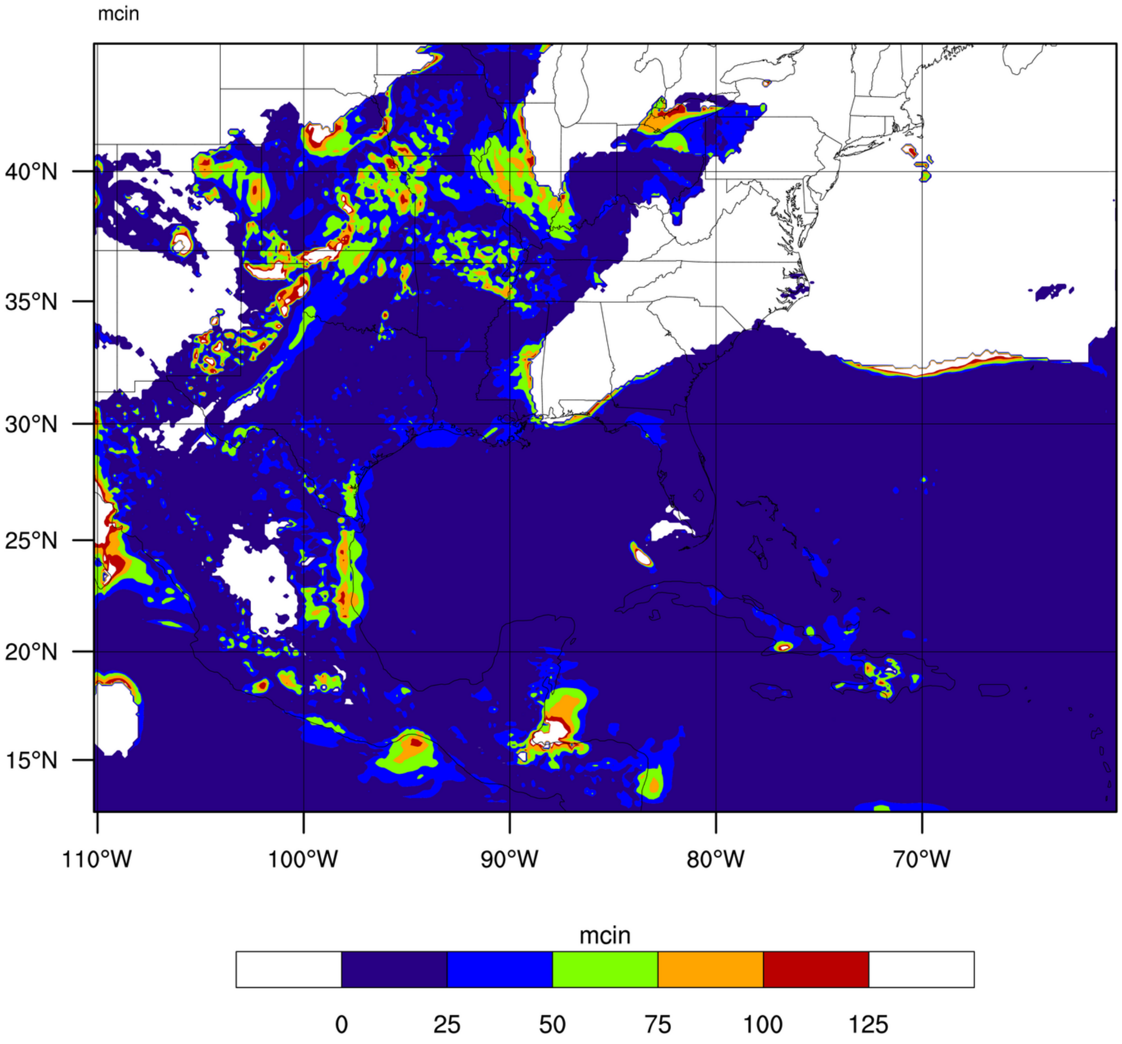
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

REAL-TIME WRF

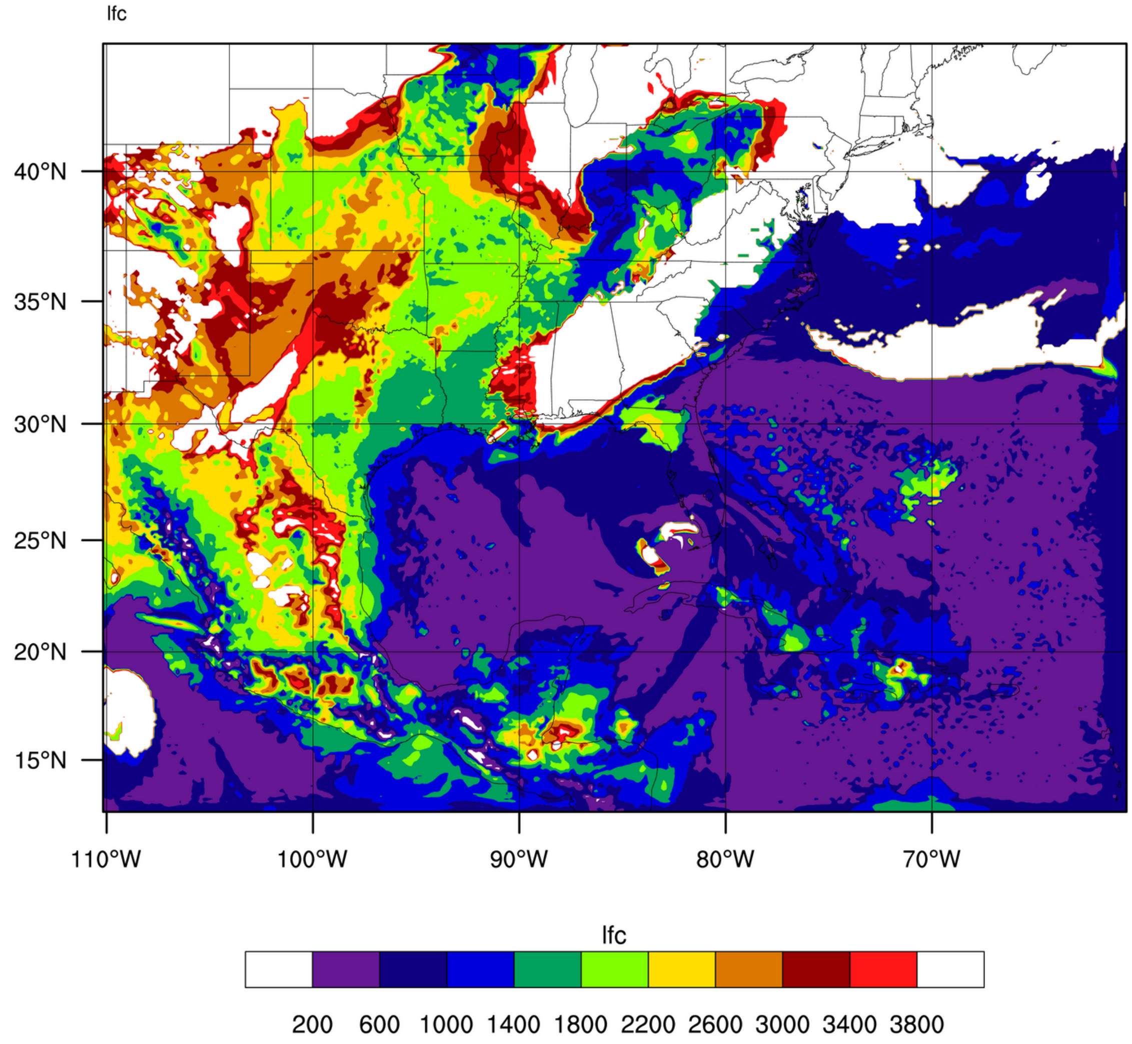
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

REAL-TIME WRF

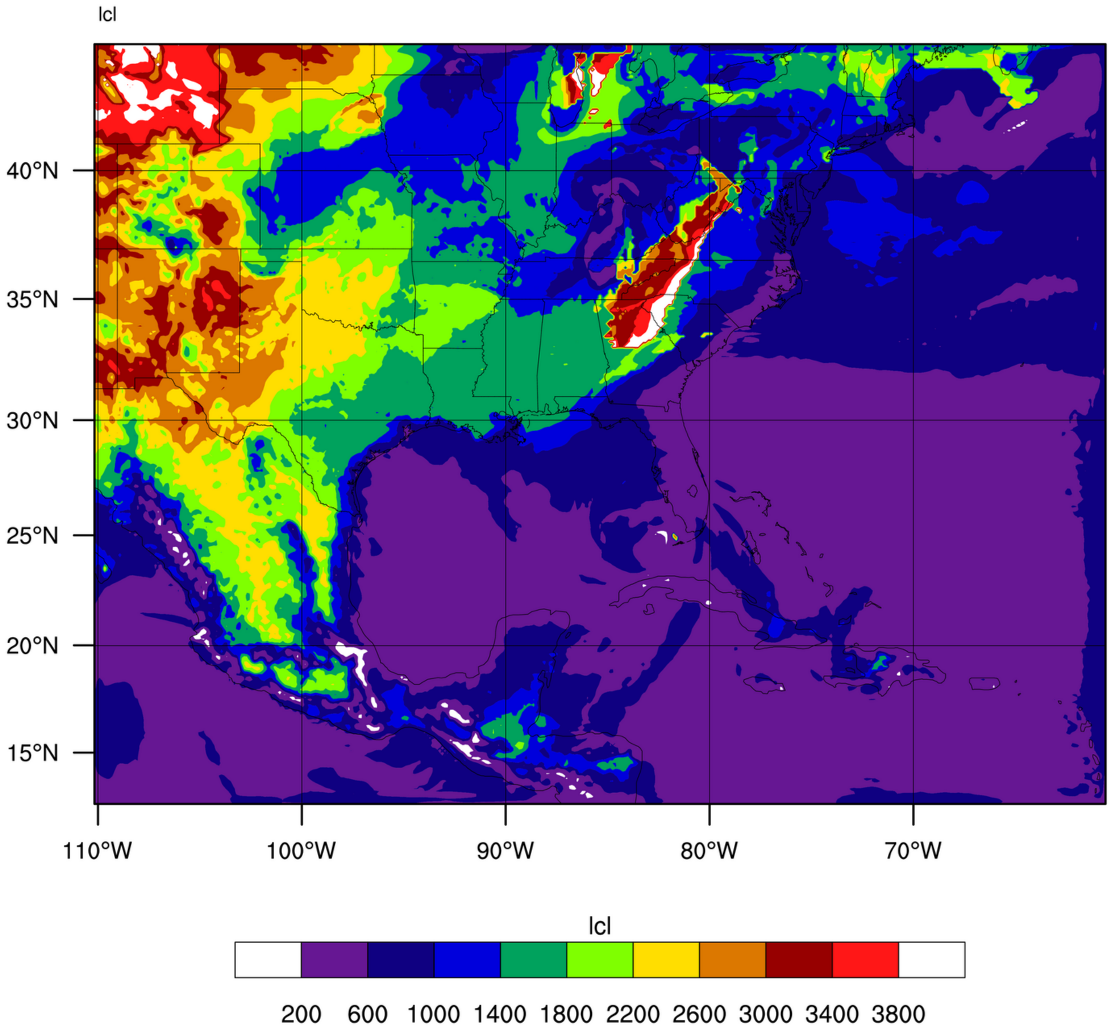
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

REAL-TIME WRF

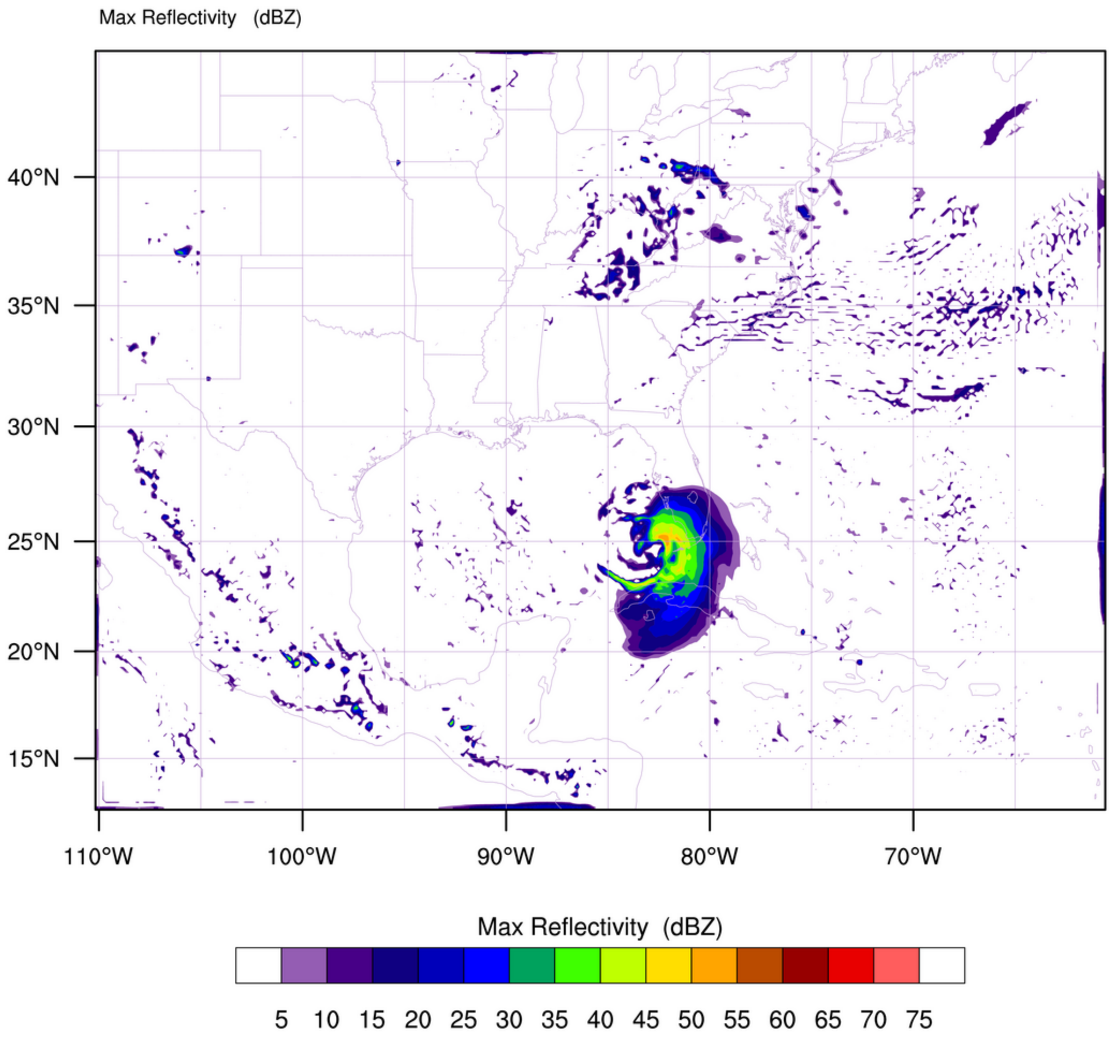
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

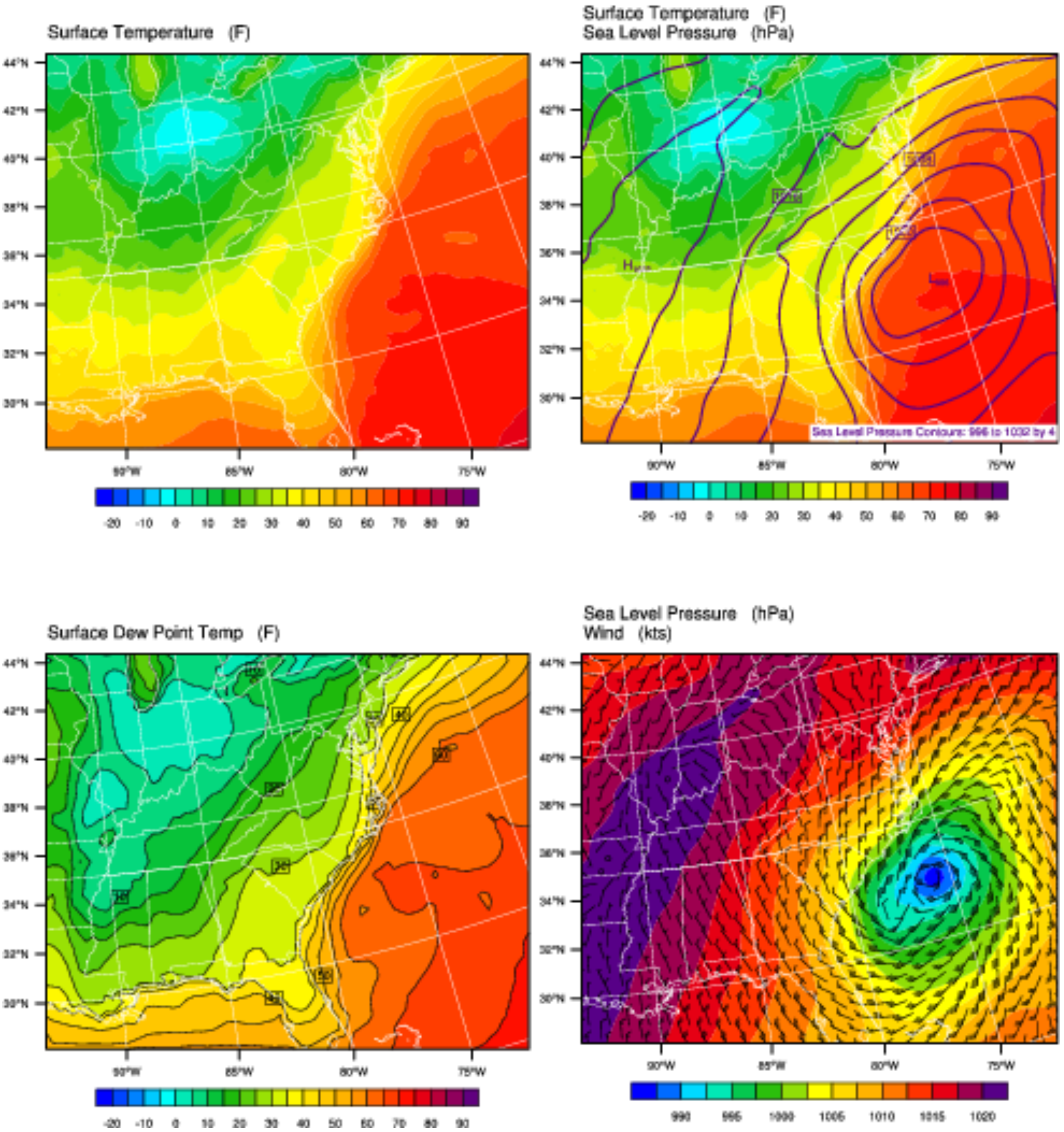
REAL-TIME WRF

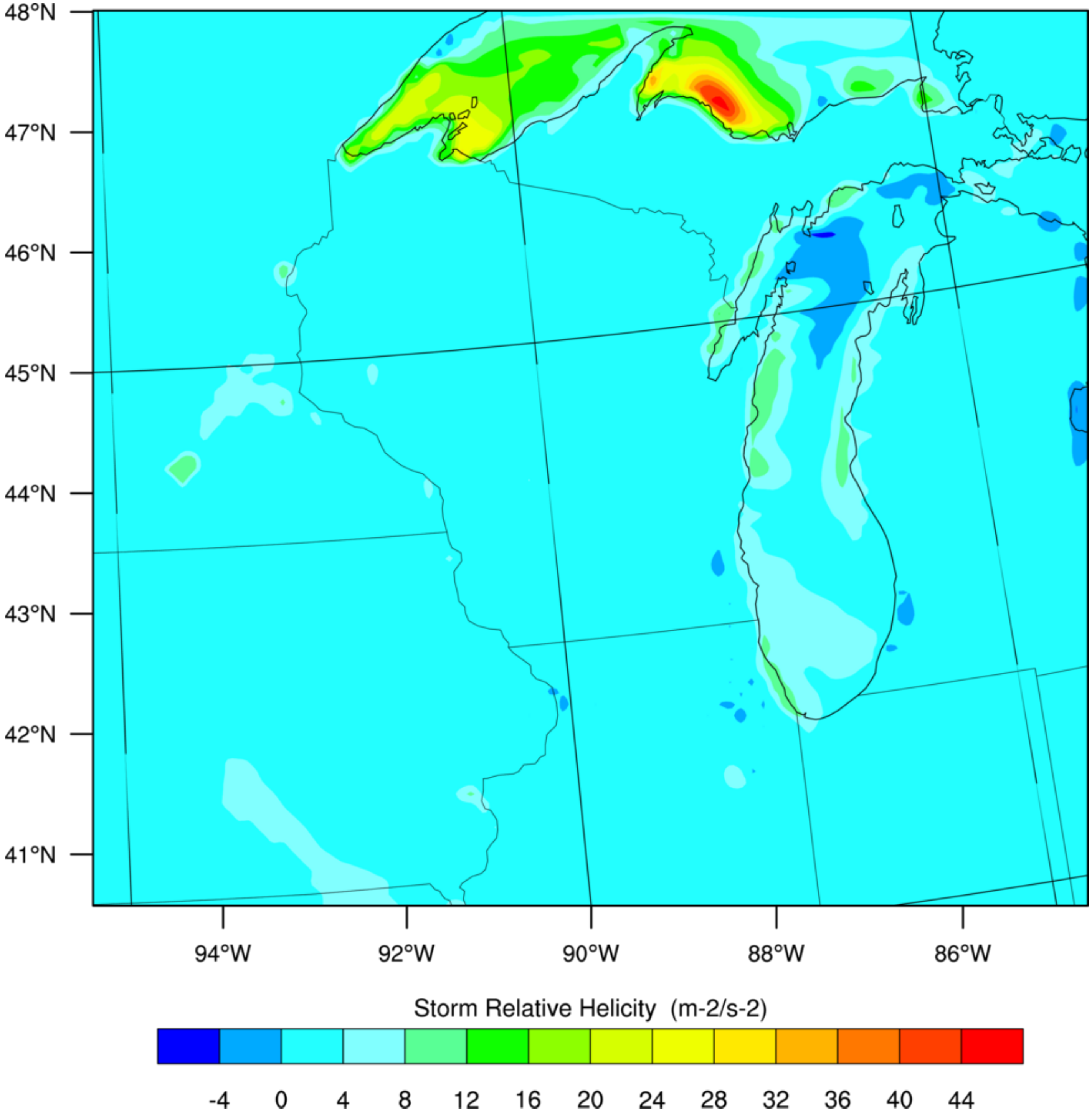
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

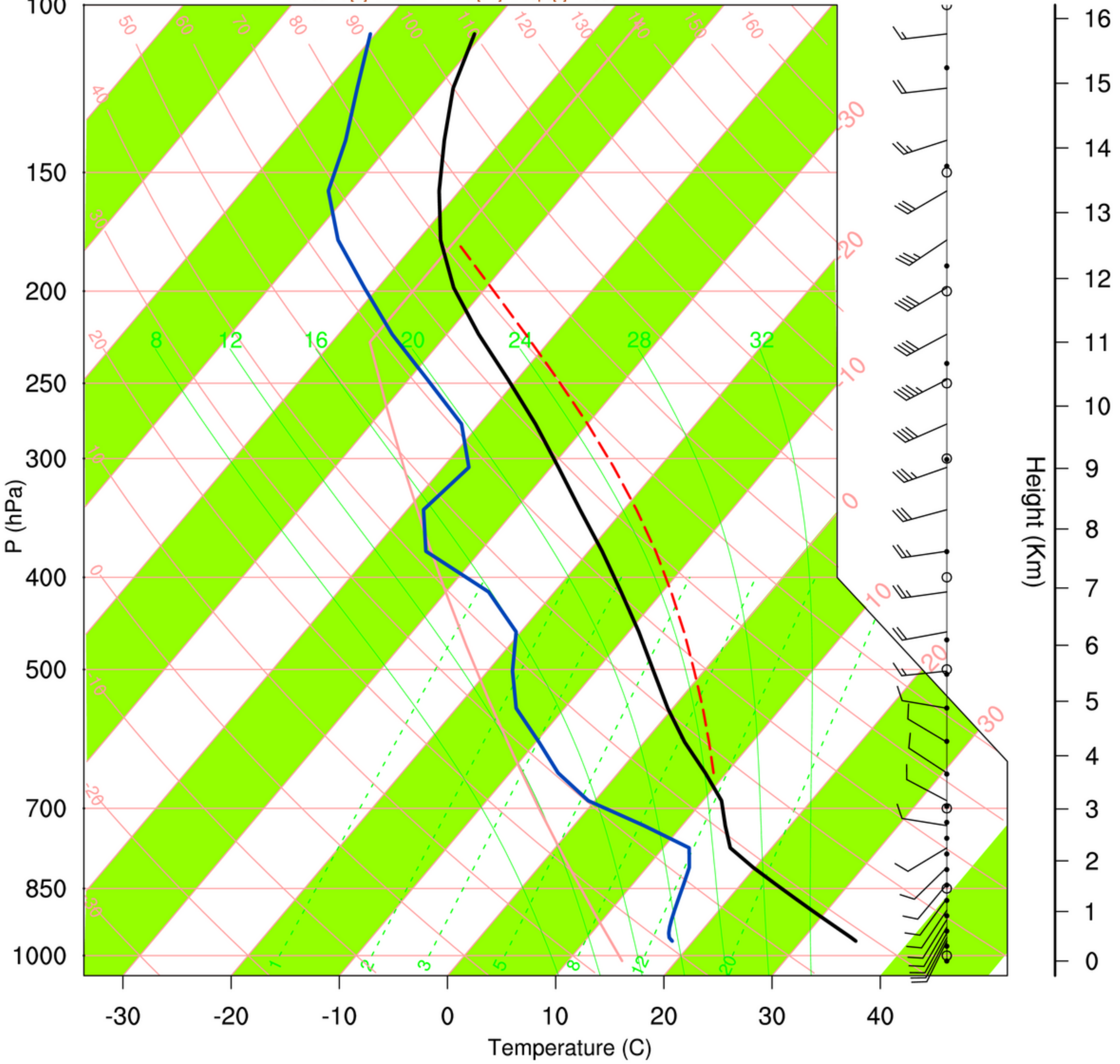
PLOTS for : 2000-01-25_00:00:00




48°N
47°N
46°N
45°N
44°N
43°N
42°N
41°N
94°W
92°W
90°W
88°W
86°W
Storm Relative Helicity (m-2/s-2)
-4
0
4
8
12
16
20
24
28
32
36
40
44

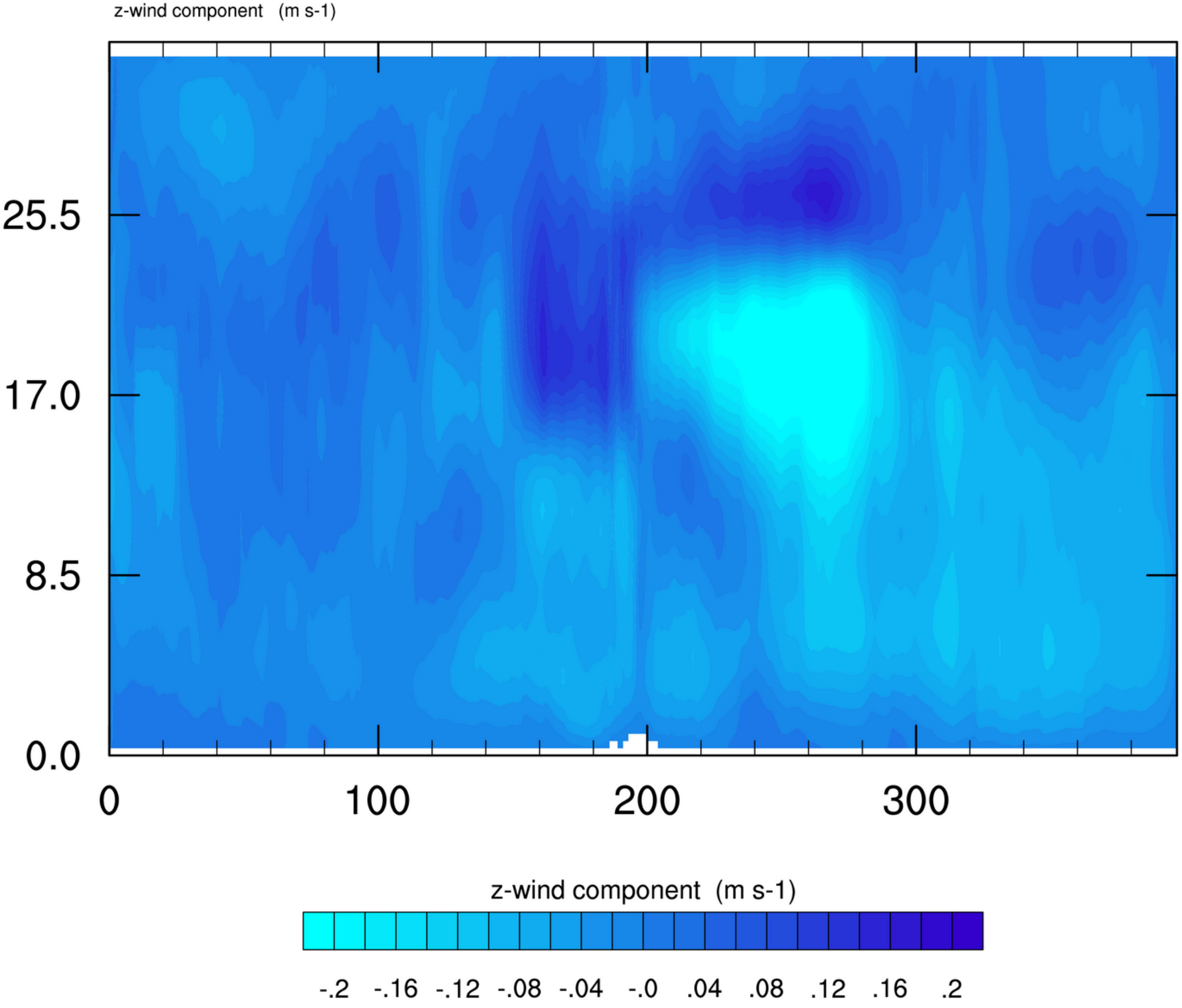
Norman (OK,USA) at 2005-08-27_00:00:00

Plcl=758 Tlcl[C]=14 Shox=-1 Pwat[cm]=4 Cape[J]= 1496

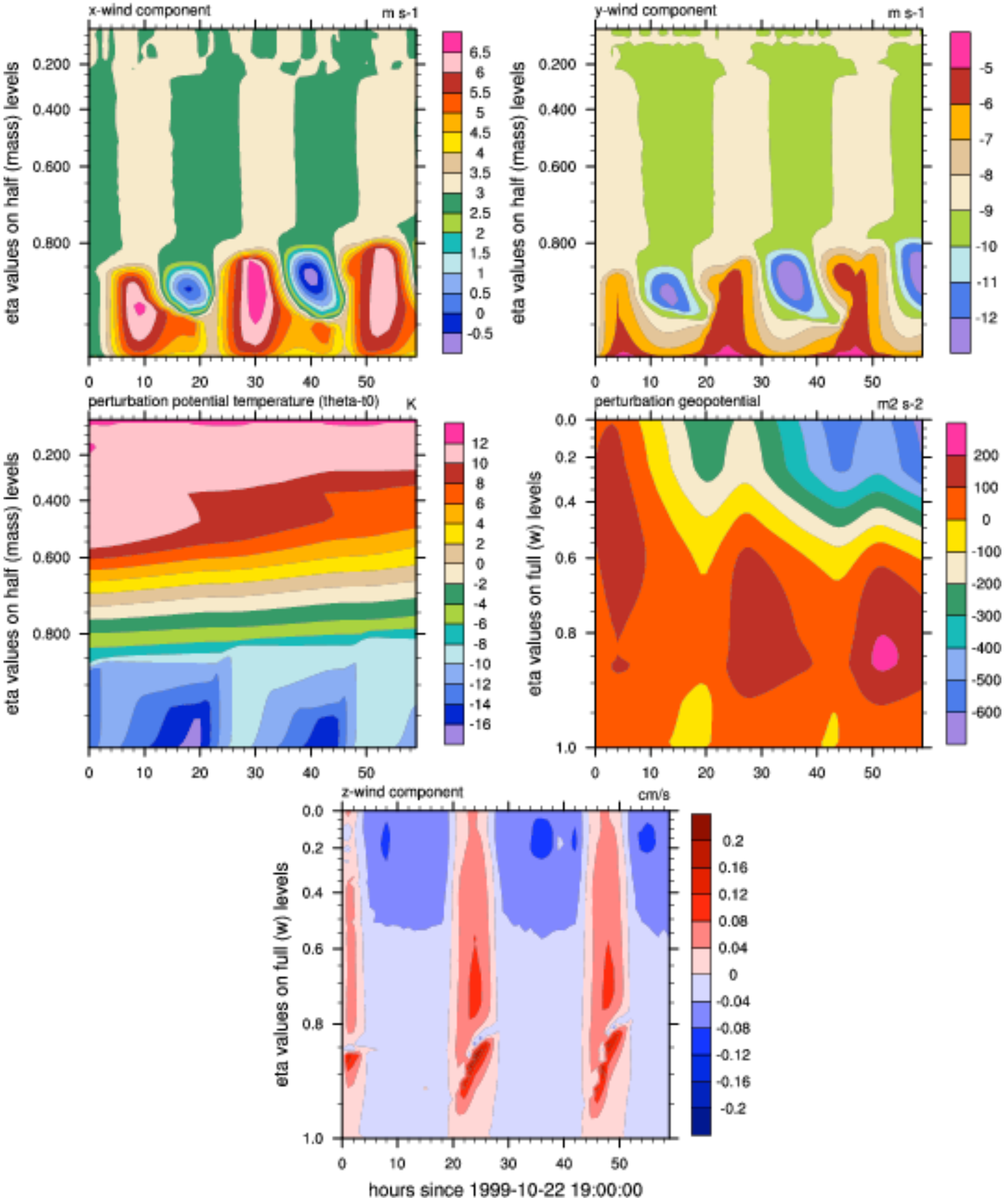


WRF SEABREEZE

Valid: 2005-08-27_00:00:00



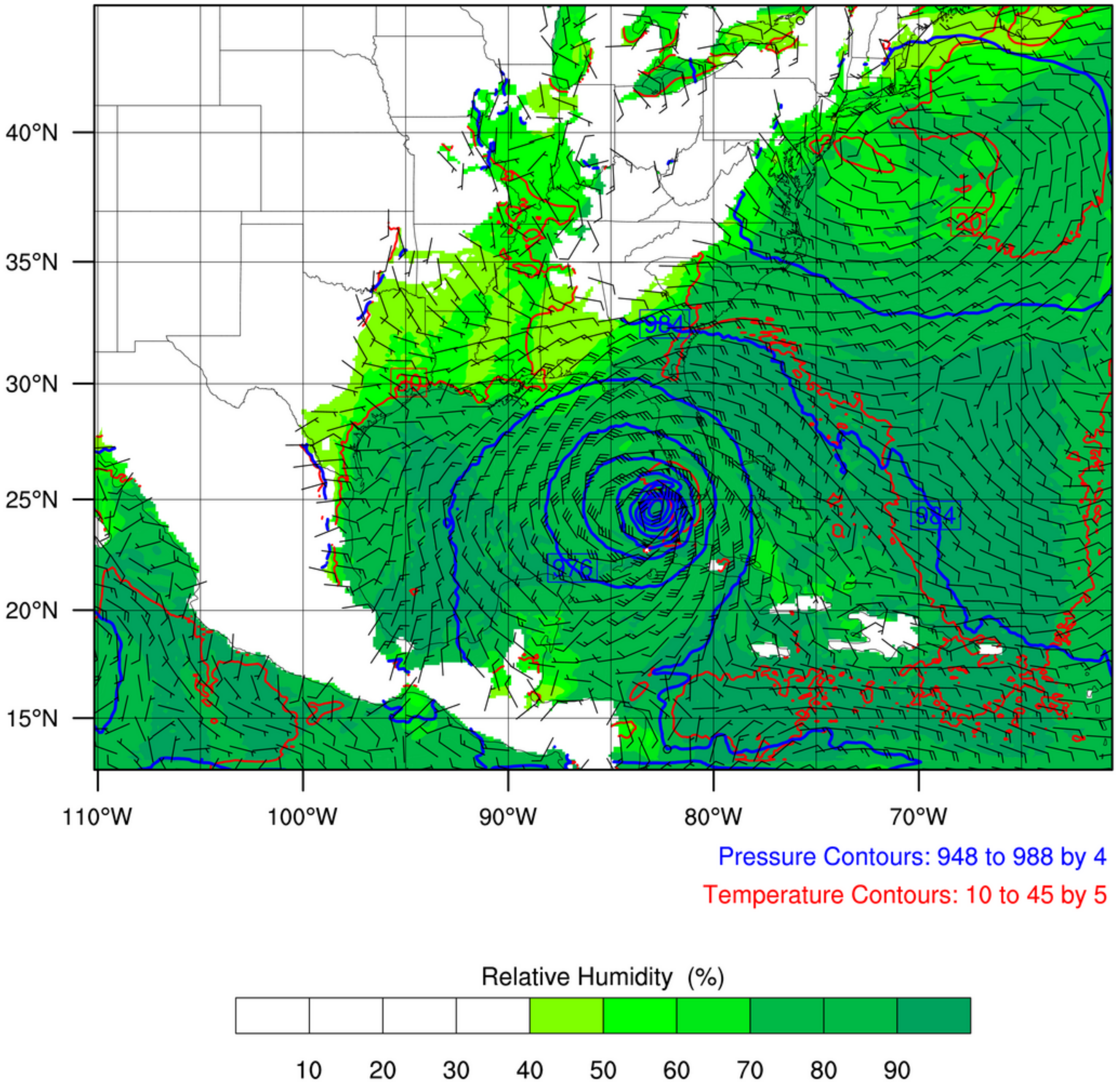
WRF-SCM: 37.60N 96.70W

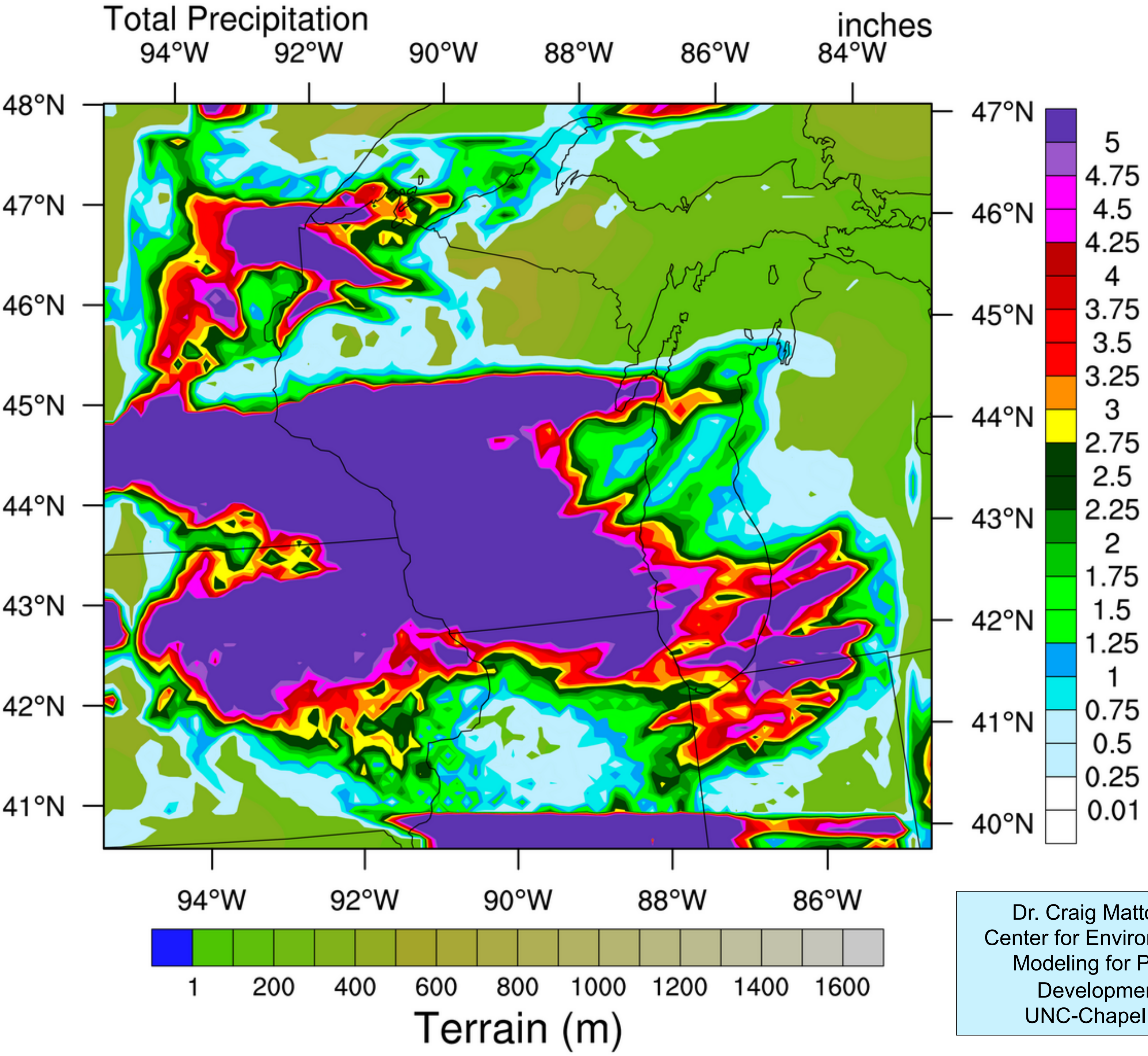


REAL-TIME WRF

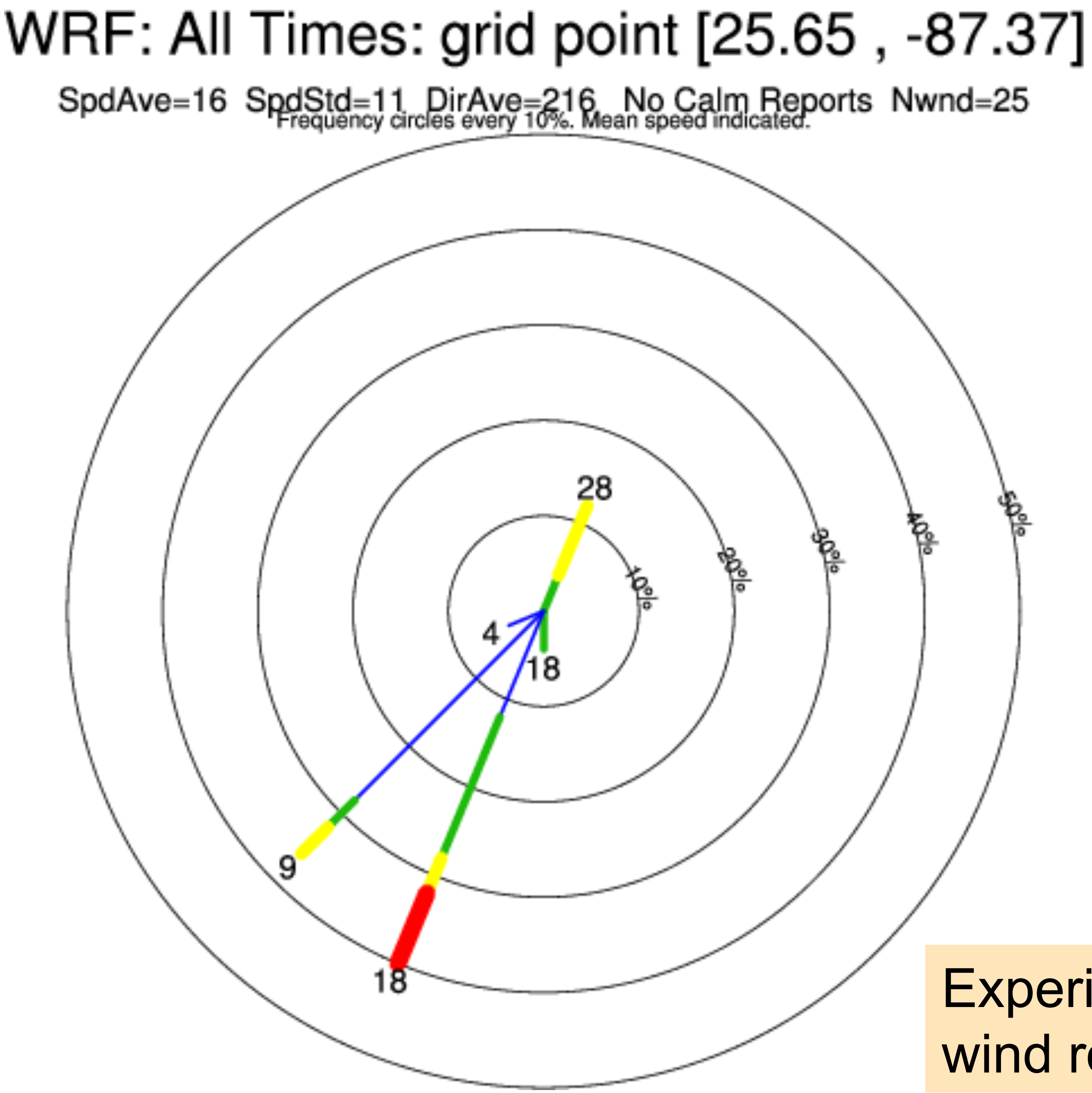
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00

Relative Humidity (%) at 0.25 km
Temperature (C) at 0.25 km
Pressure (hPa) at 0.25 km
Wind (kts) at 0.25 km



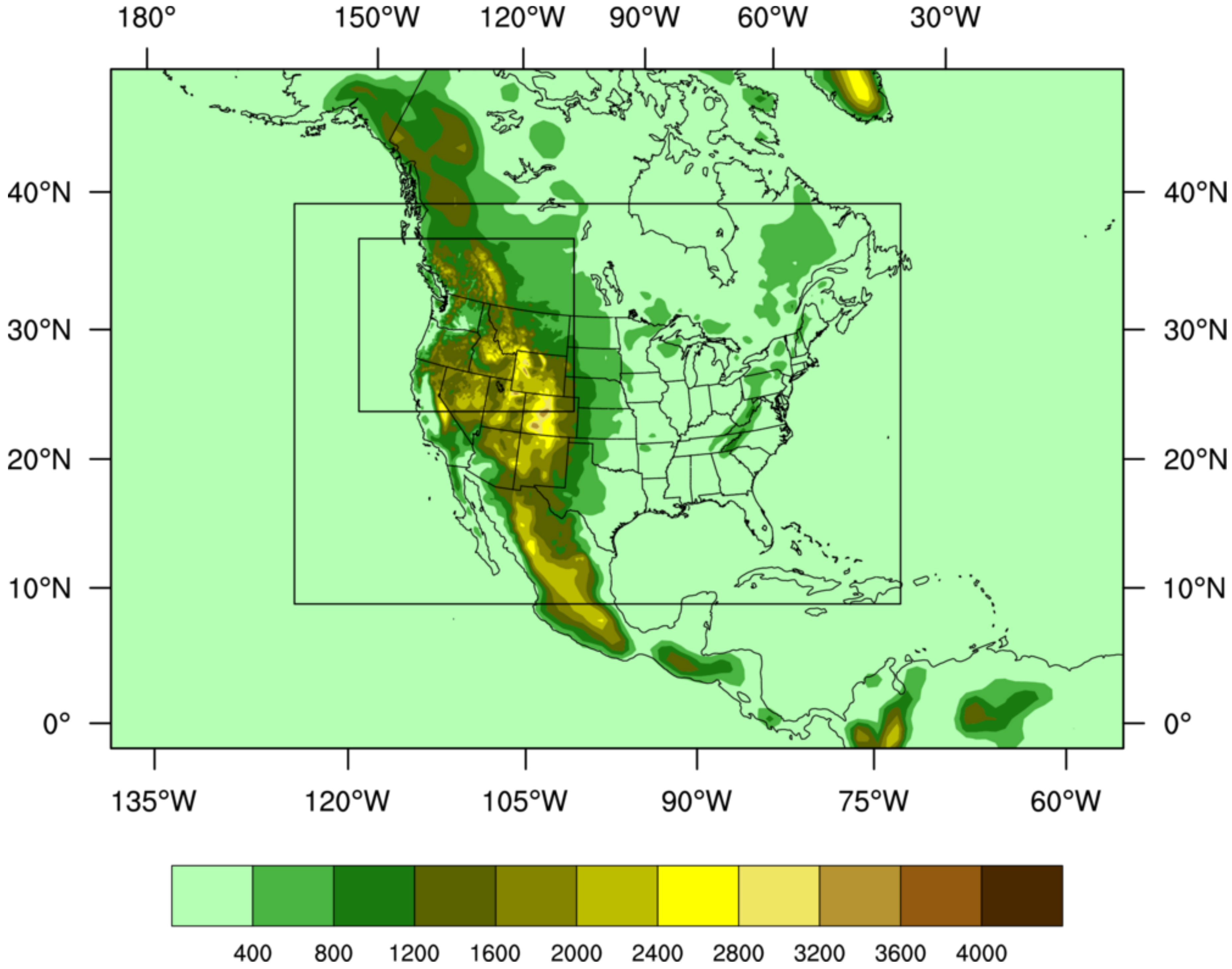


Dr. Craig Mattocks,
Center for Environmental
Modeling for Policy
Development,
UNC-Chapel Hill

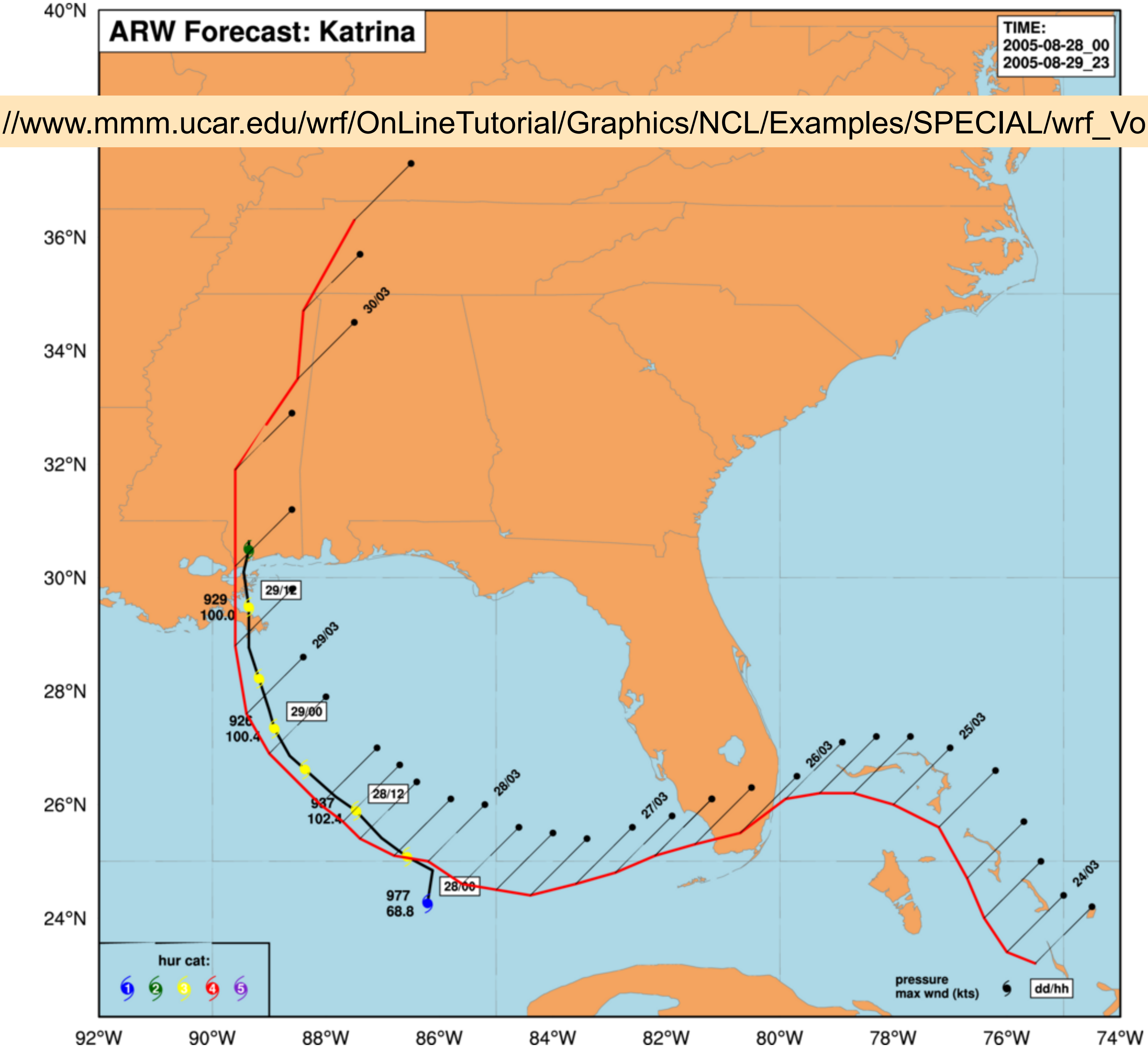


Experimental: wind roses

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/Examples/EXPERIMENTAL/wrf_wind_rose.htm


180°
150°W
120°W
90°W
60°W
30°W
40°N
30°N
20°N
10°N
0°
135°W
120°W
105°W
90°W
75°W
60°W
400
800
1200
1600
2000
2400
2800
3200
3600
4000

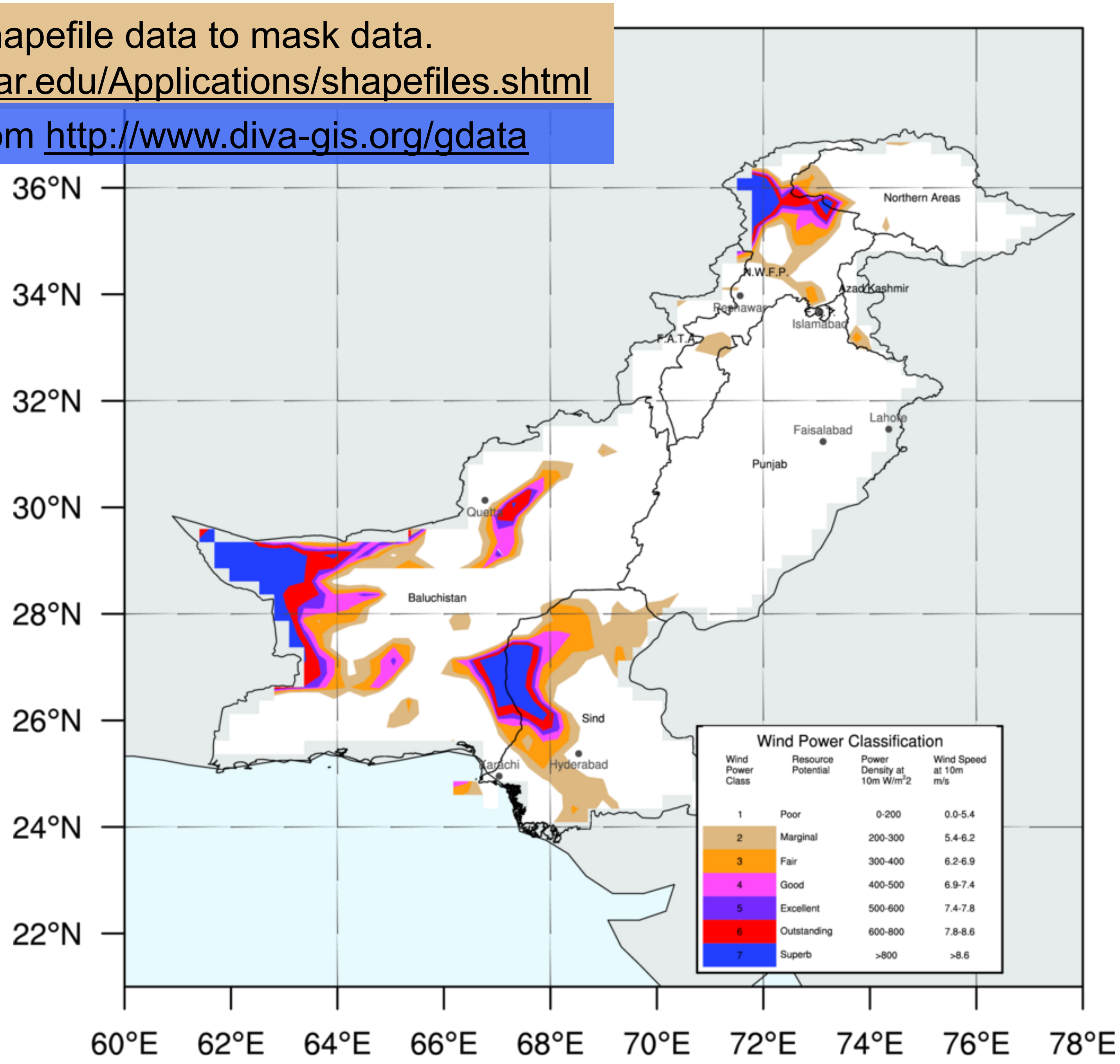
http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/Examples/SPECIAL/wrf_Vortex.htm



Wind Speed: 10m

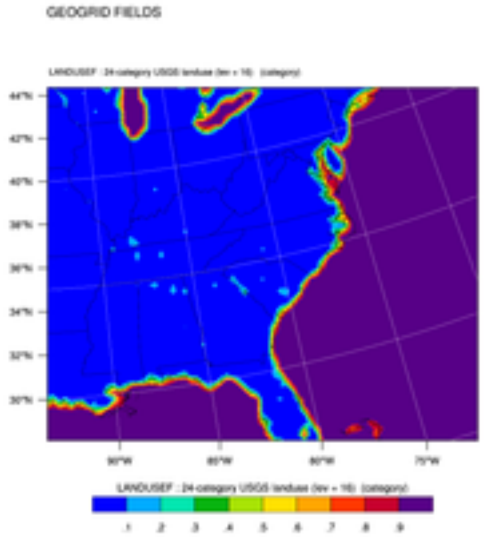
Uses shapefile data to mask data.
http://www.ncl.ucar.edu/Applications/shapefiles.shtml

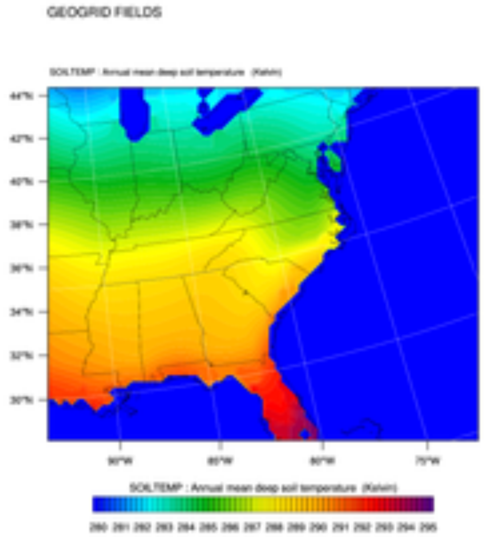
Shapefiles from http://www.diva-gis.org/gdata

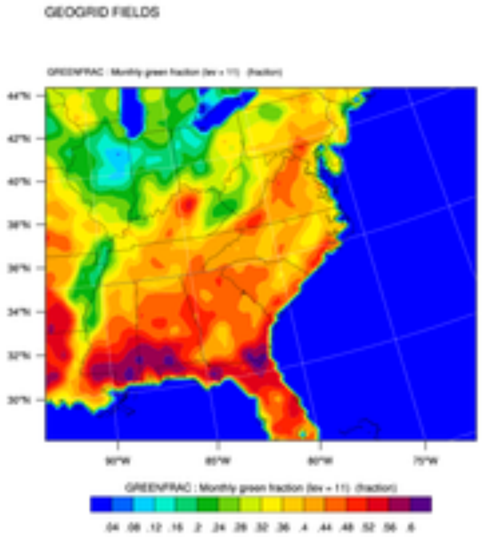


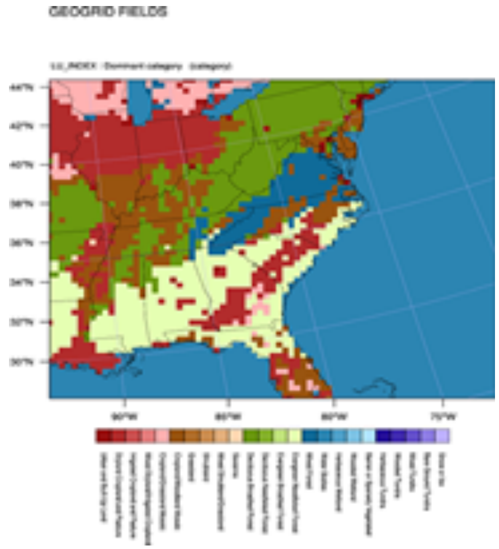
Plotting all fields in a GEO_EM file

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/Examples/GEO_EM/geo_em_2.htm

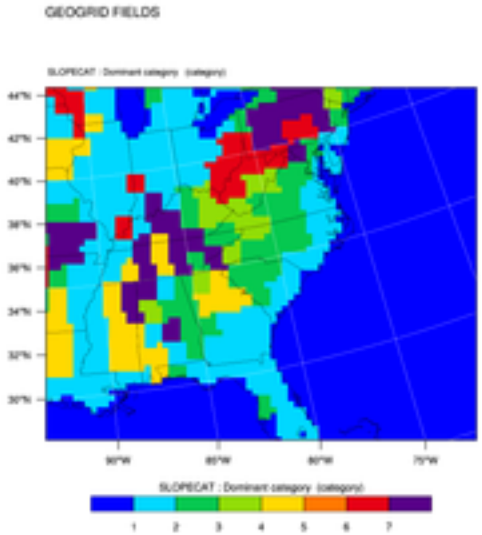

GEOGRID FIELDS
LANDUSEF : 24-category USGS landuse (lev = 16) (category)
44°N
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
LANDUSEF : 24-category USGS landuse (lev = 16) (category)
.1
.2
.3
.4
.5
.6
.7
.8
.9




GEOGRID FIELDS
GREENFRAC : Monthly green fraction (lev = 11) (fraction)
44°N
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
GREENFRAC : Monthly green fraction (lev = 11) (fraction)
.04
.08
.12
.16
.2
.24
.28
.32
.36
.4
.44
.48
.52
.56
.6



GEOGRID FIELDS
LANDMASK : Landmask : 1=land, 0=water (none)
44°N
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W


GEOGRID FIELDS
SLOPECAT : Dominant category (category)
44°N
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
SLOPECAT : Dominant category (category)
1
2
3
4
5
6
7

Other NCL visualizations

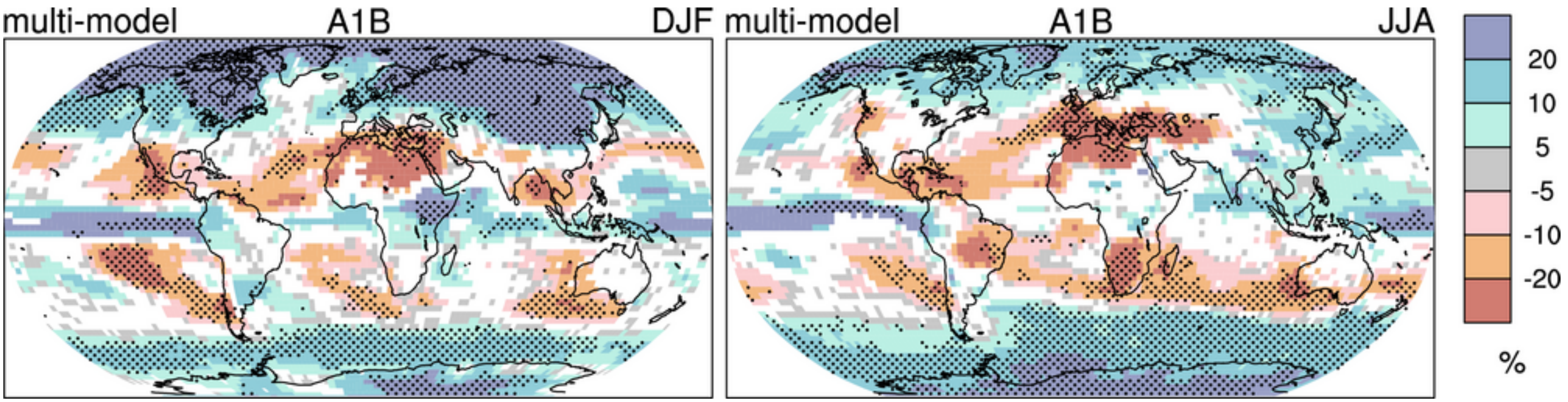
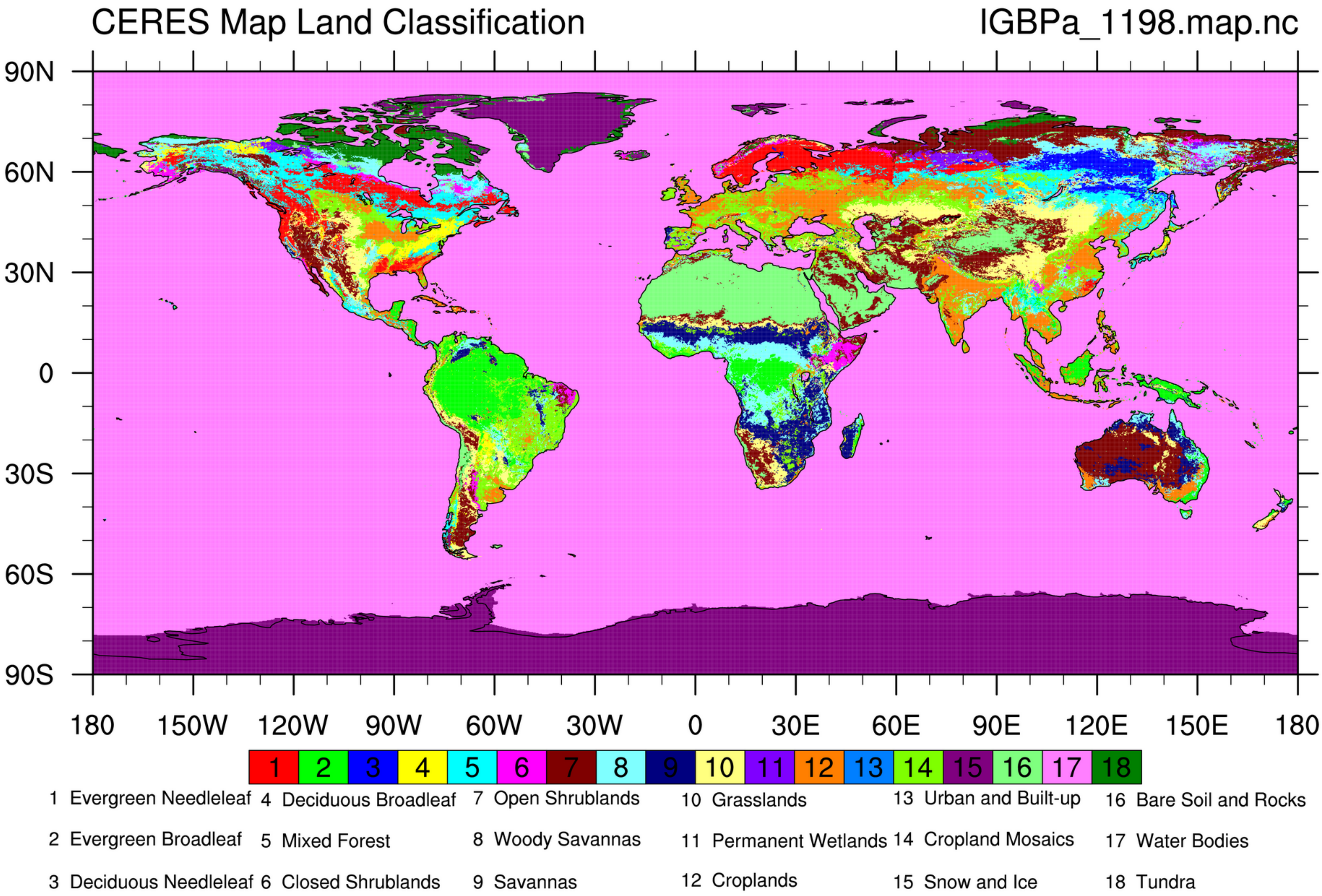
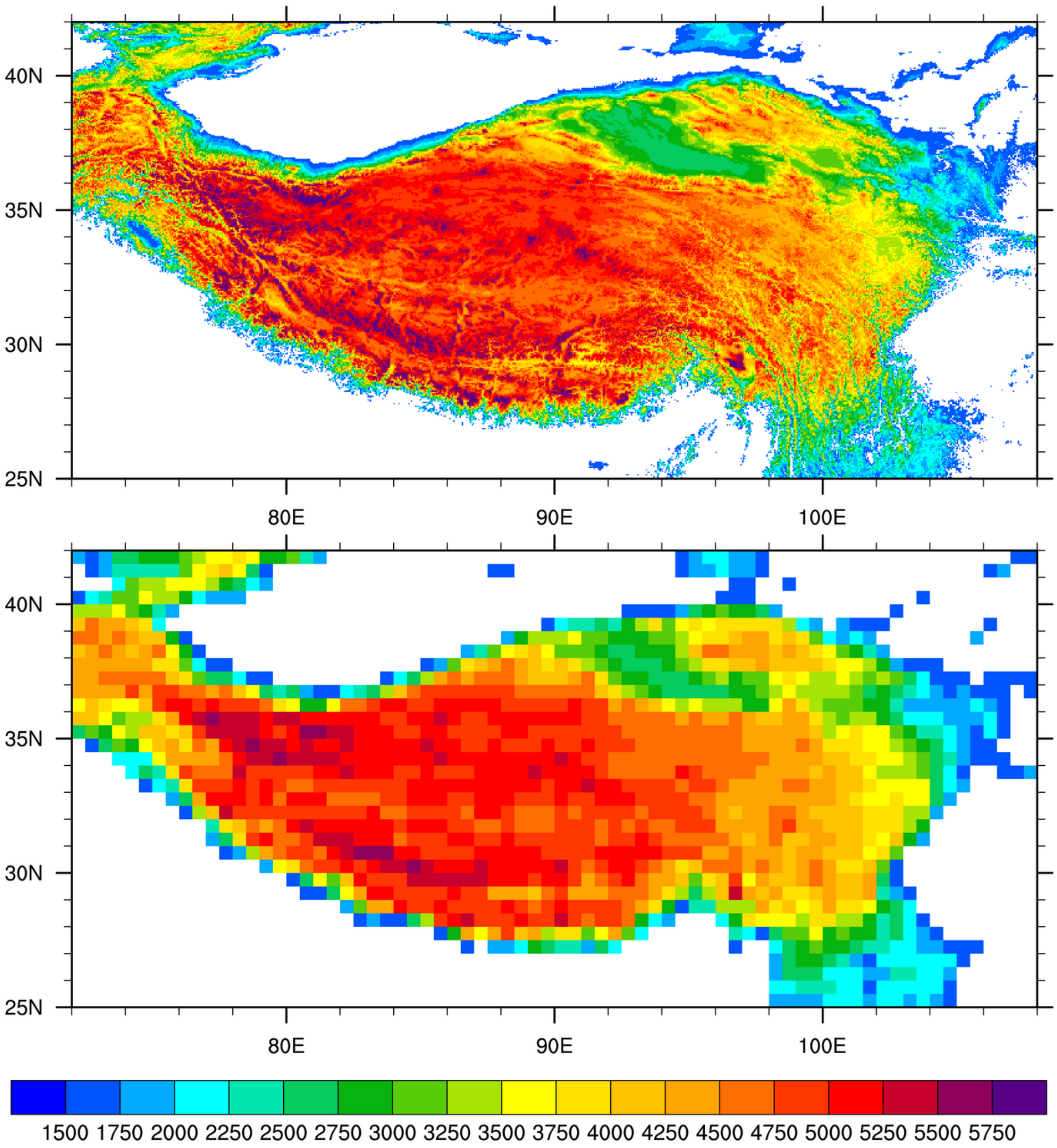


Image courtesy of Julie Arblaster
Bureau of Meteorology, University of Melbourne

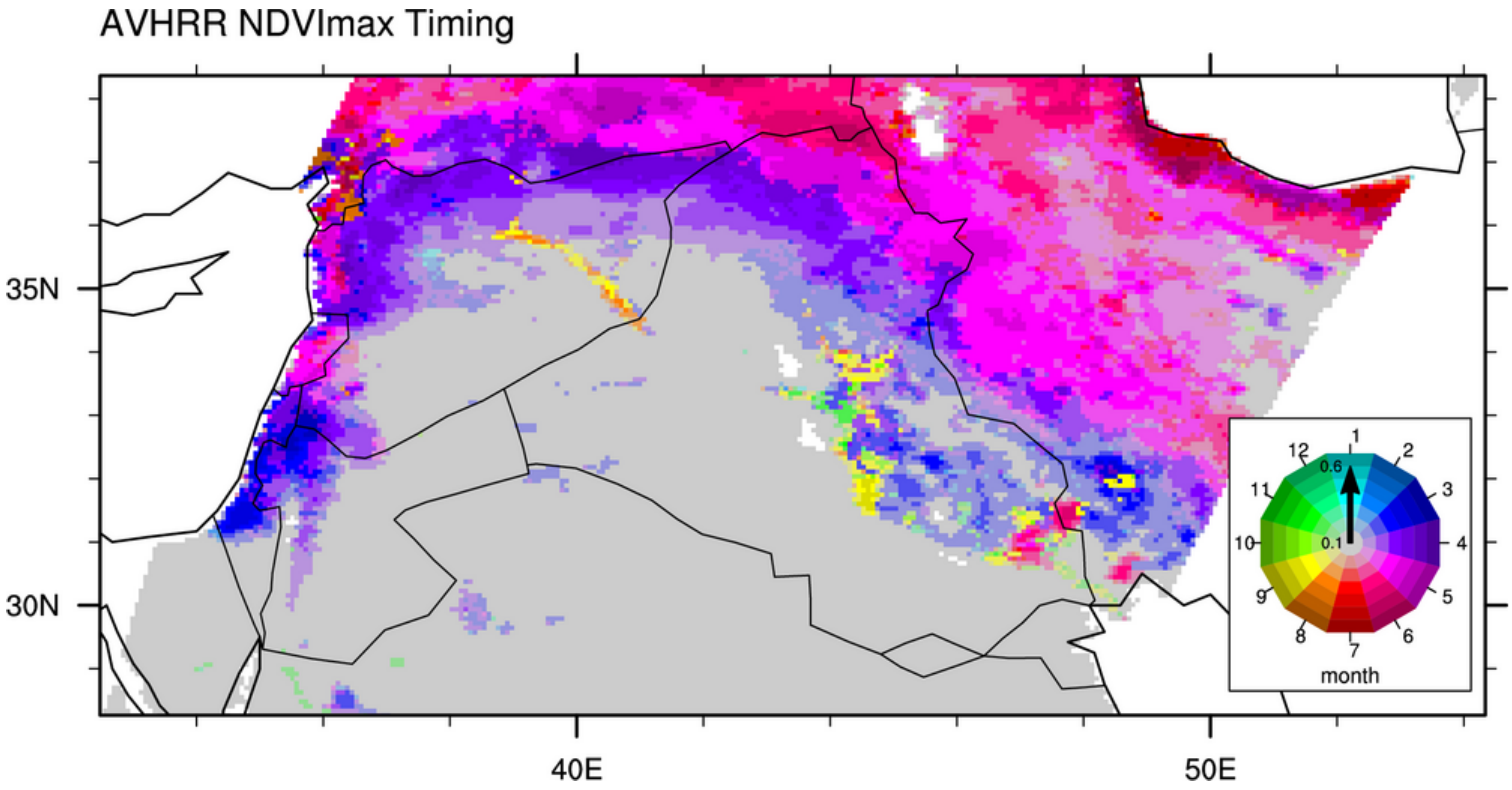


Classification data example, courtesy of Dennis Shea, NCAR/CGD

NGDC, ETOPO2 Global 2’ Elevations: Tibet: 1500

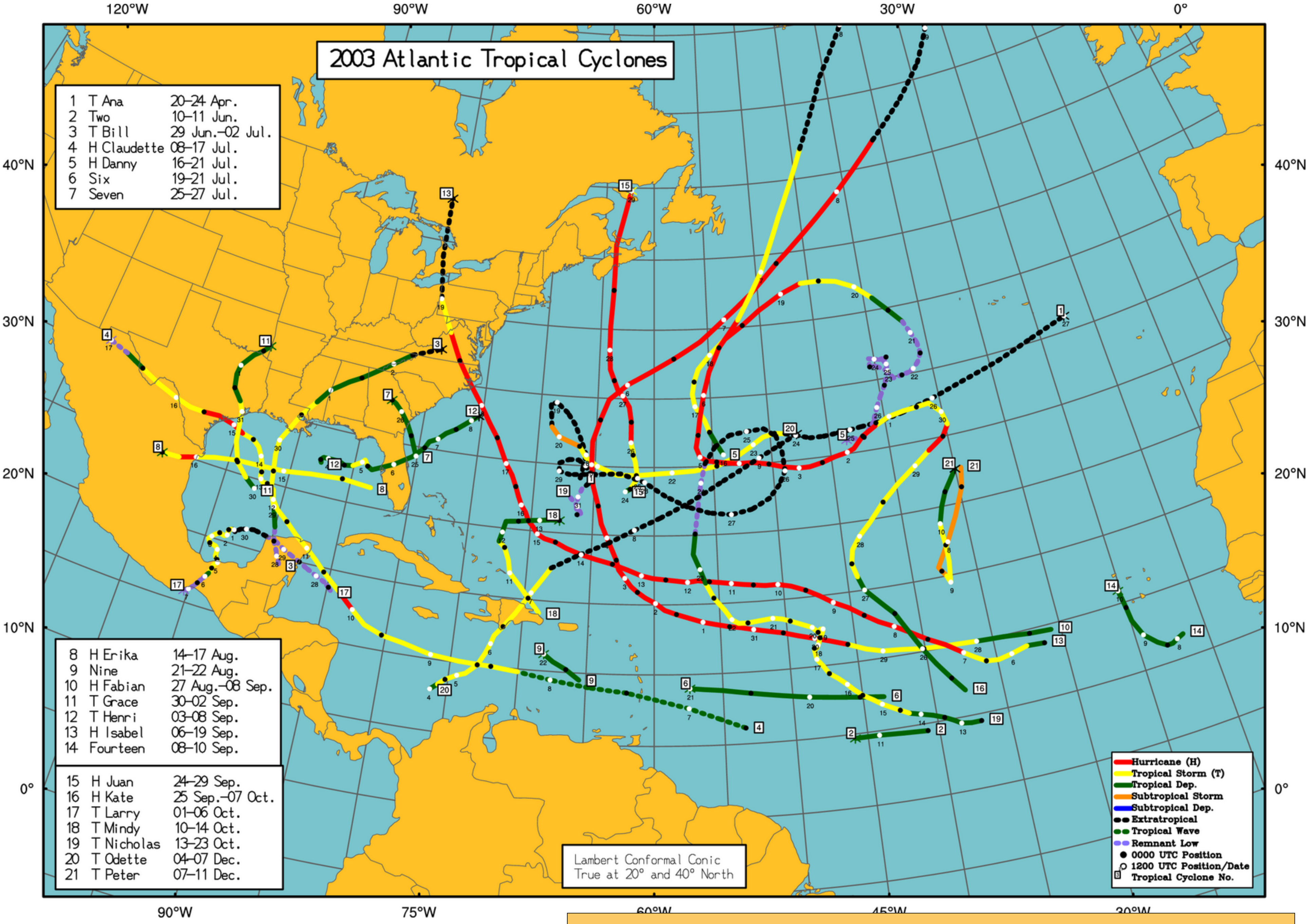


Interpolating from a higher resolution grid to a lower resolution using conservative remapping courtesy Dennis Shea NCAR/CGD



Evans plot - Created by Jason Evans of Univ of New South Wales.

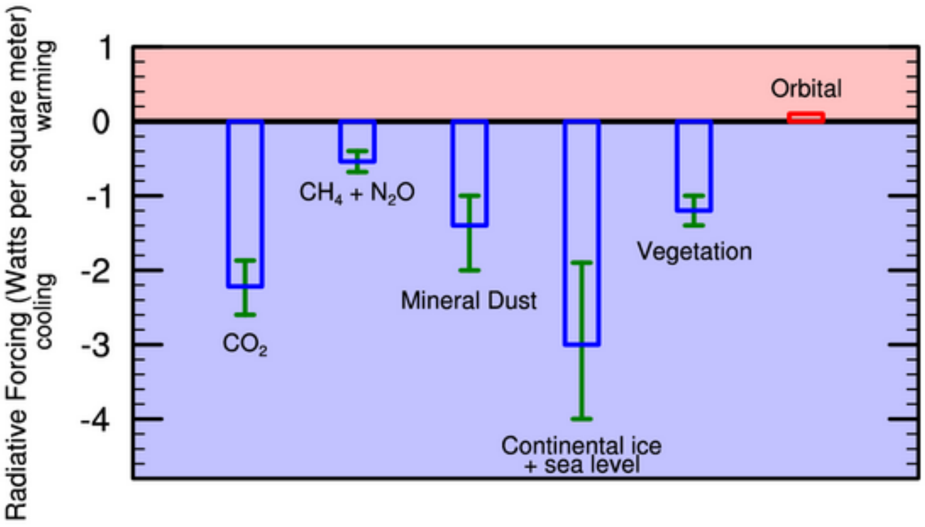
An Evans plot is a way to visualize spatially, two variables of interest, one of which provides some measure of "importance".

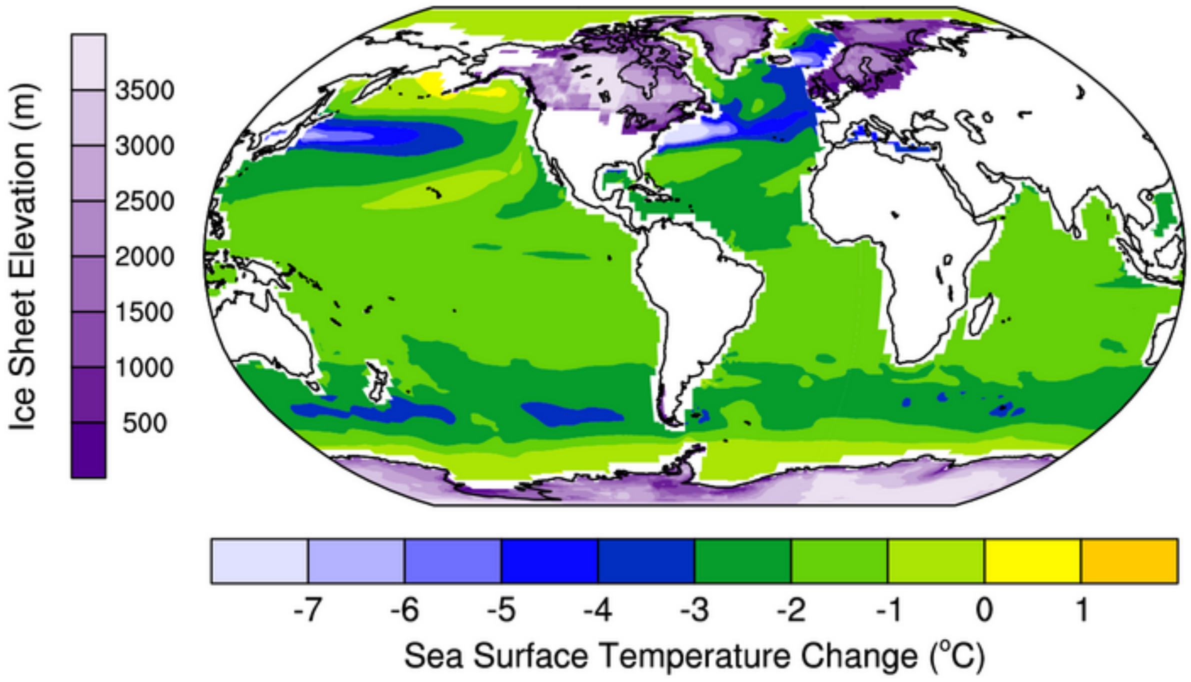


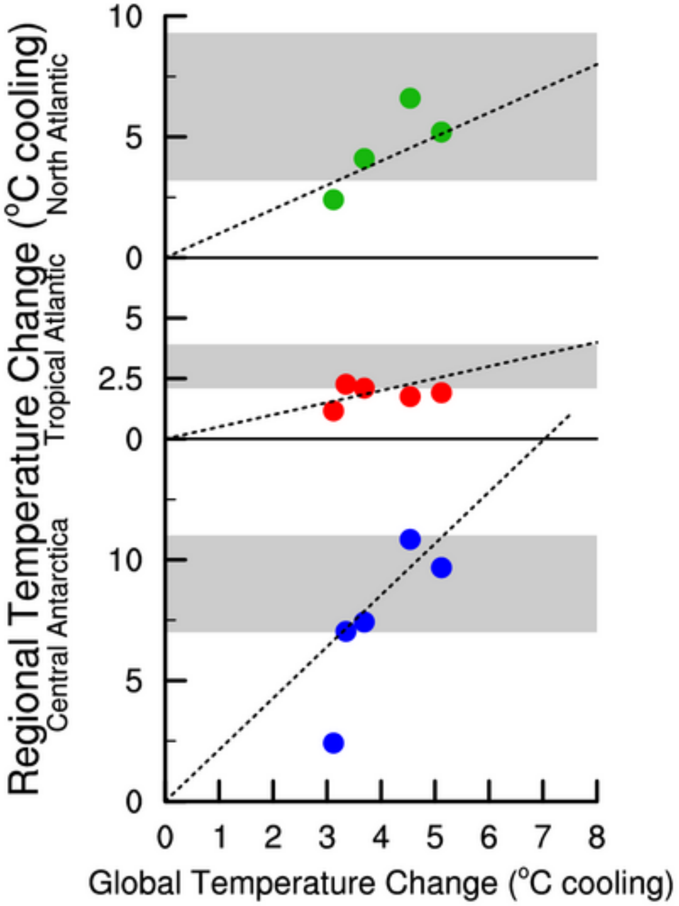
2003 Atlantic Tropical Cyclones

1	T Ana	20–24 Apr.
2	Two	10–11 Jun.
3	T Bill	29 Jun.–02 Jul.
4	H Claudette	08–17 Jul.
5	H Danny	16–21 Jul.
6	Six	19–21 Jul.
7	Seven	25–27 Jul.
8	H Erika	14–17 Aug.
9	Nine	21–22 Aug.
10	H Fabian	27 Aug.–08 Sep.
11	T Grace	30–02 Sep.
12	T Henri	03–08 Sep.
13	H Isabel	06–19 Sep.
14	Fourteen	08–10 Sep.
15	H Juan	24–29 Sep.
16	H Kate	25 Sep.–07 Oct.
17	T Larry	01–06 Oct.
18	T Mindy	10–14 Oct.
19	T Nicholas	13–23 Oct.
20	T Odette	04–07 Dec.
21	T Peter	07–11 Dec.

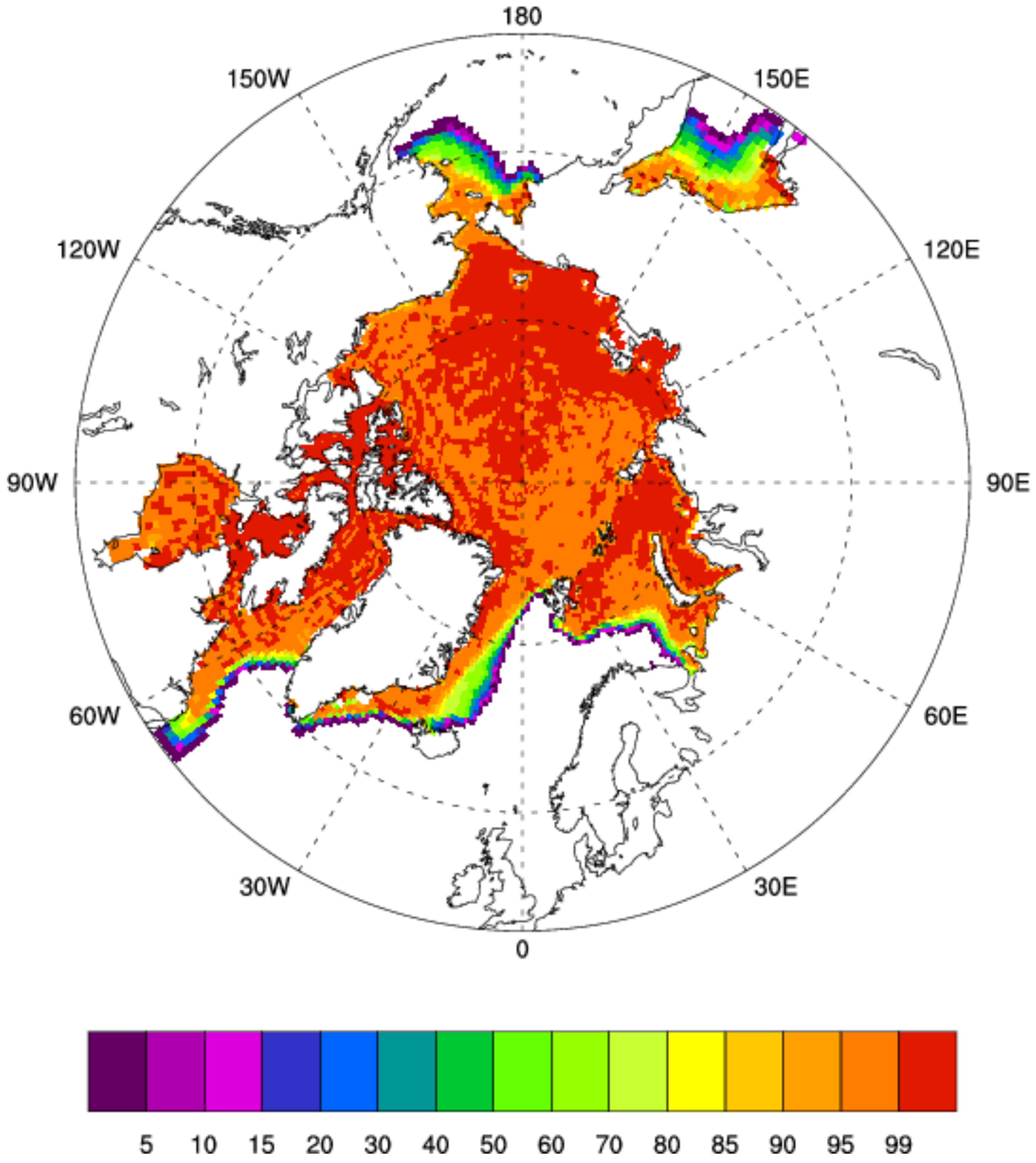
Graphic by Jonathan Vigh, Postdoc @ NCAR







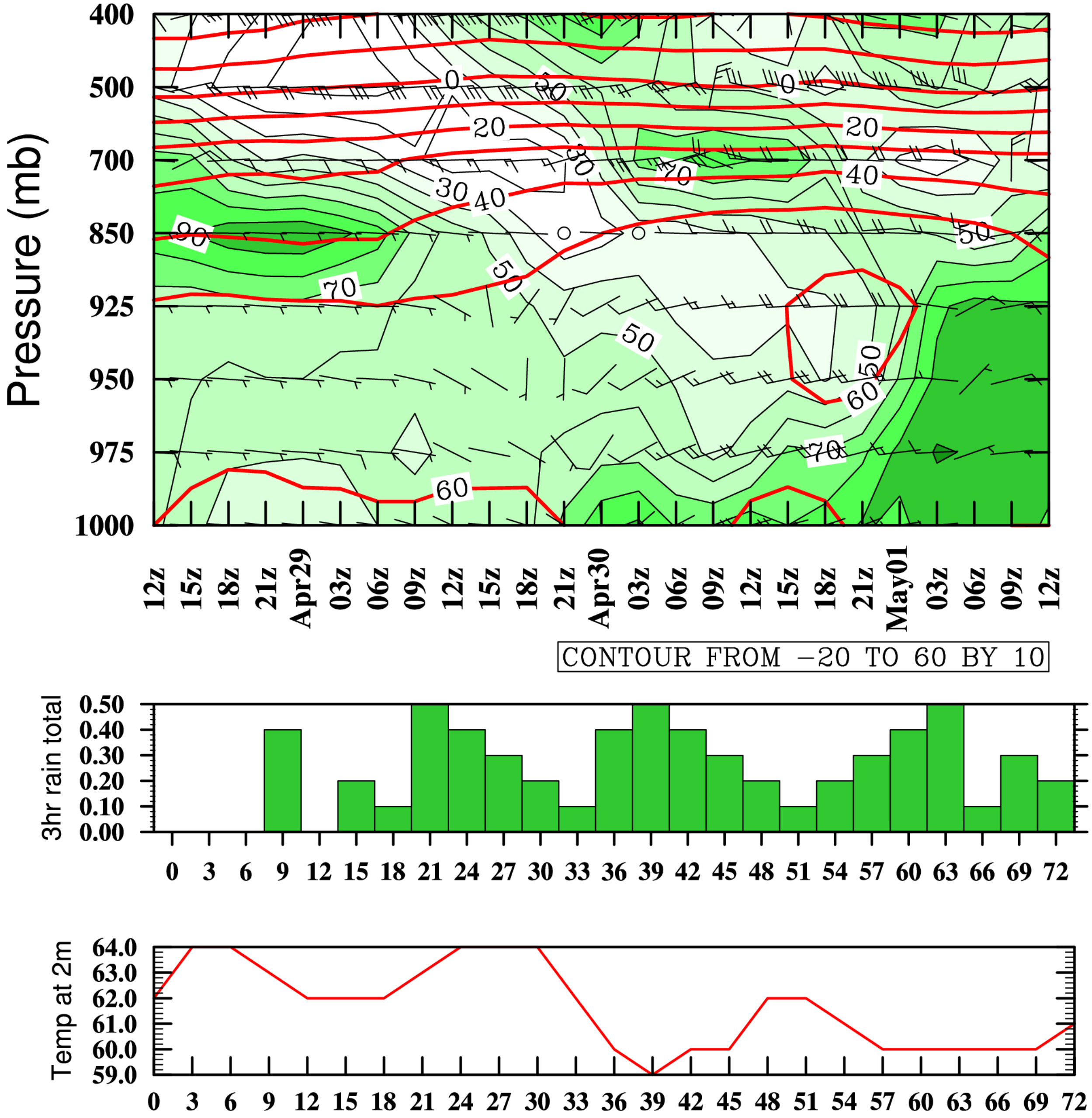
Based on a visualization of Adam Phillips


t-grid extended with u-grid

A CICE T-fold Tripole grid. Data and tips for plotting provided by Petteri Uotila of CSIRO Marine & Atmospheric Research Victoria, Australia

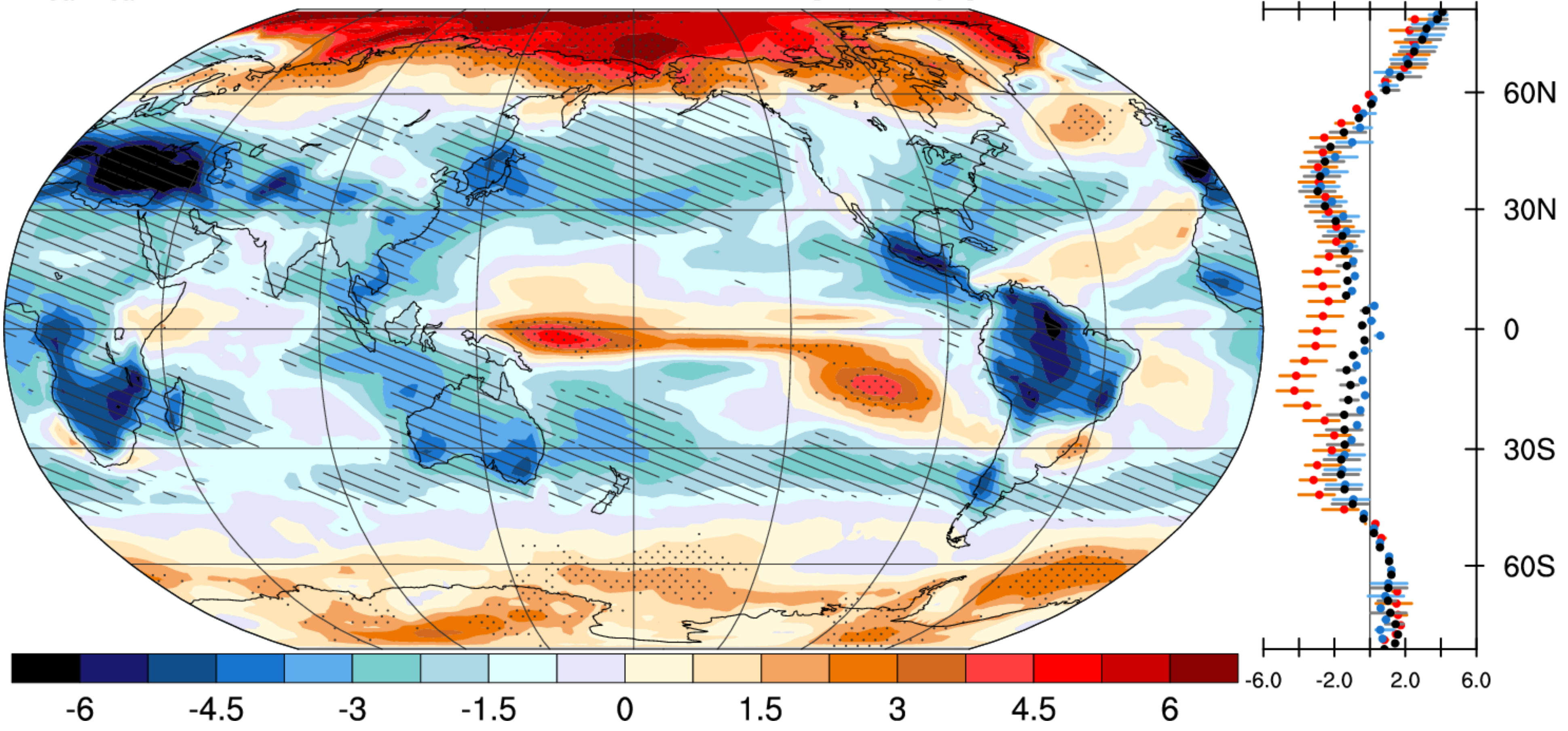
From John Ertl, FNMOC

Meteogram for LGSA, 28/12Z



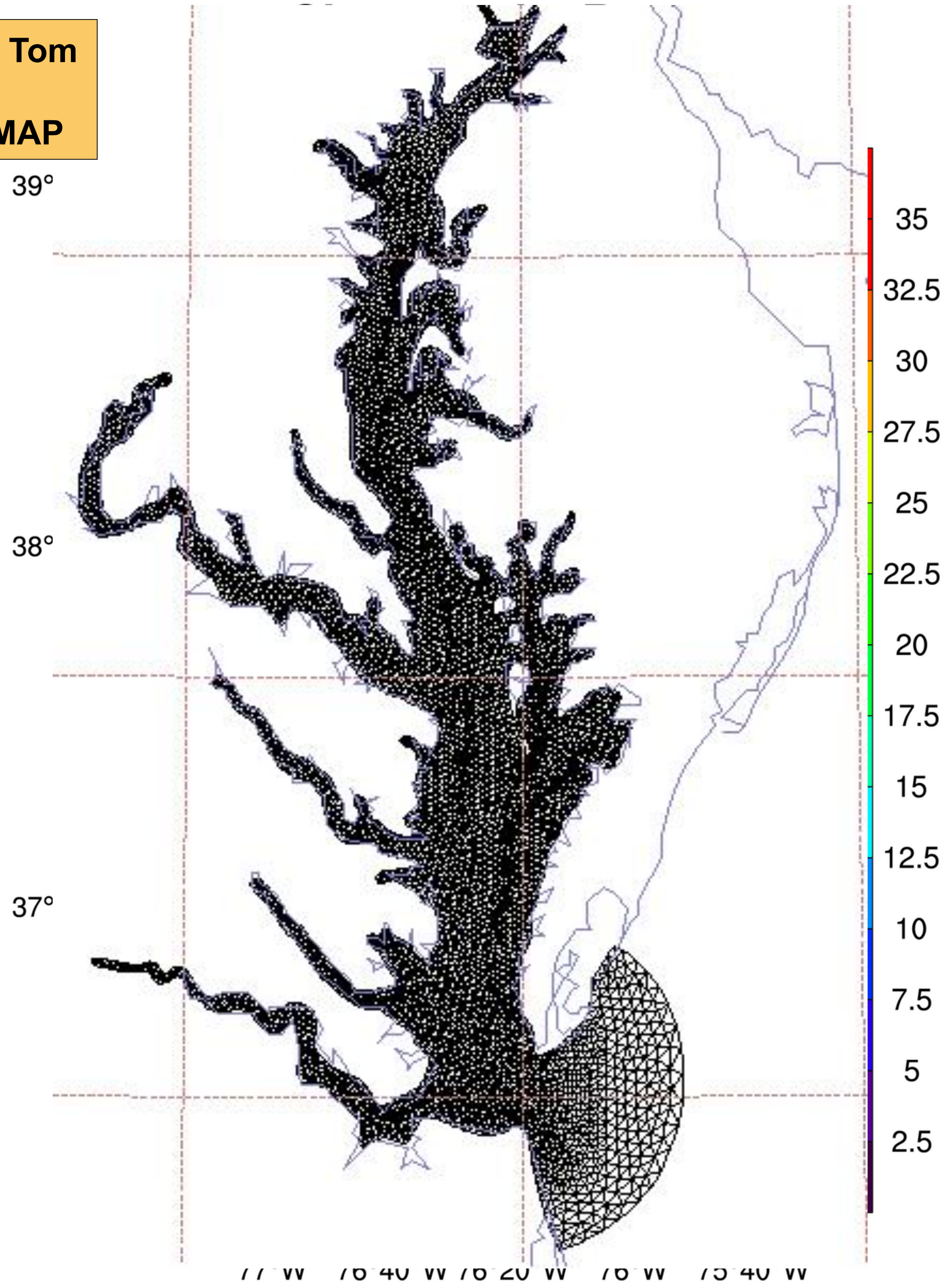
Annual Mean

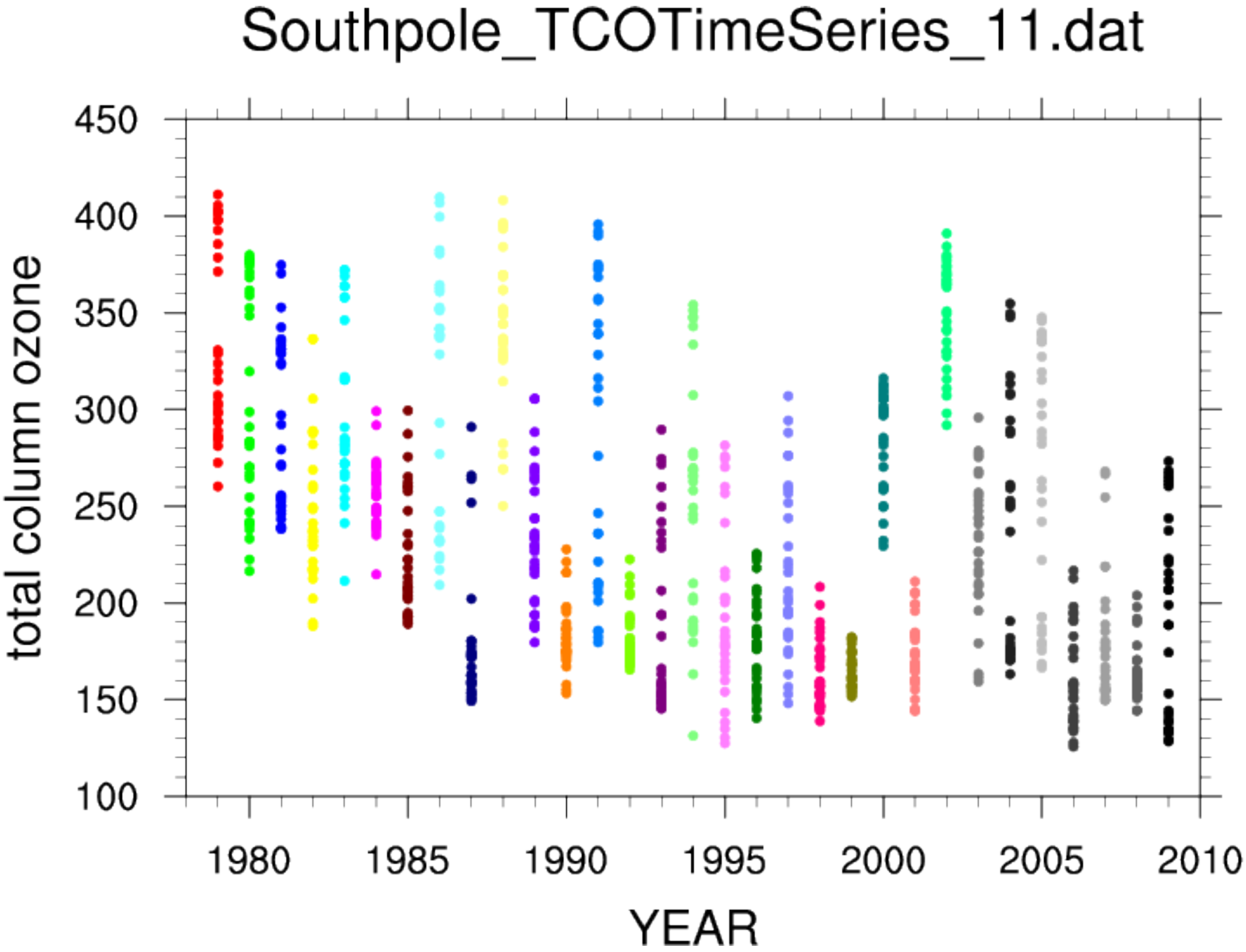
Cloud % Trend: SRES-A1B: 2000-2100 [% century^{-1}]

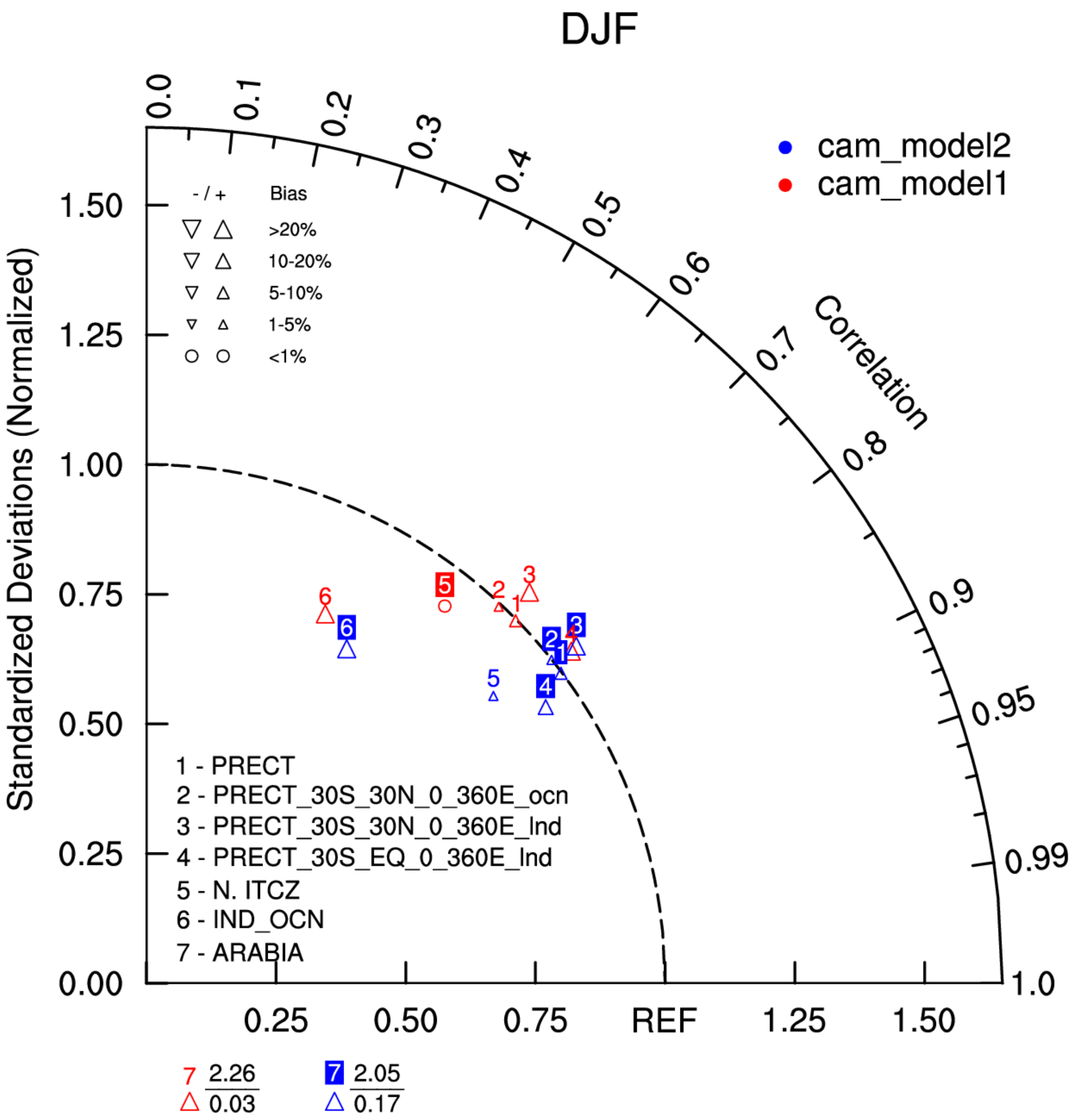


John Fasullo, NCAR/CGD

Triangular mesh from Tom Gross NOAA/NOS/CSDL/MMAP

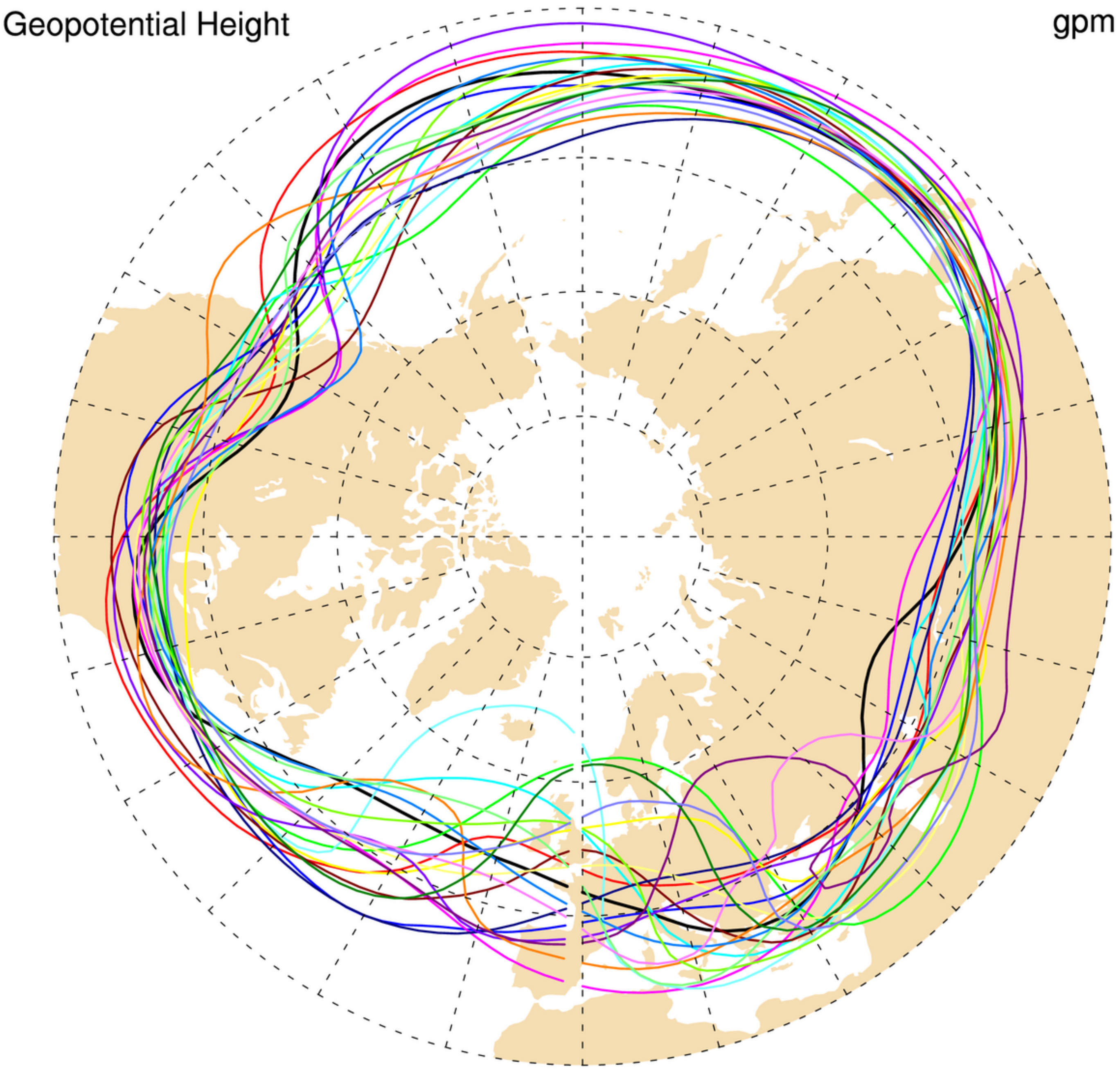



Southpole_TCOTimeSeries_11.dat
total column ozone
450
400
350
300
250
200
150
100
1980
1985
1990
1995
2000
2005
2010
YEAR

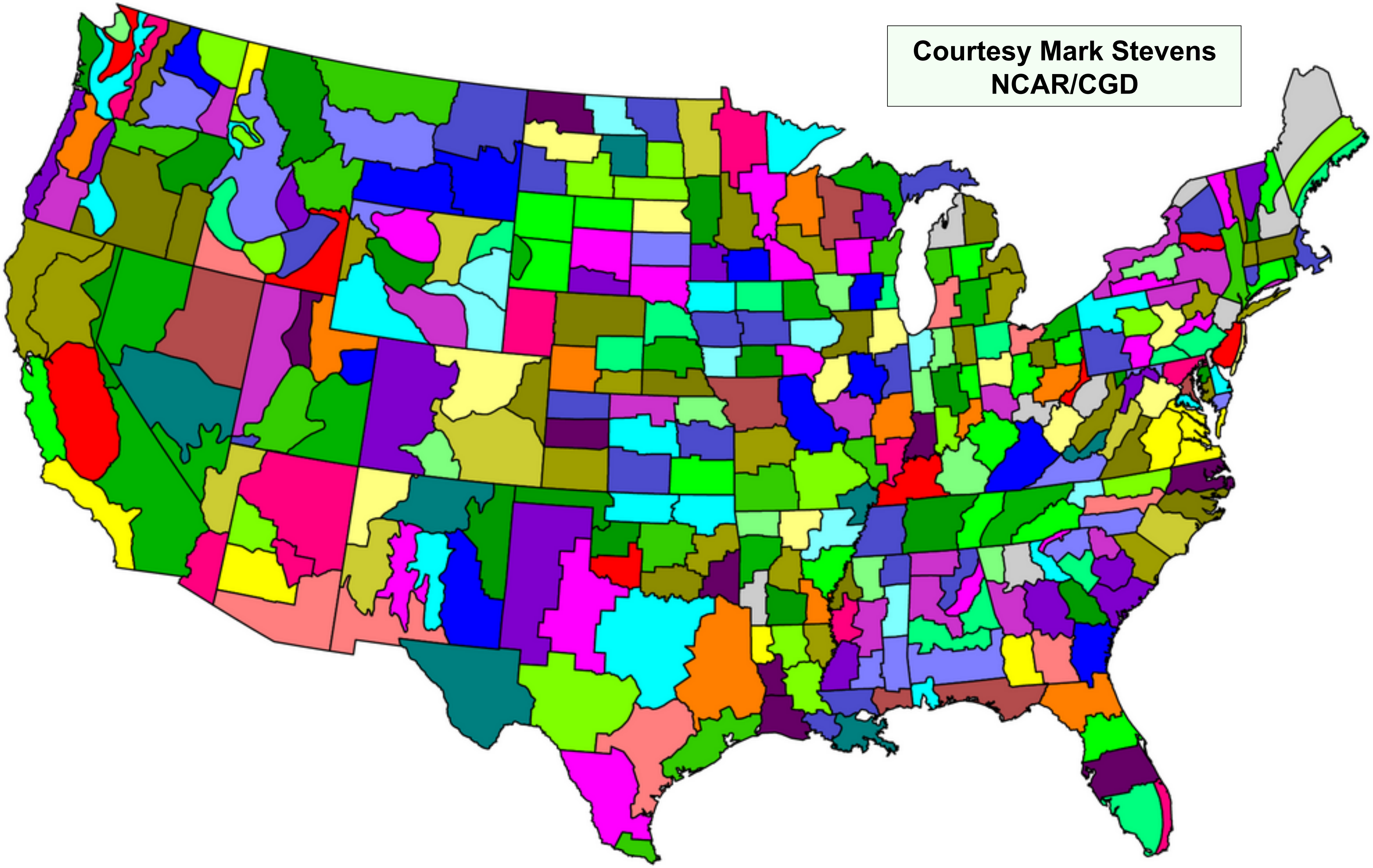


Taylor diagram Courtesy of Dennis Shea and Adam Phillips, CGD

Spaghetti-style contours



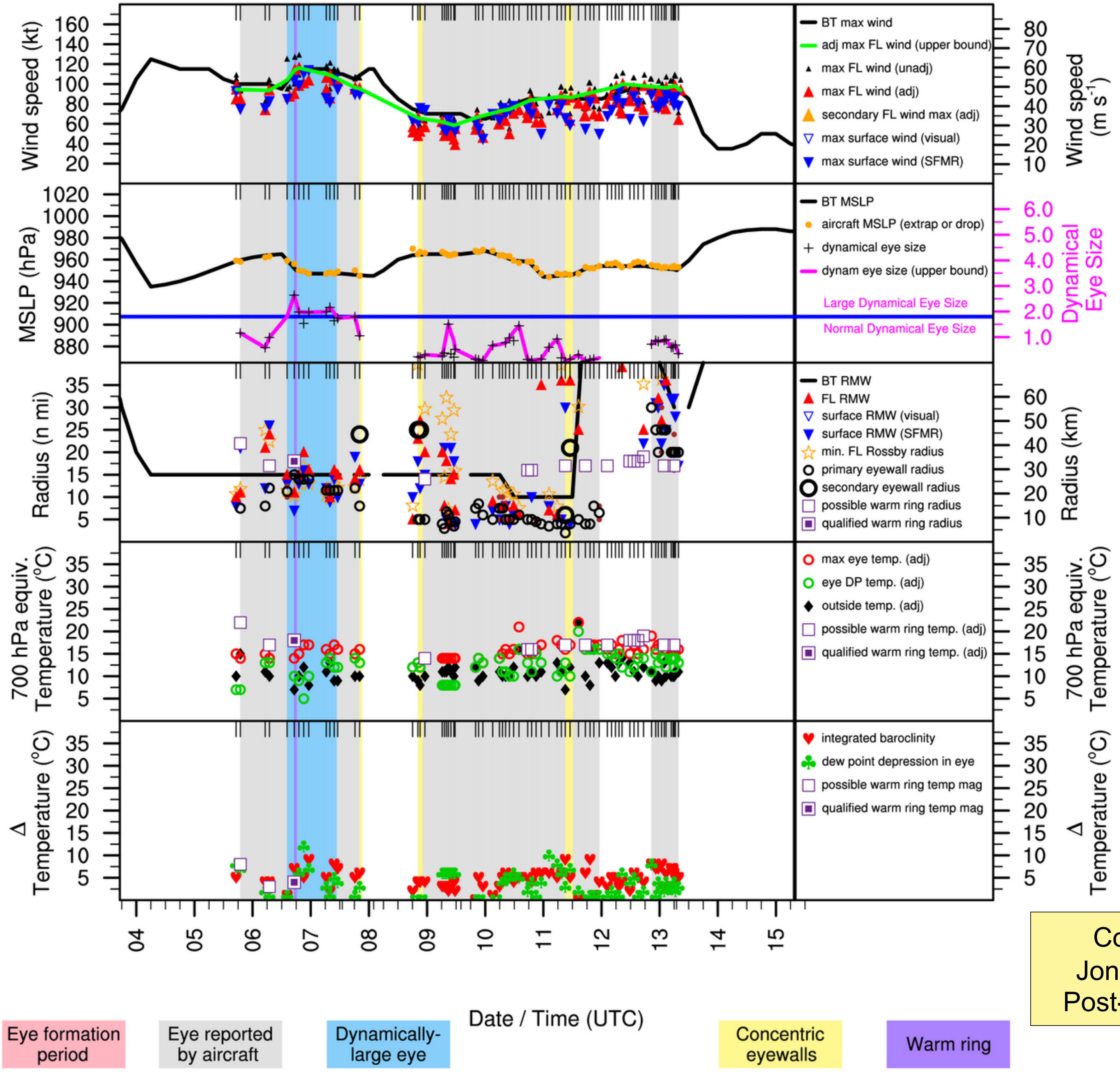
Climate divisions



Courtesy Mark Stevens
NCAR/CGD

Climate divisions are built into NCL and PyNGL

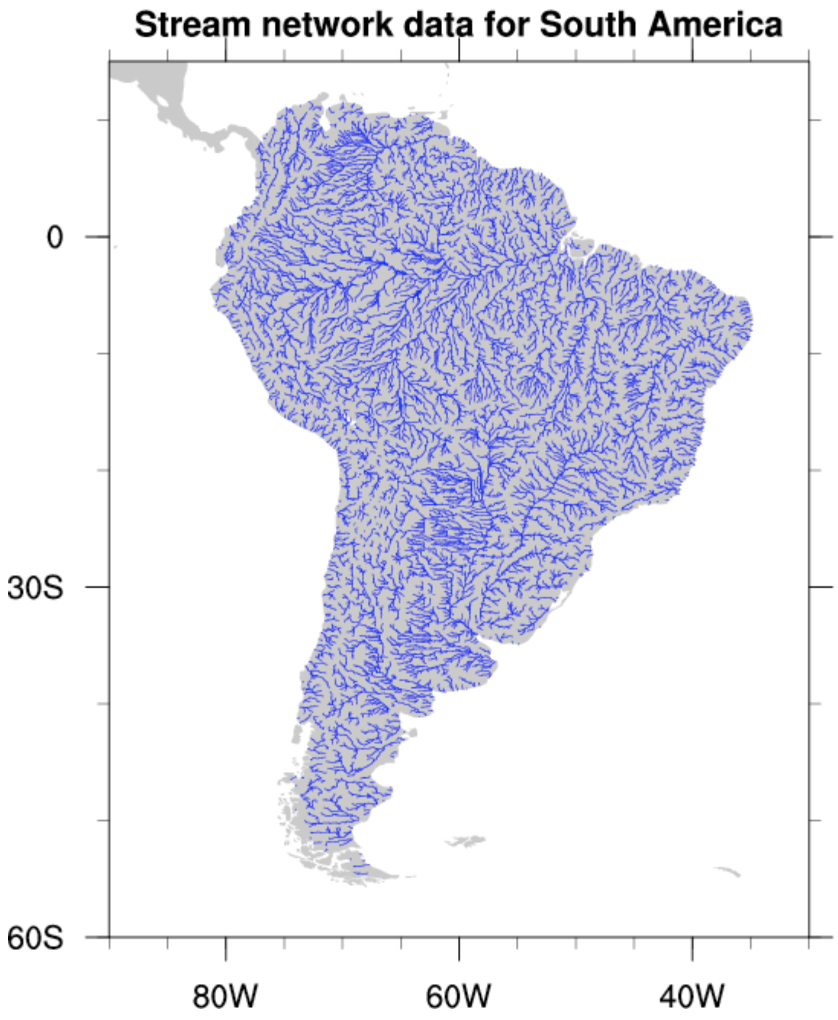
IKE (AL092008)

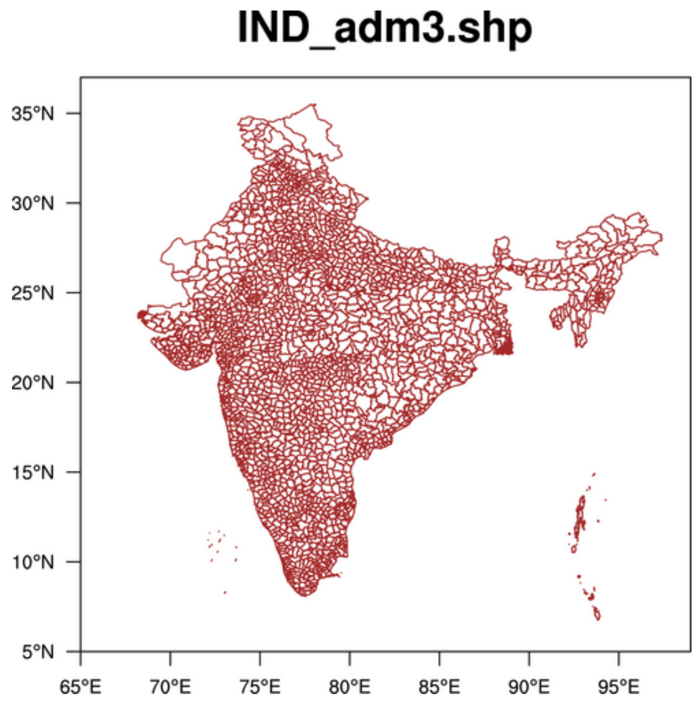


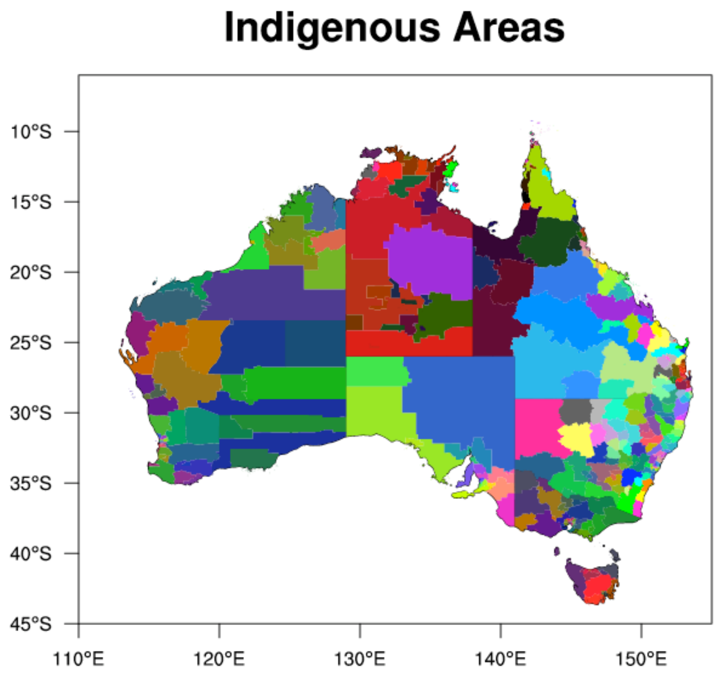
Courtesy of
Jonathan Vigh
Post-doc, NCAR

NCL has support for shapefiles, allowing you to use the numerous free shapefiles for adding your own map outlines

http://www.gadm.org







Topics

- Overview
- What's new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

What’s new in NCL V6.0.0

- *Released May 30, 2011 (many months of beta testing)*
- Major overhaul: can now create larger than 2 GB variables (on 64-bit systems)

```
tc = wrf_user_getvar(f,”tc”,-1) ; tc can be > 2 GB
```

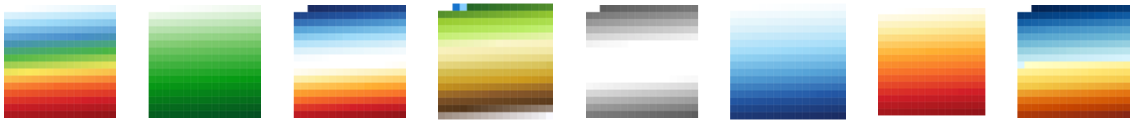
- ***Default missing values changed***
- Can delete multiple variables with “delete” command!

```
 slp = wrf_user_getvar(a,"slp",time)
tc2 = wrf_user_getvar(a,"T2",time)
u10 = wrf_user_getvar(a,"U10",time)
. . .
delete([/slp,tc2,u10/])
```

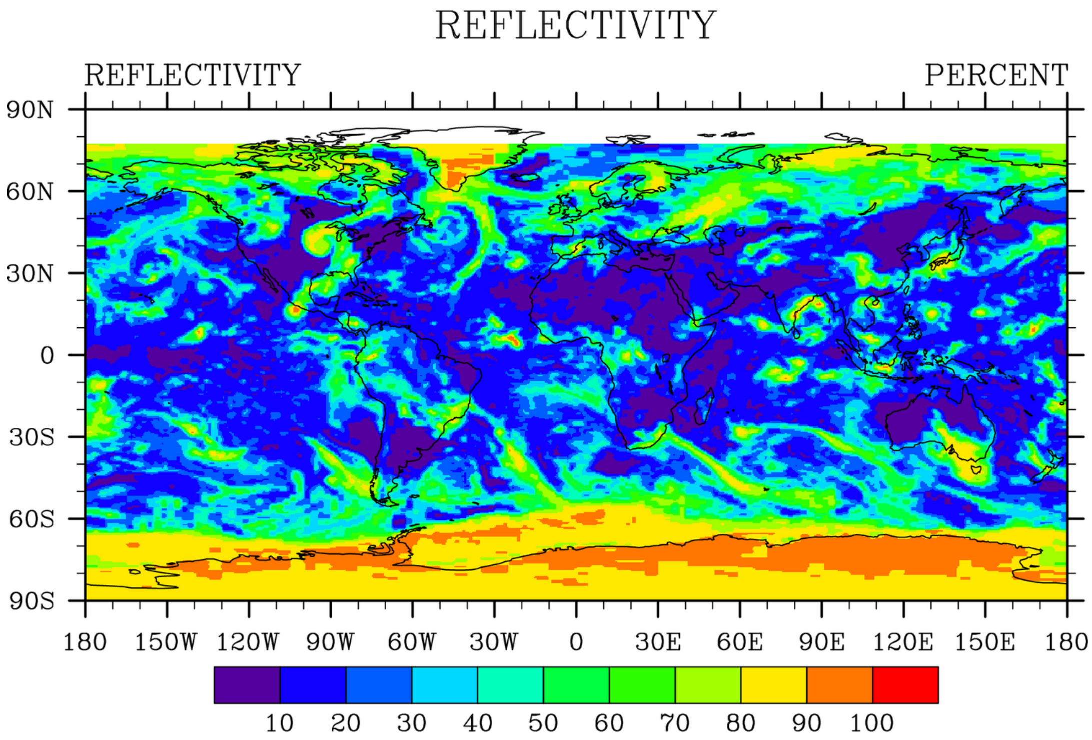
- Meaning of “byte” and “character” swapped, (“unsigned byte” added as new type)

What’s new in NCL V6.0.0 (cont’d)

- New functions
- Lots of bug and memory fixes
- New color tables



- HDF5 reader (alpha testing)



What’s new – WRF specific

- wrf_user_getvar, wrf_user_ij_to_ll, wrf_user_ll_to_ij, wrf_user_list_times can now take direct input from variable returned by addfiles variable:

```
fnames = systemfunc(“ls -1 wrfout*”) + “.nc”
f      = addfiles(fnames,”r”)
slp    = wrf_user_getvar(f,”slp”,-1)
```

- Experimental examples added to WRF-ARW online tutorial:
  - Moving nest domains
  - Wind roses

  http://www.mmm.ucar.edu/wrf/OnLineTutorial/index.htm

- In the pipeline: pressure/height interpolation code will be able to extrapolate below ground

What's coming in V6.1.0 & future

Next IPCC assessment report will place heavy demand on NCL

- Extremely large datasets
- Compute intensive calculations
- Comparing data from different models and grids
- NCL team in close dialog with researchers to handle these scalability issues

***Parvis** – a three year DOE project run by Argonne to parallelize components of NCL for ultra-large datasets*

- File input/output
  - HDF-EOS5, HDF5, NetCDF 4 (native)
  - Better compression
  - Parallel NetCDF support
- Computational
  - Faster algorithms
  - Specialized (ocean)
- Visualization
  - "Quick-look" utility
  - Direct png and geotiff output
  - Support for flexible color tables
  - Support for transparency
  - Vectors on a triangular mesh

Topics

- Overview
- What’s new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

To “run” an NCL script:

- *Install NCL and set up environment (covered later)*

- Make sure you have “~/.hluresfile”

- Create a file using a UNIX editor that contains NCL script commands, say, “myfile.ncl”. *Use examples on WRF-ARW online tutorial for help!*

- Run the file on the UNIX command line with:

```
    ncl myfile.ncl
```

- Look at output data or view graphical file

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl”
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”

begin
  print(“Hello, world”)

; Open a netCDF file and print its contents
  f = addfile(“wrfout_d01_2000-01-24_12:00:00.nc”,”r”)
  print(f)

; Read a variable and print its info
  slp = wrf_user_getvar(f,”slp”,0)
  printVarSummary(slp)

  wrf_smooth_2d( slp, 3 )                ; Smooth slp

  td2  =  wrf_user_getvar(f,”td2”,0)      ; td2 in C
  td_f = 1.8 * td2 + 32.                 ; Convert to F
  td_f@description = “Surface Dew Point Temp”
  td_f@units       = “F”

. .
end
```

begin/end are optional

Comments begin with “;”
Either on line by itself, or end of line

Open the file

This is like doing an “ncdump –h”

Retrieves WRF variable

Use print/printVarSummary for debugging

array arithmetic, like f90

To run this script (“wrf.ncl”) on UNIX command line, type:

```
ncl wrf.ncl
```

Scalar variable assignment

```
; Explicit scalar assignment

ndys = 30                                  ; integer
x_f  = 2983.599918                         ; float

pi   = 3.14159265358979d                   ; double

ll   = 32676l                              ; long
ishort = 10h                               ; short

done = True                                ; logical (False)
long_name = “Water Vapor”                  ; string
```

Use “literals” to force type

New unsigned types introduced in NCL V5.2.1 and 6.0.0

Mixing types

```
; Mixing types, “largest” type used
i = 7/10        ; integer (i=0)
x = 7/10.       ; float (x=0.7)

y = (22./7)/2d  ; double (1.571428537368774)

z = (i+5) * x   ; float (z=3.5)

; Use “+” for string concatenation
s1 = “hello”
s2 = “world”
s3 = s1 + “, “ + s2   ; s3 = “hello, world”

j = 2      ; Can mix strings and numerics
s = “var_“ + (j+1) + “_f”   ; s = “var_3_f”
```

Type conversions

```
; Can't change to "higher" type; use delete
ff = 1.5e20    ; float
ff = 1000      ; this is ok, still a float
ff = 1d36      ; not okay, "type mismatch"

delete(ff)
ff = 1d36      ; double
```

Old ones were "doubletofloat", "floattoint", etc.

```
;  lower  type
```

Note about "tointeger" issue in WRFUserARW.ncl

```
fx = tofloat(dx)             ; 345.789
ix = tointeger(dx)           ; 345
```

Arrays

- Row major. . . like C/C++ (*Fortran is column major)*
- Leftmost dimension varies the slowest, rightmost varies fastest (this matters for speed)
- Dimensions are numbered left to right (0,1,…)
- Use “dimsizes” function to get dimension sizes
- Indexes (subscripts) start at 0 (0 to n-1)
- Use parentheses to access elements:

```
dx = x(2) – x(1)   ; 3rd value minus 2nd value

; Assume Y is 3D (nx,ny,nz)
y1 = y(0,0,0)           ; first value of array
yn = y(nx-1,ny-1,nz-1)  ; last value of array
```

Array assignment: (/. . ./)

```
; 1D float array, 3 elements
lat = (/-80,0.,80/)


; string array, 4 elements
MM = (/"March","April","May","June"/)

; 3 x 2 double array
z = (/(/1,2d/),(/3,4/),(/9,8/)/)
```

```
; Create 3D double array, 10 x 64 x 128
x = new((/10,64,128/),double)

; Will be filled with 9.969209968386869e+36
```

Special functions for arrays

```
; Very useful “where” function
  q = where(z.gt.pi .and. z.lt.pi2, pi*z, 0.5*z)
```

```
; “num”, “any”, “all”

  npos = num (xTemp.gt.0.0)

  if (.not.any(string_array.eq.”hello world”)) then
    do something
  end if

  if (all(xTemp.lt.0)) then
    do something
  end if
```

```
; “ind” function, only on 1D arrays
  ii = ind(pr.lt.500 .and. pr.gt.60)
```

“where” is usually better than “ind”

Metadata

- Metadata is information about variables or files.
- In NetCDF-land, metadata consists of:
  - Attributes – *describes the file or variable* (units, history, grid type, long name, map projection)
  - Named dimensions – *describes the dimensions* ("time", "lat", "lon", "levels")
  - Coordinate arrays – *provides coordinate locations of data* (must be one-dimensional)
- WRF-ARW data doesn't normally have traditional 1D coordinate arrays. WRF coordinates are generally 2D or 3D and called XLAT/XLONG

Metadata (continued)

- The "_FillValue" attribute is a special one indicating a variable's missing value.

- When you do an "ncdump -h" or "ncl_filedump" on a NetCDF file, you see all the metadata

- NCL variables are based on this metadata model. Even if you read in a GRIB, HDF, or shapefile, it will "look" like a NetCDF file with attributes, named dimensions, and possibly coordinate arrays.

Missing values (_FillValue attribute)

- "_FillValue" is a special NetCDF *and* NCL reserved attribute for missing values
- Most NCL functions ignore _FillValue:

```
x             = (/1,2,3,-999,5/) ; no msg val yet
xavg          = avg(x)           ; = -197.6
x@_FillValue  = -999             ; now has a msg val
xavg          = avg(x)           ;(1+2+3+5)/4 = 2.75
```

- Must be same as type of variable

Use '@' to reference attributes

- "missing_value" attribute has **no** special status to NCL.
If "T" has "missing_value" attribute and no "_FillValue":

```
T@_FillValue = T@missing_value
```

- Best not to use zero as a _FillValue

"default_fillvalue" – returns default missing value for the given type
"set_default_fillvalue" – change the default missing value for the given type

"print" / "printVarSummary" will print _FillValue value.

"print" is very verbose

NCL default missing values

NCL type	Old	New	Special Note
integer	-999	-2147483647	
float	-999.	9.96921e+36	
double	-9999.	9.969209968386869e+36	
string	“missing”	“missing”	
short	-99	-32767	
byte	0xff	-127	now signed
ubyte	---	255	(new in 6.0.0) unsigned
character	0	0x00	now unsigned
logical	Missing	Missing	

```
fmsg = default_fillvalue(“float”)
```

Missing value functions

- Use any, all, and ismissing functions to query a variable for missing values:

```
if (.not.any(ismissing(T))) then
  do something
end if
if (all(ismissing(T))) then
  do something
end if
```

- Use num & ismissing to count missing values:

```
nmsg = num(ismissing(T))
```

- Use the new "default_fillvalue" and "set_default_fillvalue" if needed

File and variable attributes

```
; Use the “@” symbol to get at global file attributes.
f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
print(f@TITLE)          ; “OUTPUT FROM WRF V2.1.2 MODEL”
print(f@START_DATE)     ; “2005-08-26_00:00:00”
print(f@MAP_PROJ)       ; 3
```

```
; Use the “@” symbol to get at variable attributes too.
uvmet = wrf_user_getvar(f, “uvmet”, 0)
print(uvmet@units)           ; “m s-1”
print(uvmet@description)     ; “u,v met velocity”
```

```
; Use “isatt” to test for an attribute first.
if(isatt(uvmet,”units”)) then
  print(“The units of uvmet are ‘” + uvmet@units + “’”)
end if

(0)     The units of uvmet are ‘m s-1’
```

Arithmetic operations on arrays, like f90

- May not need to loop over arrays to do calculations
- Arrays need to be same size, but scalars can be used any time
- Highest “type” will be assigned to variable on left of “=”

```
; Can do arithmetic like Fortran 90
  ch4 = ch4 * 1e6     ; convert to ppm, assign to same var

  A = data_DJF - data_JJA   ; data_DJF/data_JJA must be same size

  zlev = (-7*log(lev/10^3))     ; evaluated as
                                ; (-7)*log(lev/(10^3))
```

Metadata not copied to A or zlev

```
; Use “conform” to promote an array
; “Twk” is (time,lat,lon,lev), “ptp” is (lat,lon)

  ptropWk = conform(Twk, ptp, (/1,2/)) ; time,lat,lon,lev
```

Array reorder, reshape, reverse

```
; Reshaping an array

  t1D = ndtooned(T)                ; Convert to 1D array
  t2D = onedtond(t1D, (/N,M/) )    ; Convert to N x M array
```

Requires named dimensions be present

```
; Reordering an array

; Let T(time,lat,lon)
  t = T(lat|:,lon|:,time|:)    ; Can’t assign to same var
```

```
; Reversing dimensions of an array

; Let T(lev,lat,lon)
  T = T(::-1,:,:)     ; Will reverse coordinate array too
```

Functions for manipulating arrays

http://www.ncl.ucar.edu/Document/Functions/array_manip.shtml

Array Subscripting

- Three kinds of array subscripting
  1. Index (uses ‘:’ and ‘::’)
  2. Coordinate (uses curly braces ‘{‘ and ‘}’)
  3. Named dimensions (uses ‘!’)
- Most WRF-ARW data does not have coordinate arrays, so can’t use #2
- You can mix subscripting types in one variable
- Be aware of dimension reduction
- Index subscripting is 0-based (Fortran by default is 1-based)

http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/NclVariables.shtml#Subscripts

Array index subscripting, : and ::

```
; Consider T(ntime x nlat x nlon)
   t = T                    ; copies metadata, don’t use T(:,:,:)
   t = (/T/)                ; doesn’t copy metadata
                            ; (_FillValue is retained)
```

```
; The following creates 2D array “t”
   t = T(0,:,::5)           ; 1st time index, all lat, every 5th lon
                            ; (nlat x nlon/5)
```

```
   t = T(0,::-1,:50)        ; 1st time index, reverse lat,
                            ; first 51 lons (nlat x 51)
```

```
   t = T(:1,45,10:20)       ; 1st two time indices, 46th index of lat,
                            ; 11th-21st indices of lon (2 x 11)
```

```
; To prevent dimension reduction
   t = T(0:0,:,::5)                 ; 1 x nlat x nlon/5
   t = T(:1,45:45,10:20)            ; 2 x 1 x 21
```

Topics

- Overview
- What's new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Opening and examining a WRF output file

```
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
print(f)
```

WRF files don't have ".nc" suffix; must add here.

print(f) results

```
Variable: f (file variable)

filename:       wrfout_d01_2005-08-27_00:00:00
path:   wrfout_d01_2005-08-27_00:00:00
   file global attributes:
      TITLE :  OUTPUT FROM WRF V2.1.2 MODEL
      START_DATE : 2005-08-26_00:00:00
      SIMULATION_START_DATE : 2005-08-26_00:00:00
      WEST-EAST_GRID_DIMENSION : 400
      SOUTH-NORTH_GRID_DIMENSION : 301
      BOTTOM-TOP_GRID_DIMENSION : 35
      DX : 12000
      DY : 12000
      GRIDTYPE : C
      DYN_OPT : 2
      DIFF_OPT : 1 KM_OPT : 4
      DAMP_OPT : 0
```

global attributes

print(f) results (continued)

more global attrs

```
KHDIF :  0
KVDIF :  0
MP_PHYSICS : 3
RA_LW_PHYSICS : 1
RA_SW_PHYSICS : 1
SF_SFCLAY_PHYSICS : 1
SF_SURFACE_PHYSICS : 1
BL_PBL_PHYSICS : 1
CU_PHYSICS : 1
WEST-EAST_PATCH_START_UNSTAG : 1
WEST-EAST_PATCH_END_UNSTAG : 399
WEST-EAST_PATCH_START_STAG : 1
WEST-EAST_PATCH_END_STAG : 400
SOUTH-NORTH_PATCH_START_UNSTAG : 1
SOUTH-NORTH_PATCH_END_UNSTAG : 300
SOUTH-NORTH_PATCH_START_STAG : 1
SOUTH-NORTH_PATCH_END_STAG : 301
BOTTOM-TOP_PATCH_START_UNSTAG : 1
BOTTOM-TOP_PATCH_END_UNSTAG : 34
BOTTOM-TOP_PATCH_START_STAG : 1
BOTTOM-TOP_PATCH_END_STAG : 35
GRID_ID : 1
PARENT_ID : 0
I_PARENT_START : 0
J_PARENT_START : 0
PARENT_GRID_RATIO : 1
DT : 60
```

print(f) results (continued)

```
. . .
 dimensions:
    Time = 1   // unlimited
    DateStrLen = 19
    west_east = 399
    south_north = 300
    west_east_stag = 400
    bottom_top = 34
    south_north_stag = 301
    bottom_top_stag = 35
    ext_scalar = 1
    soil_layers_stag = 5
 variables:
    character Times ( Time, DateStrLen )
    float LU_INDEX ( Time, south_north, west_east )
       FieldType :     104
       MemoryOrder :   XY
       description :   LAND USE CATEGORY
       units :
       stagger :
    float U ( Time, bottom_top, south_north, west_east_stag )
       FieldType :     104
       MemoryOrder :   XYZ
       description :   x-wind component
       units :         m s-1
       stagger :       X
```

variable dimension names

variables

print(f) results (continued)

```
float V ( Time, bottom_top, south_north_stag, west_east )
   FieldType :    104
   MemoryOrder :  XYZ
   description :  y-wind component
   units :        m s-1
   stagger :      Y

float W ( Time, bottom_top_stag, south_north, west_east )
   FieldType :    104
   MemoryOrder :  XYZ
   description :  z-wind component
   units :        m s-1
   stagger :      Z

float PH ( Time, bottom_top_stag, south_north, west_east )
   FieldType :    104
   MemoryOrder :  XYZ
   description :  perturbation geopotential
   units :        m2 s-2
   stagger :      Z

float PHB ( Time, bottom_top_stag, south_north, west_east )
   FieldType :    104
   MemoryOrder :  XYZ
   description :  base-state geopotential
   units :        m2 s-2
   stagger :      Z
```

more variables

Using “ncl_filedump” on UNIX command line

Don’t need to write a script to quickly look at a WRF file. On the UNIX command line, type:

```
ncl_filedump –h

ncl_filedump wrfout_d01_2005-08-27_00:00:00.nc

ncl_filedump –v RAINC wrfout_d01_2005-08-27_00:00:00.nc
```

Can use ncl_filedump on other files that NCL’s “addfile” supports: GRIB 1 and 2, HDF4, HDF-EOS2, etc

```
ncl_filedump TES-Aura_L3-ATM-TEMP_r0000003459_F01_05.he5

ncl_filedump z_tigge_c_rjtd_20061119120000_0072_sl_glob_prod.grb2

ncl_filedump states.shp
```

Two ways to read a variable off a file

- Use “->” syntax

- Use “wrf_user_getvar” function
  - Developed to make it easier to get derived variables
  - It is an NCL script function, so must load “WRFUserARW.ncl” script
  - You can modify this script (more later)
  - Only use with WRF-ARW data

Reading (and examining) a variable off a file (method 1)

```
f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
u = f->U
printVarSummary(u)
; print(u)            ; Same as printVarSummary, but includes values
```

```
Variable: u
Type: float
Total Size: 16320000 bytes
            4080000 values
Number of Dimensions: 4
Dimensions and sizes:   [Time | 1] x [bottom_top | 34] x [south_north
| 300] x [west_east_stag | 400]
Coordinates:
Number Of Attributes: 5
  FieldType :    104
  MemoryOrder : XYZ
  description : x-wind component
  units :       m s-1
  stagger :     X
```

printVarSummary(u) results

named dimensions

no coordinate arrays

variable attributes

Reading (and examining) a variable off a file (method 2)

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl”
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”

f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
slp = wrf_user_getvar(f,”slp”,0)
printVarSummary(slp)
```

printVarSummary(slp) results

```
Variable: slp
Type: float
Total Size: 478800 bytes
            119700 values
Number of Dimensions: 2
Dimensions and sizes:   [south_north | 300] x [west_east | 399]
Coordinates:
Number Of Attributes: 5
  description : Sea Level Pressure
  units :       hPa
  FieldType :   104
  MemoryOrder : XYZ
  stagger :
```

Further querying a variable

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl”
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”

f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
slp = wrf_user_getvar(f,"slp",0)
```

```
print(dimsizes(slp))   ; Print dimension sizes of slp
print(min(slp))        ; Print minimum of slp
print(max(slp))        ; Print maximum of slp
print(typeof(slp))     ; Print type of slp
print(getvaratts(slp)) ; Print attributes of slp
```

```
; Can assign to variables
dims    = dimsizes(slp)
slp_min = min(slp)
slp_max = max(slp)
attrs   = getvaratts(slp)
slp_avg = avg(slp)
```

Most of above info is printed as part of printVarSummary procedure

Creating a new variable & adding attributes

```
f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
td2 = wrf_user_getvar(f,”td2”,0)   ; Units are “C”

td_f = 1.8 * td2 + 32.    ; Can operate on whole array
td_f@units          = “F”    ; Add some attributes
td_f@description = “Surface Dew Point Temp”


; To preserve metadata
td_f = td2 ; Easy way to copy metadata, can be expensive
td_f = 1.8 * td2 + 32
td_f@description = “Surface Dew Point Temperature”
td_f@units          = “F”
printVarSummary(td_f)

; To write new variable to an existing file
f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”w”)
. . .
f->td_f = td_f    ; Write “td_f” to same file
```

Topics

- Overview
- What’s new
- NCL language basics
- File input/output
- **Data Analysis**
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

WRF-NCL Functions

- Two kinds:
  - ***Built-in*** - mainly functions to calculate diagnostics. *Seldom need to use these directly.*

```
slp = wrf_slp(z, tk, P, QVAPOR)
```

  - ***“WRFUserARW.ncl”*** - developed to make it easier to calculate derived variables and generate plots, calls some built-in functions

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”
slp = wrf_user_getvar(f,”slp”,time)
```

http://www.ncl.ucar.edu/Document/Functions/wrf.shtml

WRF-NCL built-in functions

Can use NCL built-in functions, in place of wrf_user_getvar, not always recommended!

```
T      = f->T(time,:,:,:)
P      = f->P(time,:,:,:)
PB     = f->PB(time,:,:,:)
QVAPOR = f->QVAPOR(time,:,:,:)
PH     = f->PH(time,:,:,:)
PHB    = f->PHB(time,:,:,:)
T = T + 300.
P = P + PB
QVAPOR = QVAPOR > 0.0 ; Set anything <= 0 to msg
PH    = ( PH + PHB ) / 9.81

z   = wrf_user_unstagger(PH,PH@stagger)
tk  = wrf_tk( P , T )
slp = wrf_slp( z, tk, P, QVAPOR )
```

Replace with single call

```
slp = wrf_user_getvar(f,”slp”,time)
```

WRF-NCL “*WRFUserARW.ncl*” functions

wrf_user_getvar - Get fields from input file

```
ter = wrf_user_getvar(a,"HGT",0)
t2  = wrf_user_getvar(a,"T2",-1)
slp = wrf_user_getvar(a,"slp",1)
```

wrf_user_getvar
is user-modifiable!
(more later)

Diagnostics

avo/pvo	Absolute/Potential Vorticity
cape_2d	2D mcape/mcin/lcl/lfc
cape_3d	3D cape/cin
dbz/mdbz	Reflectivity
geopt/geopotential	Geopotential
p/pres/pressure	Pressure
rh/rh2	Relative Humidity
slp	Sea Level Pressure
td/td2	Dew Point Temperature
tc/tk	Temperature
th/theta	Potential Temperature
ua/va/wa	Wind on mass points
uvmet/uvmet10	U/V components of wind rotated to earth coords
z/height	Height

http://www.ncl.ucar.edu/Document/Functions/WRF_arw/

Other WRF-NCL “*WRFUserARW.ncl*” functions

- wrf_user_list_times
  Get list of times available in input file

```
times = wrf_user_list_times (f)
```

- wrf_user_unstagger
  Unstaggers an array

```
ua = wrf_user_unstagger (U, “X”)
ua = wrf_user_getvar(f,”ua”,time)
```

- wrf_map_overlays
  Draws plots over a map background

```
map = wrf_map_overlays(a, wks, \
   (/contour,vector/), pltres, mpres)
```

Other WRF-NCL "*WRFUserARW.ncl*" functions

- wrf_user_intrp3d

  Interpolate horizontally to a given pressure or height level

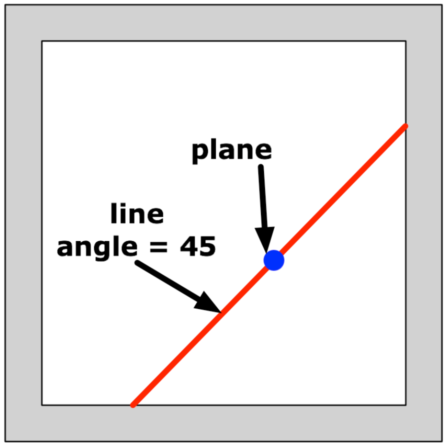
  Interpolate vertically *(pressure/height)*, along a given line

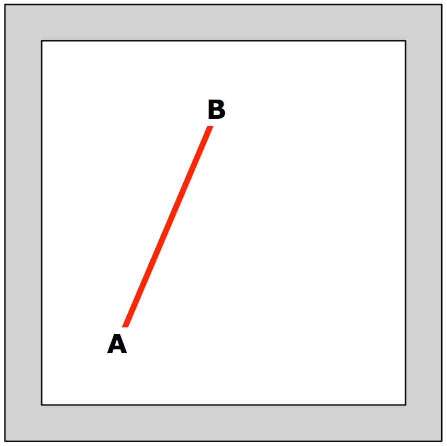
```
tc_plane = wrf_user_intrp3d( tc, p, "v", (/30,25/), \
                             45., False )
```

- wrf_user_intrp2d

  Interpolate along a given line

```
t2_plane = wrf_user_intrp2d(t2, (/12,10,25,45/), \
                            0., True)
```





Other WRF-NCL “*WRFUserARW.ncl*” functions

- wrf_user_ll_to_ij / wrf_user_ij_to_ll
  Convert: lat/lon ij

```
locij = wrf_user_ll_to_ij (f, 100., 40., res)
locll = wrf_user_ij_to_ll (f, (/10, 12/), \
                              (/40, 50/), res)
```

res@useTime - Default is 0
Set to a time index value if you want the reference longitude/latitudes to come from a different time index - only use this for moving nest output which has been stored in a single file.

res@returnInt - Default is True
If set to False, the return values will be real.
(wrf_user_ll_to_ij *only)*

Modifying wrf_user_getvar function

- Copy the following file to your own directory:
  "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

- Edit your copy and look for line that starts with:

```
function wrf_user_getvar
```

- Before the lines:

```
  return(var)
end
```

Add these lines, replacing "*newvar*" as appropriate:

```
if( variable .eq. "newvar" ) then
  . . .fill in code here. . .
  return(newvar)
end if
```

Modifying wrf_user_getvar function (cont’d)

- To use the new version of this function, you can do one of two things:

1. Load your modified script instead of the system one:

```
load “./WRFUserARW.ncl”
xxx = wrf_user_getvar(f,”XXX”,0)
```

2. Remove all but the modified “wrf_user_getvar” function from your copy, rename the function (“wrf_user_getvar2”), and rename the file (“my_new_script.ncl”). To use the new function, you need to load the above script and your new script:

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”
load “./my_new_script.ncl”

xxx = wrf_user_getvar2(f,”XXX”,0)
```

Topics

- Overview
- What’s new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Links for visualization scripts

- WRF-ARW online tutorial
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/index.htm

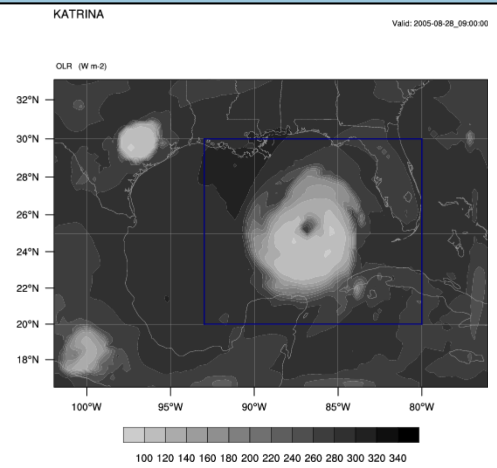
- NCL/WRF examples page
  http://www.ncl.ucar.edu/Applications/wrf.shtml

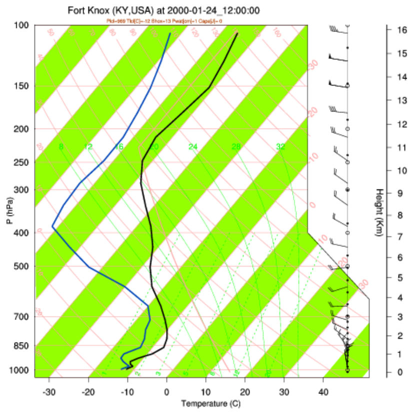
  NCL Home Page -> Examples -> WRF

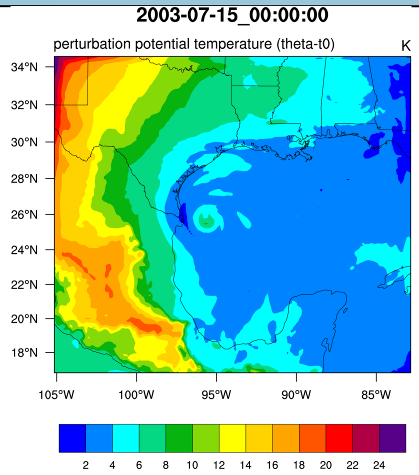
- Description of WRF-NCL functions
  http://www.ncl.ucar.edu/Document/Functions/wrf.shtml

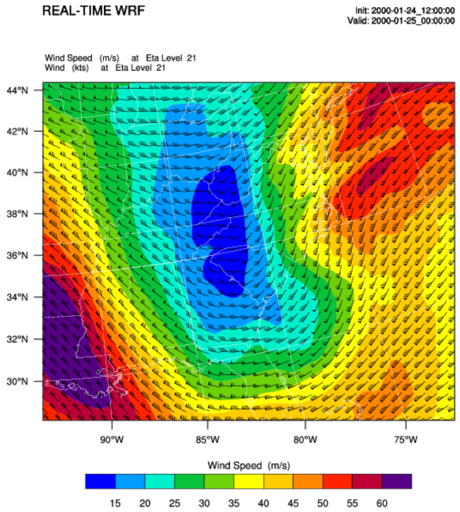
  NCL Home Page -> Functions -> Category -> WRF

Step-by-step WRF-ARW visualizations

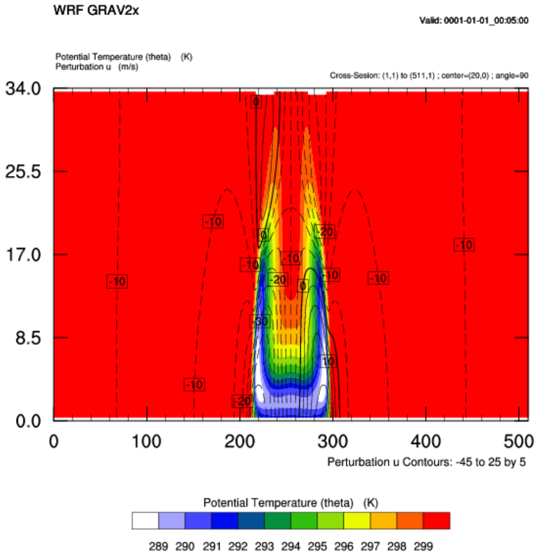


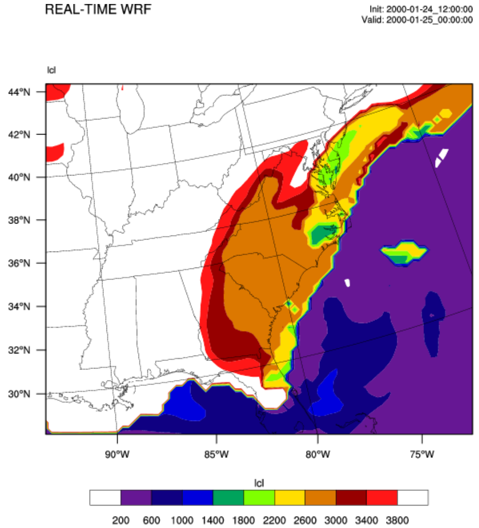





REAL-TIME WRF
Init: 2000-01-24_12:00:00
Valid: 2000-01-25_00:00:00
Wind Speed (m/s) at Eta Level 21
Wind (kts) at Eta Level 21
44°N
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
Wind Speed (m/s)
15 20 25 30 35 40 45 50 55 60

OUTPUT FROM WRF V3.0.1.1 MODEL
WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1





OUTPUT FROM WRF V3.0.1.1 MODEL
WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

Step-by-step: filled contours using wrf_xxxx

```
; Load the necessary scripts
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl”
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”

; Open a file and read a variable
f = addfile(“wrfout_d01_2005-08-27_00:00:00.nc”,”r”)
hgt = wrf_user_getvar(f,”HGT”,0)

wks = gsn_open_wks(“ps”,”hgt”)   ; “hgt.ps”

; Set some plotting resources
res = True
res@cnFillOn = True
```

These are plot options, also known as “resources”

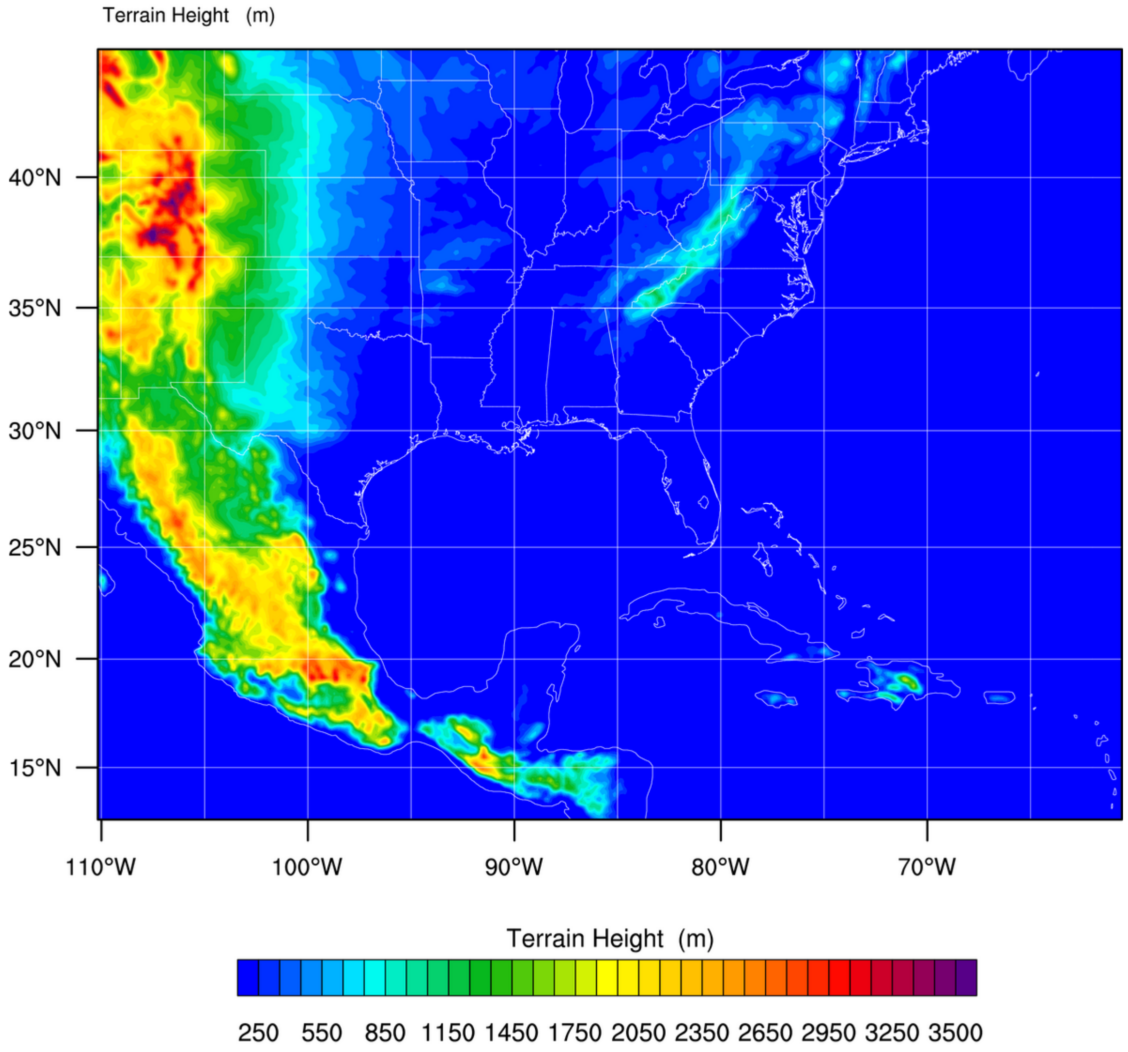
```
; These are special wrf_xxxx resources
res@MainTitle              = "GEOGRID FIELDS”
res@ContourParameters = (/ 250., 3500., 100. /)
contour = wrf_contour(f,wks,hgt,res)

pltres  = True
mpres   = True
plot    = wrf_map_overlays(f,wks,contour,pltres,mpres)
```

wrf_map_overlays looks at file to determine map projection

GEOGRID FIELDS

Init: 2005-08-26_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

Step-by-step: line/fill contours, vectors

```
. . .
slp = wrf_user_getvar(f,"slp",0)
t2  = wrf_user_getvar(f,"T2",0)
u10 = wrf_user_getvar(f,"U10",0)
v10 = wrf_user_getvar(f,"U10",0)

wks = gsn_open_wks(“ps”,”wrf”)   ; Open “wrf.ps” file for output
```

```
; Line contours
os                   = True
os@cnLineColor       = “NavyBlue”
os@cnLineThicknessF = 2.0
c_slp                = wrf_contour(f,wks,slp,os)
```

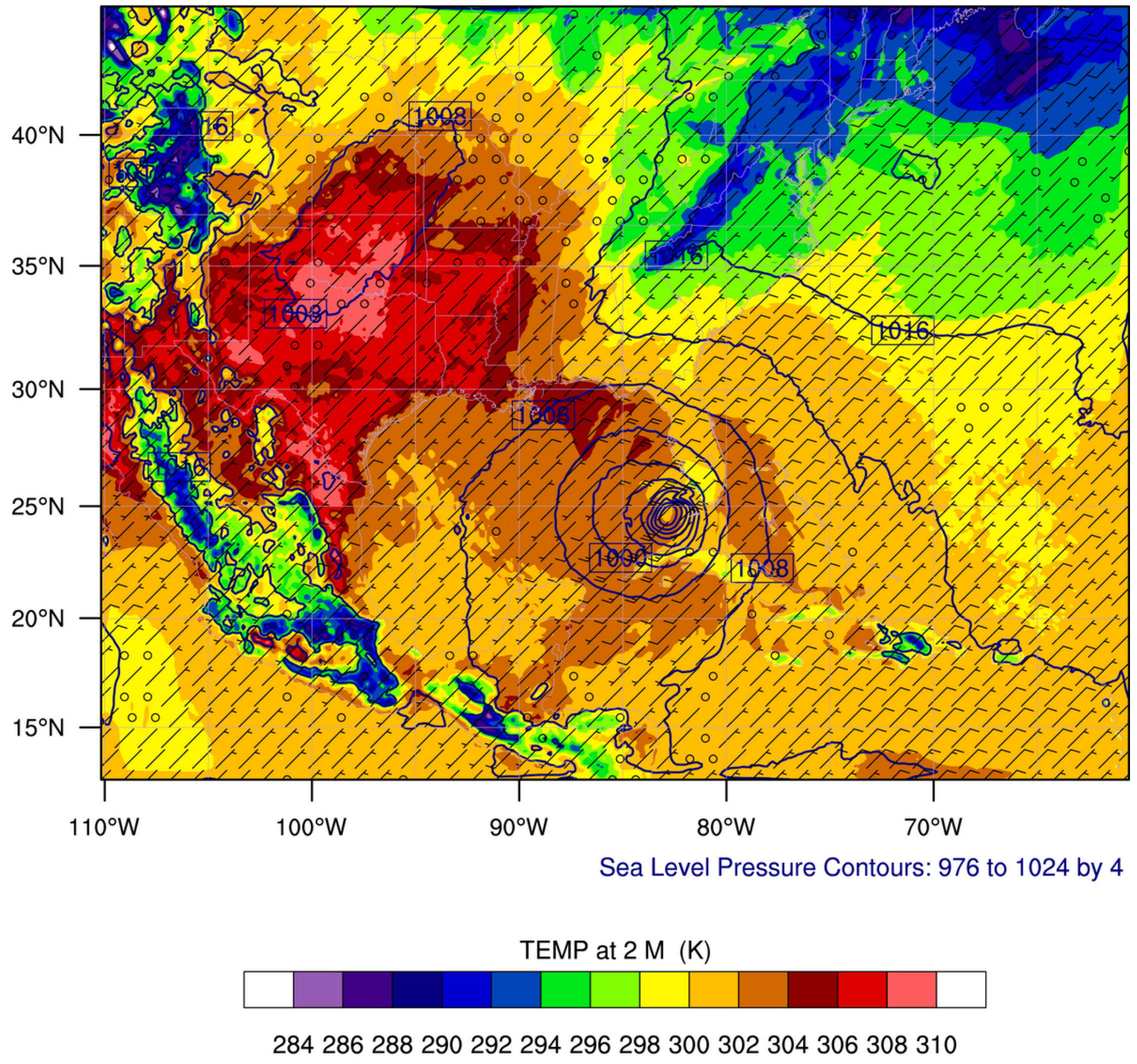
```
; Filled contours
ot               = True
ot@cnFillOn      = True
c_tc             = wrf_contour(f,wks,t2,ot)
```

```
; Vectors
ov               = True
ov@NumVectors    = 47
vec              = wrf_vector(f,wks,u10,v10,ov)
```

```
; Overlay everything on a map
mpres   = True
pltres = True
plot = wrf_map_overlays(f,wks,(/c_tc,c_slp,vec/),pltres, mpres)
```

Init: 2005-08-26_00:00:00

TEMP at 2 M (K)
Sea Level Pressure (hPa)
U at 10 M (m s-1)



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

wrf_contour/wrf_vector

Create line/shaded/filled contours and vectors

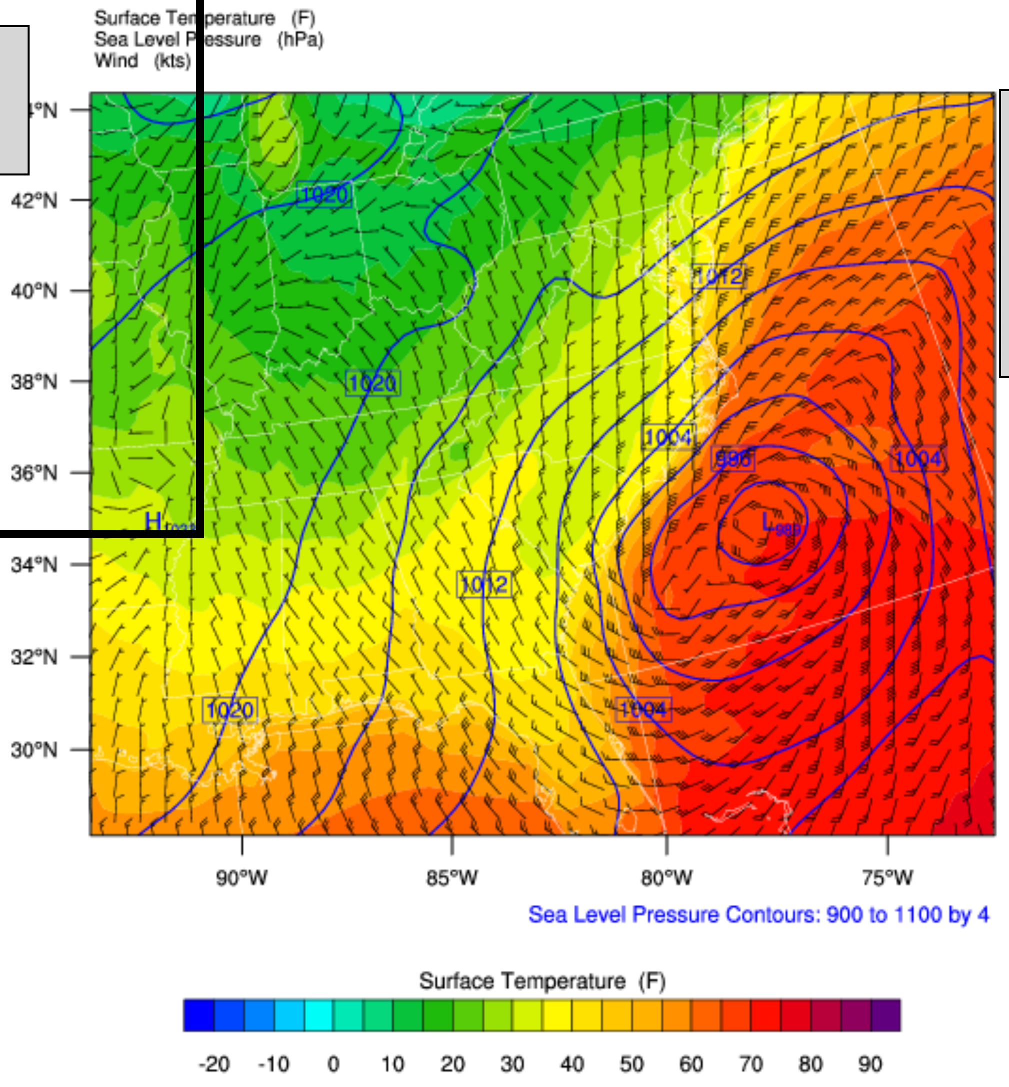
```
contour = wrf_contour(f, wks, ter, copts)
vector = wrf_vector(f, wks, u, v, vopts)
```

opts@MainTitle	Main title on the plot
opts@MainTitlePos	Main title position (default=left)
opts@NoHeaderFooter	Turn off headers & footers (default=False)
opts@Footer	Add model information as a footer (default=True)
opts@InitTime	Plot initial time on graphic (default=True)
opts@ValidTime	Plot valid time on graphic (default=True)
opts@TimeLabel	Label to use for valid time
opts@TimePos	Time position (default=right)
opts@ContourParameters	Contour parameters
opts@FieldTitle	Overwrite the field title
opts@UnitLabel	Overwrite the field units
opts@PlotLevelID	Add level information to field title
opts@NumVectors	Density of wind vector *(wrf_vector) (default=25)*

opts@MainTitle
opts@MainTitlePos

REAL-TIME WRF

Init: 2000-01-24_12:00:00
Valid: 2000-01-25_00:00:00



opts@InitTime
opts@ValidTime
opts@TimeLabel
opts@TimePos

Resources for wrf_contour & wrf_vector

opts@NoHeaderFooter

OUTPUT FROM WRF V2.2 MODEL
Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1 ; WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km

opts@Footer

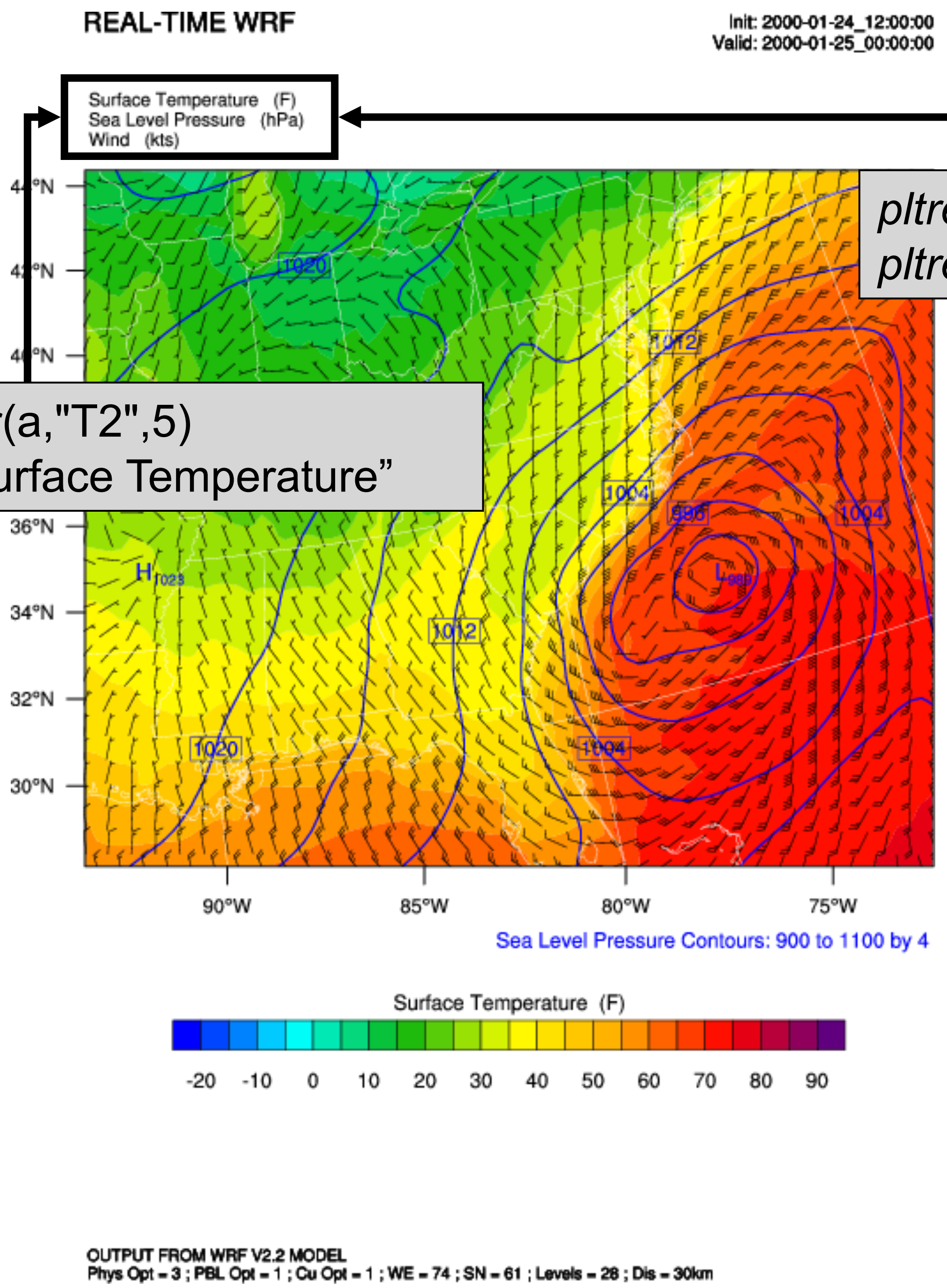
wrf_map_overlays/wrf_overlays

Overlay plots created with wrf_contour and wrf_vector

```
plot = wrf_map_overlays (f, wks, (/contour,vector/), \
                               pltres, mpres)
plot = wrf_overlays (f, wks, (/contour,vector/), \
                          pltres)
```

- mpres@mpGeophysicalLineColor; mpres@mpNationalLineColor; mpres@mpUSStateLineColor; mpres@mpGridLineColor; mpres@mpLimbLineColor; mpres@mpPerimLineColor
- To zoom in, set:
  mpres@ZoomIn = True, and
  mpres@Xstart, mpres@Xend, mpres@Ystart, mpres@Yend, to the corner x/y positions of the zoomed plot.

• pltres@NoTitles	Turn off all titles
• pltres@CommonTitle	Common title
• pltres@PlotTitle	Plot title
• pltres@PanelPlot	Whether a panel plot is to be drawn
• pltres@FramePlot	Whether to advance the frame


REAL-TIME WRF
Init: 2000-01-24_12:00:00
Valid: 2000-01-25_00:00:00
Surface Temperature (F)
Sea Level Pressure (hPa)
Wind (kts)
44°N
42°N
40°N
36°N
34°N
32°N
30°N
1020
1012
1004
996
1004
H 1023
L 989
1012
1020
1004
90°W
85°W
80°W
75°W
Sea Level Pressure Contours: 900 to 1100 by 4
Surface Temperature (F)
-20
-10
0
10
20
30
40
50
60
70
80
90
OUTPUT FROM WRF V2.2 MODEL
Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1 ; WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km

pltres@NoTitles
pltres@CommonTitle

```
t2 = wrf_user_getvar(a,"T2",5)
t2@description = "Surface Temperature”
```

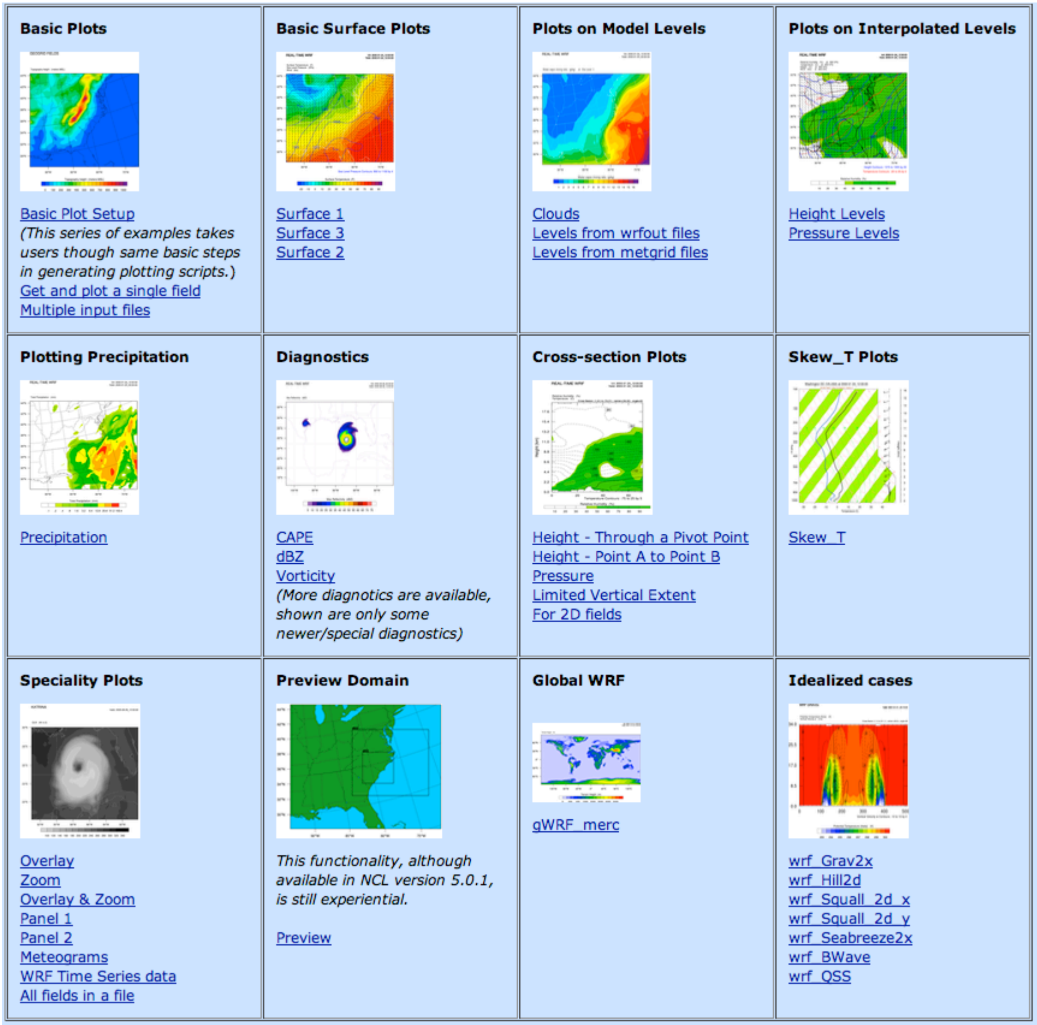
Resources for wrf_overlays

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

Scripts maintained by Cindy Bruyère.

Latest version of WRFUserARW.ncl file usually available here.

Scripts and full-sized images available.



Basic Plots	Basic Surface Plots	Plots on Model Levels	Plots on Interpolated Levels
Basic Plot Setup *(This series of examples takes users though same basic steps in generating plotting scripts.)* Get and plot a single field Multiple input files	Surface 1 Surface 3 Surface 2	Clouds Levels from wrfout files Levels from metgrid files	Height Levels Pressure Levels
Plotting Precipitation	**Diagnostics**	**Cross-section Plots**	**Skew_T Plots**
Precipitation	CAPE dBZ Vorticity *(More diagnostics are available, shown are only some newer/special diagnostics)*	Height - Through a Pivot Point Height - Point A to Point B Pressure Limited Vertical Extent For 2D fields	Skew_T
Speciality Plots	**Preview Domain**	**Global WRF**	**Idealized cases**
Overlay Zoom Overlay & Zoom Panel 1 Panel 2 Meteograms WRF Time Series data All fields in a file	*This functionality, although available in NCL version 5.0.1, is still experiential.* Preview	gWRF_merc	wrf_Grav2x wrf_Hill2d wrf_Squall_2d_x wrf_Squall_2d_y wrf_Seabreeze2x wrf_BWave wrf_QSS

More info on plot resources

- The special WRF-NCL graphical functions have special resources they recognize

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_functions.htm

- Most general NCL resources can also be used to tweak plots (some are set internally and can't be changed)

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

Topics

- Overview
- What's new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Calling Fortran codes from NCL

- Easier to use F77 code, but works with F90 code
- Need to isolate definition of input variables and wrap with special comment statements:

```
C NCLFORTSTART
C NCLEND
```

- Use a tool called WRAPIT to create a *.so file
- Load *.so file in NCL script with “external” statement
- Call Fortran function with special “::” syntax
- Must preallocate arrays! (using NCL’s “new” statement)

http://www.ncl.ucar.edu/Document/Tools/WRAPIT.shtml

Example F77 code: *myTK.f*

```
C NCLFORTSTART
      subroutine compute_tk(tk,pressure,theta,nx,ny,nz)
      implicit none
      integer nx, ny, nz
      real    tk(nx, ny, nz)
      real    pressure(nx, ny, nz), theta(nx, ny, nz)
C NCLEND
      integer i, j, k
      real    pi

      do k=1,nz
        do j=1,ny
          do i=1,nx
            pi = (pressure(i,j,k)/1000.)**(287./1004.)
            tk(i,j,k) = pi*theta(i,j,k)
          end do
        end do
      end do
      end
```

Create “myTK.so” file and use in script

```
% WRAPIT myTK.f
```

This will create a “myTK.so” file

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl”
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”
external myTK “./myTK.so”

begin
      t = wrf_user_getvar(a,”T”,5)
      t = t + 300
      p = wrf_user_getvar(a,”pressure”,5)

; Must preallocate space for output arrays
      dim = dimsizes(t)
      tk  = new( dimsizes(t), typeof(t) )

; Remember, Fortran/NCL arrays are ordered differently
      myTK :: compute_tk (tk,p,t,dim(2),dim(1),dim(0))
end
```

Calling Fortran 90 codes from NCL

- Can use simple Fortran 90 code
- Your F90 program cannot contain any of the following features:
  - pointers or structures as arguments
  - missing or optional arguments
  - keyword arguments
  - recursive procedures
- The input arguments must be reproduced in a separate F77-like “stub” file
- “WRAPIT” is a modifiable script

Example F90 code: *myTK.f90*

myTK.f90

```
subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer :: nx,ny,nz
  real, dimension (nx,ny,nz) :: tk, pres, theta, pi

  pi = (pres/1000.)**(287./1004.)
  tk = pi * theta

end subroutine compute_tk
```

Example F90 code: *myTK.f90 + stub*

myTK.f90

```
subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer :: nx,ny,nz
  real, dimension (nx,ny,nz) :: tk, pres, theta, pi

  pi = (pres/1000.)**(287./1004.)
  tk = pi * theta

end subroutine compute_tk
```

myTK.stub

```
C NCLFORTSTART
  subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer nx,ny,nz
  real    tk(nx,ny,nz)
  real    pres(nx,ny,nz), theta(nx,ny,nz)
C NCLEND
```

Create “myTK.so” file and use in script

```
% WRAPIT myTK.stub myTK.f90
```

Should create a “myTK.so” file. Script will be exactly the same.

```
load “$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load “$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”
external myTK “./myTK.so”

begin
      t = wrf_user_getvar(a,”T”,5)
      t = t + 300
      p = wrf_user_getvar(a,”pressure”,5)

; Must preallocate space for output arrays
      dim = dimsizes(t)
      tk  = new( dimsizes(t), typeof(t) )

      myTK :: compute_tk (tk,p,t,dim(2),dim(1),dim(0))
end
```

Topics

- Overview
- What's new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- **Debugging, common mistakes**
- Installation, setup, URLs

Common mistakes or problems

- Forgot .hluresfile (colors and fonts will look wrong)

- Call wrf_xxxx functions with the wrong units

- "*cnLineColour" is not a resource in ContourPlot at this time*"
  - Misspelling a resource, "cnLineColour"
  - Using the wrong resource with the wrong plot (i.e. using "vcRefMagnitudeF" in a contour plot).

- Data values in plot look off-scale
  - Maybe "_FillValue" attribute not set or not correct.

Debugging tips

- Start with an existing script, if possible
- Use indentation (even though not needed)
- Use “ncl_filedump” to look at file quickly
- Use “printVarSummary” to examine variables
  - Check for no “_FillValue” or wrong “_FillValue” value
- To further examine data, use:
  - print(min(x)) and print(max(x))   ; Minimum/maximum of data
  - print(num(ismissing(x)))   ; Count number of msg vals
- For graphics, make sure spelling the resource name correctly
- Group graphical resources alphabetically
- Read errors and warnings carefully

Things to watch for: *memory & efficiency*

- Nested do loops, unnecessary code in do loops
  - Try to use f90-style arithmetic where possible
  - If code doesn't need to be in do loop (like initializing a variable), move it outside the loop
- Copying metadata unnecessarily. Use (/ and /) to avoid this:

```
ch4_tmp = (/ch4/)
```

- Creating lots of big arrays and not deleting them when no longer needed. Use NCL's "delete" procedure to clean up.
- Reordering the same array multiple times
  - Do once and store to local variable

Topics

- Overview
- What’s new
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Installing NCL and setting up environment

- ESG one-time registration (login/password)
- Download appropriate precompiled binary
- “gunzip” and “tar –xvf” the file
- setenv NCARG_ROOT to parent directory
- Add $NCARG_ROOT/bin to search path
- Copy “.hluresfile” to home directory

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

http://www.ncl.ucar.edu/Download/install.shtml

Problems installing or running NCL?

- Send email to ncl-install@ucar.edu (must subscribe first):

  http://mailman.ucar.edu/mailman/listinfo/ncl-install

- Be specific about problem:
  - What kind of machine (“uname –a”)
  - Which version of NCL, or which file did you download?
  - What exactly is the problem? Include what you are trying to do, and exactly what error message you got.

Customizing your NCL graphics environment

~/.hluresfile

- Download ".hluresfile" file, put in home directory!!
  - Changes your background, foreground colors to white/black
  - Changes font from **times-roman** to **helvetica**
  - Changes "function code" (default is a colon)
  - ***WRF-NCL users:*** use to change the default color map

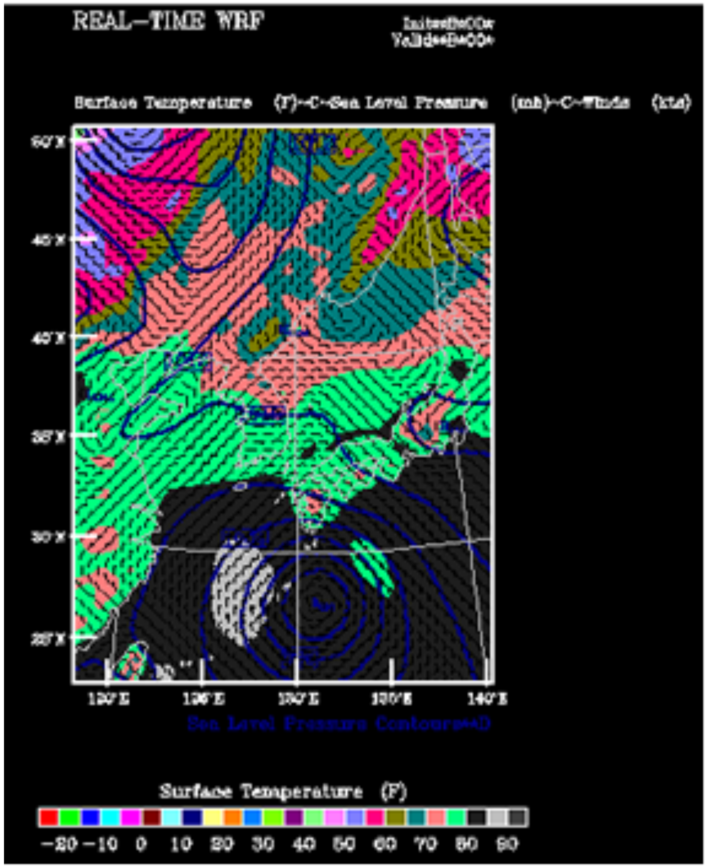
http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml

Sample ".hluresfile"

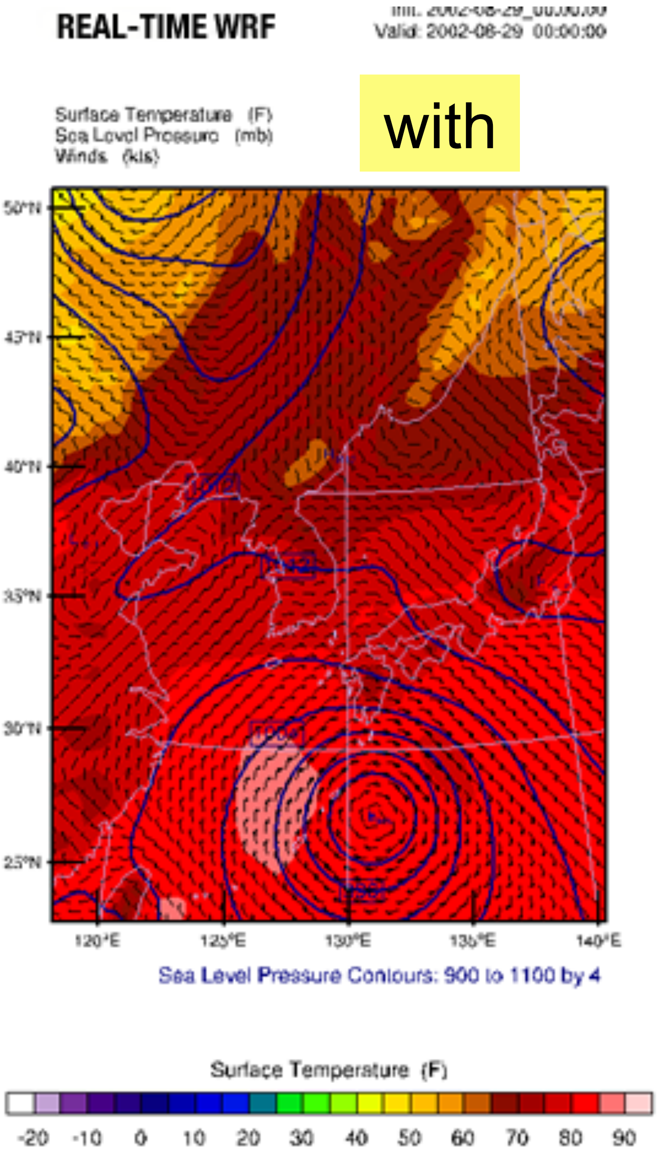
```
*wkForegroundColor   : black
*wkBackgroundColor   : white
*wkColorMap          : BlAqGrYeOrReVi200
*Font                : helvetica
*TextFuncCode        : ~
*wkWidth             : 900
*wkHeight            : 900
```

With and without a “.hluresfile”

without



with



Useful URLS

- Online WRF-NCL Graphics Tutorial
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/
- WRF-NCL functions (built-in and “WRFUserARW.ncl”)
  http://www.ncl.ucar.edu/Document/Functions/wrf.shtml
- Graphical resources
  http://www.ncl.ucar.edu/Document/Graphics/Resources/
- Download NCL
  http://www.ncl.ucar.edu/Download/
- Application examples (includes WRF examples)
  http://www.ncl.ucar.edu/Applications/
- Detailed NCL reference manual
  http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/
- NCL Workshops
  http://www.ncl.ucar.edu/Training/Workshops/
- NCL email lists to join
  http://www.ncl.ucar.edu/Support/email_lists.shtml

Questions?

wrfhelp@ucar.edu

Questions specific to WRF-NCL

ncl-talk@ucar.edu

Issues with NCL (must subscribe first)

http://mailman.ucar.edu/mailman/admin/ncl-talk