Tutorial session on MME and Downscaling

APEC Climate Center
24 July 2010

Purpose and scope

- Introduce NCL – a powerful visualization/data analysis software
- Introduce and practice with higher level NCL libraries developed for MME and downscaling
- Demonstrate use of AFS and DFT libraries for easily and painlessly implementing MME techniques
- Flash quickly through the downscaling library
- Introduce clustering techniques, time permitting

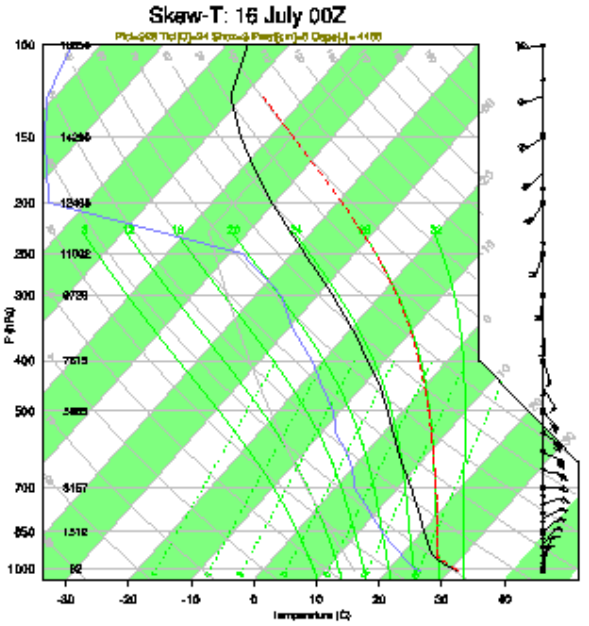
NCL

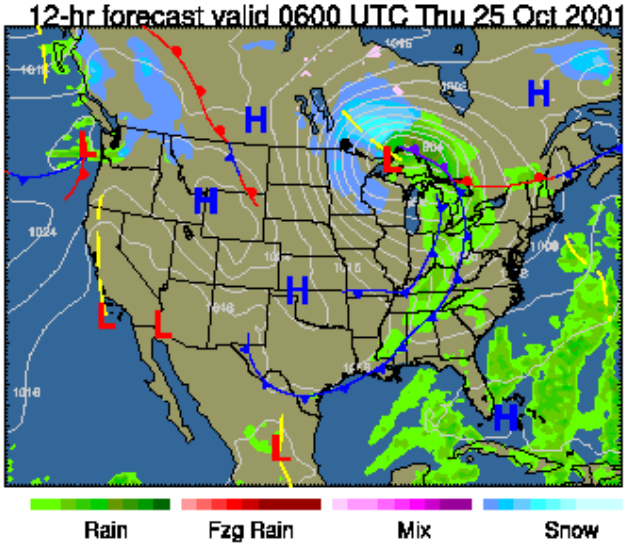
Introduction

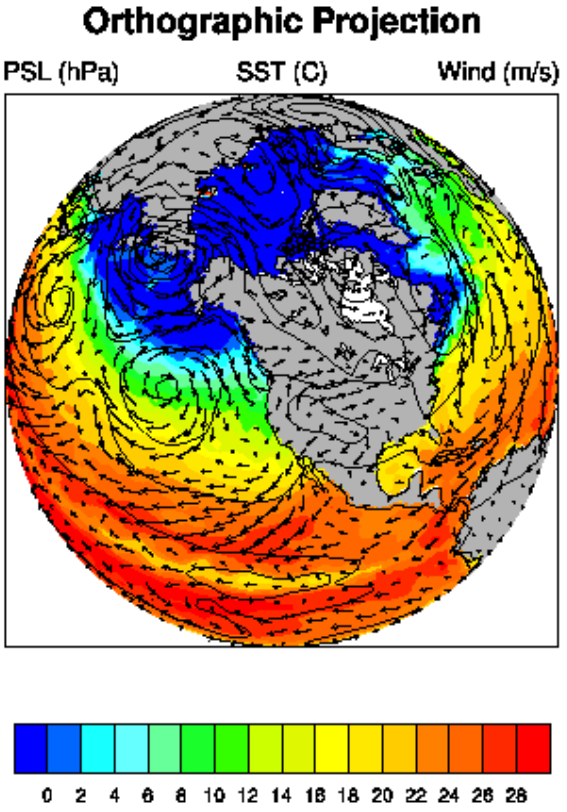
Taken from Dennis Shea's lecture notes

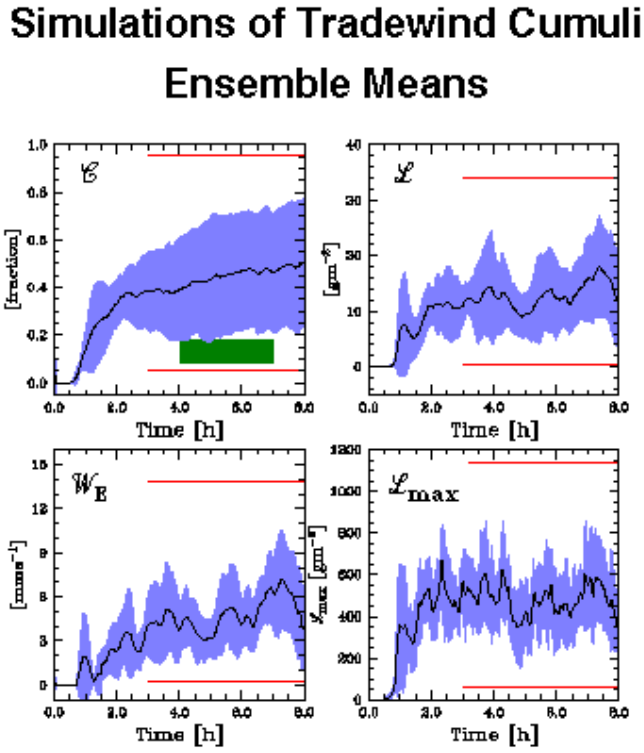
available online @

National Center for Atmospheric Research









Introduction to netCDF

NCL variable model is based on the netCDF variable model

Let's dig in

- Change Dir to /apcc11/AFS_DB
- cd 3-MON/MME_IN/HINDCAST/GCPS/
- cd JUL/1982/
- ncdump -h prec.nc
- ncdump -v time prec.nc
- ncdump -v lat prec.nc
- ncdump -v lon prec.nc

netCDF files

self describing

- (ideally) **all** info contained within the file
- no external information needed to determine file contents

portable [machine independent]

Supported by many software tools / languages

- NCL, IDL, Matlab, ferret, GrADS, F, C, C++,Java
- command line file operators: NCO, CDO
  - perform various tasks very efficiently on files
    - http://nco.sourceforge.net
    - http://www.mpimet.mpg.de/fileadmin/software/cdo
- ncview
  - quick visualization: COARDS
    - http://meteora.ucsd.edu/~pierce/ncview_home_page.html

Many modelling groups use netCDF [IPCC]

Examining a netCDF file

- **ncdump** file_name | less
  - dumps the entire contents of a file; prints every value
- **ncdump -h** file_name | less
  - Dumps header information [most commonly used]
  - NCL equivalent: **ncl_filedump file_name** | less
- **ncdump -v** U file_name | less
  - NCL equivalent: **ncl_filedump –v U file_name** | less
- Note: ncdump is a Unidata utility
  - not a netCDF Operator (NCO / CDO)
  - not associated with NCL

- **ncview:** visualize file contents [COARDS conven]

- **ncl_filedump** file_name [more general]
  - netCDF3/4, GRIB-1,GRIB-2, HDF, HDF-EOS [HDF5]

Parts of netCDF file

ncdump –h foo.nc (or ncl_filedump foo.nc)

DIMENSION SIZES

```
dimensions:
    lat  = 64
    lon  = 128
    time = 12
```

time=UNLIMITED (12 currently)

FILE ATTRIBUTES

```
global attributes:
    title = “Temp: 1999”
    source = “NCAR”
    conventions =
"None”
```

exercise:

ncl_filedump FOO.nc | less

VARIABLES:
Names , Types, Attributes,
Coordinate Variables

```
variables:
  float lat(lat)
    lat:long_name = "latitude"
    lat:units = "degrees_north"
  float lon(lon)
    lon:long_name = "longitude"
    lon:units = "degrees_east"
  int time(time)
    time:long_name = "time"
    time:units = "Month of Year"
  double T(time, lat, lon)
   T:long_name = “Temperature”
   T:units = “C"
   T:missing_value = 1.e+20f
   T:_FillValue = 1.e+20f
```

netCDF/NCL variable

- **array** [could be of length 1 (scalar)]
- **(may) have additional information**

x

```
  4.35     4.39    0.27   -3.35   -6.90
  4.36     4.66    3.77   -1.66    4.06
  9.73    -5.84    0.89    8.46   10.39
    17     3.68    5.08    0.14   -5.63
 -0.63    -4.12   -2.51    1.76   -1.43
 -4.29     0.07    5.85    0.87    8.65
```

name: x
type: float [real]
shape: 2-dimensions
size: 7 (rows) x 5 (columns)
values: x(2,3) = 8.46 [row major]

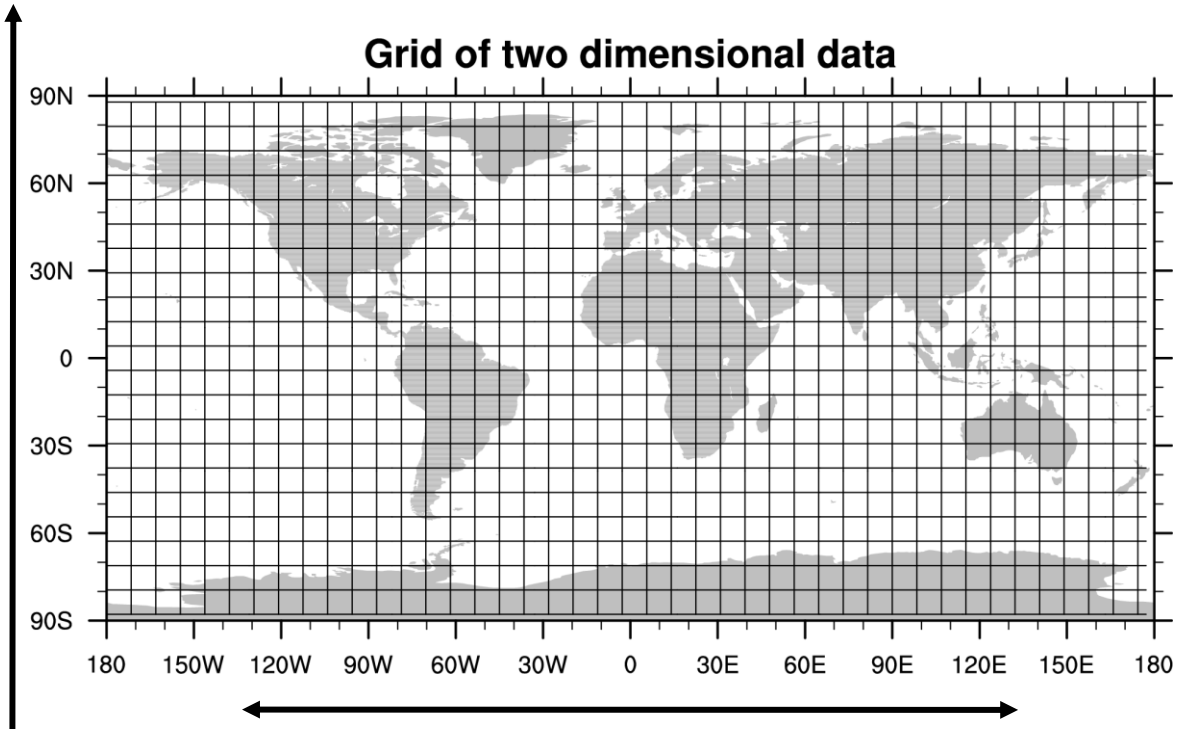
long_name: “Temperature”
units: “C”
named dimensions: x(time,lat)
lat: (/ -60, -30 ,0, 30, 60 /)
time: (/2000,2001,2002,2003,2004,2005,2006 /)

Meta data

visual: simple 2D netCDF Variable

coordinate variables (rectilinear grid)

Latitude coordinate variable (1D, **&**)



Longitude coordinate variable (1D, **&**)

attributes @:

- long_name
- units

NCL is **NOT LIMITED** to netCDF conforming variables

eg: 2D coordinate arrays (curvilinear coords)

netCDF [NCL] Variable model

X

Scalar
or Array

attributes

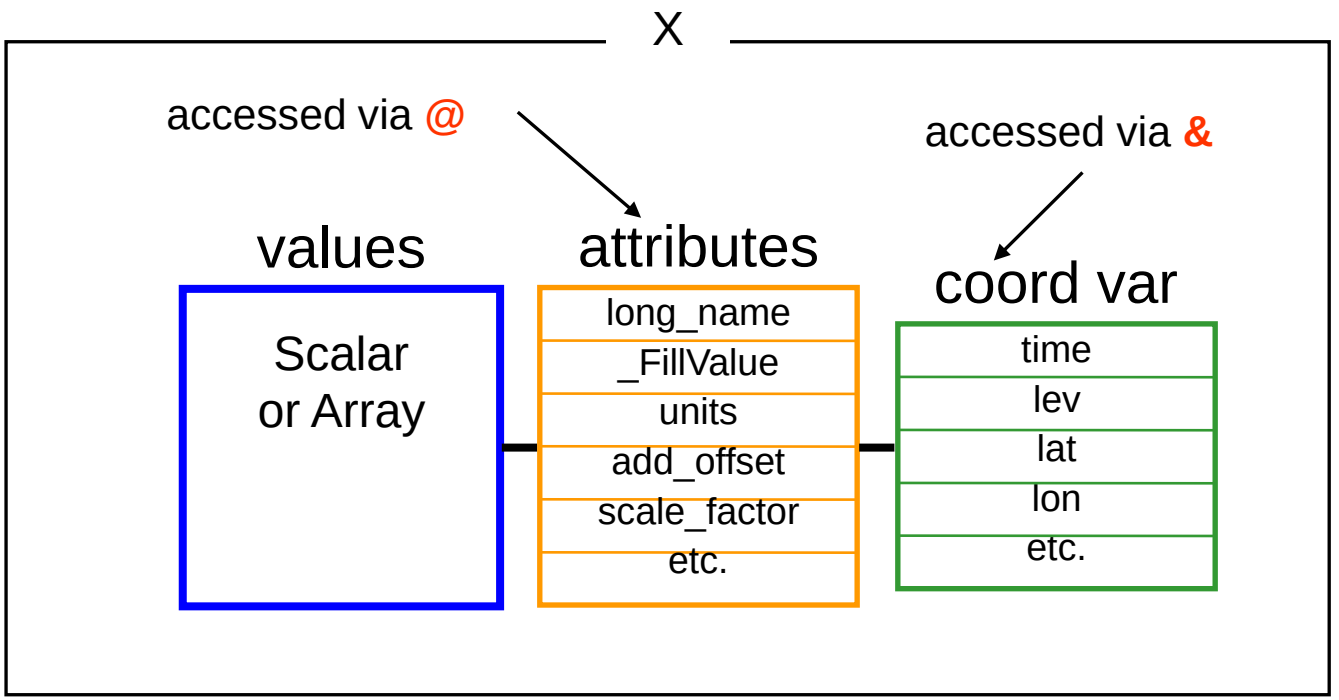
long_name
_FillValue
units
add_offset
scale_factor
etc.

coordinates

time
lev
lat
lon
etc.

```
f = addfile("foo.nc","r")    ; grb/hdf
x = f->X
```

NCL reads the scalar/array, attributes, and coordinate variables as an object



Detailed Look netCDF Variable (NCL)

```
ncl <return>                                   ; interactive mode
ncl 0 > f = addfile ("UV300.nc", "r")          ; open file
ncl 1 > u = f->U                               ; import STRUCTURE
ncl 2 > printVarSummary (u)                    ; overview of variable
```

```
Variable:  u
Type: float
Total Size:  65536 bytes
             16384 values
Number of Dimensions:  3
Dimensions and Sizes:   [time | 2] x  [lat | 64] x [lon | 128]
Coordinates:
             time:  [ 1 .. 7 ]
             lat:  [ -87.8638 ..  87.8638 ]
             lon: [ 0 .. 357.185]
Number of Attributes: 5
    _FillValue :         1e36
    units :              m/s
    long_name  :         Zonal Wind Component
    short_name :         U
    missing_value :    1e36
```

Classic netCDF Variable Model

NCL syntax/funcs
query
use
modify
add
any aspect of data object

Read same file using NCL

- ncl
- fin=addfile(“prec.nc”,”r”)
- print(fin)
- var=fin->prec
- printVarSummary(var)
- new_var=var(:,0,{-30:30},:)
- printVarSummary(new_var)

Doing something more

- Make a seasonal average
- Use function dim_avg_Wrap
- http://ncl.ucar.edu/Document/Functions/Built-in/dim_avg.shtml
- Calculates average w.r.t a variable's rightmost dimension
- Try var3=dim_avg_Wrap(new_var)
- printVarSummary(var3)
- How to make a seasonal average, i.e., average along time?

Variable Reshaping/Reordering

- functions **may** require data in specific order
  - map plot functions want array order T(...,lat,lon)

- **can and should be done without loops**
  - use NCL syntax or functions
  - very fast for variables in memory

- how? … two approaches: let T(time,lat,lon)
  - named dimensions: t = T(lat**|:**, lon**|:**, time**|:**)
  - NCL functions:
    - **ndtooned**: t1D = **ndtooned**(T)
    - **onedtond**: t2D = **onedtond**(t1D, (/N,M/))

Visualizing data

- var_seas=dim_avg_Wrap(new_var(lat|:,lon|:,time|:)
- cp /apcc11/Tutorial/NCL/vis1.ncl .
- vi vis1.ncl (use nano or less if needed)

vis1.ncl

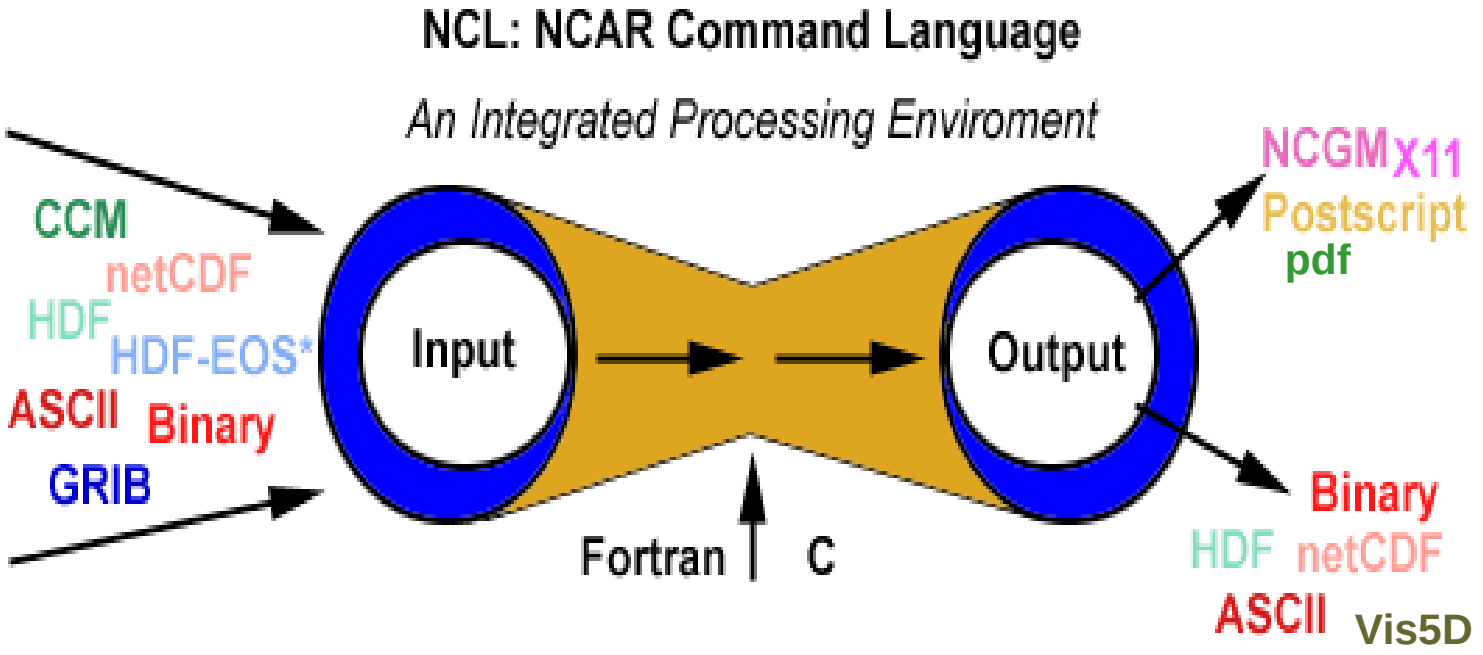
```
load
  "$AFS2/lib/ncl/helper_libs.n
  cl"

data_dir =
  "/home/saji/AFS_DB/3-MON
  /MME_IN/HINDCAST/GCPS/JUL/19
  82"

fin=addfile(data_dir+"/prec.n
  c","r")
```

Shaping Plots

- Use of attributes to passing on various options to the plotting programs
  - cnFillOn = True ; yields shaded plot
  - cnLinesOn = False ; suppresses line contours
- Different plotting functions, different results
  - gsn_csm_contour ; contour only plot
    gsn_csm_contour_map ; contours atop a map

NCL Overview

- **Integrated processing environment**



- **freeware: supported, public domain**
- **portable: linux/unix, windows (cygwin), MacOS**
- **general purpose: unique capabilities**
- **excellent graphics (2D, limited 3D)**

More Info on NCL:

- Accompanying Notes
- http://ncl.ucar.edu

AFS Libraries – a set of higher level libraries aimed to make life with MMEs more easier

AFS Libs

- A collection of functions and procedures
- Some as simple/ridiculous as providing an alias to existing function.
  - e.g. info is an alias to PrintVarSummary – it makes for less typing
- Some a bit more clever
  - e.g ensure_last_dim(var,dim) – works wonders for you in reordering arrays with minimal effort

Our first AFS function is:

single_year_predictions() = a function that helps to retrieve prediction data Easily from the AFS Data Base

```
load "$AFS2/lib/ncl/helper_libs.ncl"
load "$AFS2/lib/ncl/AFS.Definitions"
load "$AFS2/lib/ncl/readers/mme_in_helper.ncl"
begin
  models=(/"SINT","NCEP","POAMA"/)
  vnam="prec"
  opt=True
  db_root="/apcc01/OPER/AFS/DATA"
  type="Forecast"
  lead=6
  smon=12
  syr=2008
  var1=single_year_predictions(db_root,type,models,vnam, \
                               smon,lead_time,syr,opt)
  info(var1)
```

TRY IT OUT NOW!!!

Using attributes to change the output!

```
opt@months = (/"Feb","Mar"/)
var2=single_year_predictions(db_root,type,models,vnam, \
                             smon,lead_time,syr,opt)
info(var2)
```

```
delete(opt@months) ; we cannot prescribe opt@months and opt@season together
opt@season="JFM"
var3=single_year_predictions(db_root,type,models,vnam, \
                           smon,lead_time,syr,opt)
info(var3)
```

What if I want to retrieve anomaly value instead of raw data

Go Ahead and Use

```
single_year_predictions_as_anom
```

What if I want to retrieve anomaly value of multiple years

You need to Use

`multi_year_hindcasts_as_anom.`

These not good enough?

With a little more effort, mme_in_reader gives you the ultimate in flexibility!

```
load "$AFS2/lib/ncl/helper_libs.ncl"
load "$AFS2/lib/ncl/AFS.Definitions"
load "$AFS2/lib/ncl/readers/mme_in_helper.ncl"

model="COLA"
var_name="prec"

opt=True
db_root="/apcc01/OPER/AFS/DATA"
data_type="Hindcast"
data_period=(/6,8,1990,1994/)
opt@ensemble_average=False
opt@season="JJA"

var1=mme_in_reader(db_root,data_type,model,var_name,data_period,opt)
print(var1&year)
info(var1)
```

A silly plot helper

- Take any 2D variable, say var

load “$AFS2/lib/ncl/helper_libs.ncl”

DebugPlot(var)

var@shaded=True

DebugPlot(var)

var@map=True

DebugPlot(var)

var@title=”What a nice plotter”

DebugPlot(var)

Exercise

- Use AFS function single_year_predictions_as_anom to retrieve 3-month lead retrospective predictions from GCPS, NCEP and CWB starting July 1982
- Calculate MME for JJA season
- Plot the result in color on a map with appropriate title, using DebugPlot.

A simple skill score

- Let's make a 3-category prediction, below-normal/normal/above-normal
- Assign score 1 if we forecast a event correctly
  - Score == 1 if we forecast normal and it was normal
  - Score == 0 if we forecast below normal, but it was normal
- Success rate = no of success/total number of forecasts

Implement success rate in NCL

- cp /apcc11/Tutorial/MME_Downscaling/AFS/SSE/skill_score.ncl .
- vi skill_score.ncl (use nano or less if needed)

skill_score.ncl

- New functions
  - **hindcast_period=find_hindcast_years(db_root,models,var_name,start_mon, lead_time,opt)**
    - **Finds common training period for a given set of models**
  - **obs0=get_obs_anom(obs_db_root,var_nam, start_mon,nmon_obs,hindcast_years,opt)**
    - **Gets observed anomalies from database**
  - **success_rate(model,obs)**
    - **Computes success rate**

success_rate(model,obs,opt)

- vi sse_helpers.ncl ; contains definition of success rate

```
function success_rate(model,obs,opt)
begin
    1. categorize data
    2. compare categorized obs and model
    3. some helper functions are used
       To maintain the dimension order in
  correct way
end
```

categorize(var)

- vi sse_helpers.ncl ; contains definition of categorize

```
function categorize(var)
begin
    1. normalize the data using std. dev
    2. use where function to assign 1,0
  and -1 to above normal, normal and
  below normal events
end
Limitation – function is limited in
  scope, since it makes three equal
  categories only
```

Synthetic Super Ensemble method of MME
(browse Yun et al during lunch time)

SSE

- Involves two steps
  - Correct spatial biases of a forecast using EOFs
  - Unequally weighted models enter the MME

Final SSE algorithm in 4 steps

- Learn how to construct EOFs - make_eofs.ncl
- Using EOFs to correct model forecast - sse1.ncl
- MME after spatially corrected forecast - sse1_mme.ncl
- Weighted MME after above step - sse2_mme.ncl

make_eofs.ncl

- To reduce clutter in the scripts, we use helper functions from get_data_helpers.ncl – this is used in all subsequent scripts to get various data from the databases.
  - `obs_anom = my_get_obs_anom(db_root, obs_db_root,models, var_name, start_mon, lead_time, opt)`
  - `mod_anom = my_get_mod_anom(db_root, models, var_name, start_mon, lead_time, opt)`

make_eofs.ncl

- Reads observed and model data and construct several EOFs
- Uses NCL function eofunc to make EOFs and eofunc_ts to make PC time series
  - Both require that data be arranged so that time is the last dimension
  - See `http://ncl.ucar.edu/Document/Functions/Built-in/eofunc.shtml`

sse1.ncl

- Corrects model forecasts using following strategy:
  - Compute observed EOFs – step 1
  - Project observed EOFs to model forecasts and derive PC time series – step 2
  - Reconstruct model forecasts from results of step1 and step2
    - Uses NCL function eof2data(eof,pc) to do the reconstruction
    - Eof2data is described at `http://ncl.ucar.edu/Document/Functions/Built-in/eof2data.shtml`

sse1.ncl

- Introduces some new plotting capabilities
- DebugPlot is too inflexible for this purpose, so we dig into NCL's own routines and AFS routines
  - wks=open_wks(a,b,c) – AFS function to open a graphic workstation, set output file type and name and set the color scheme
  - myfill(a,b) – AFS function; helps make shaded plots
  - mymap(a,b) – AFS function; helps set map attributes
  - gsn_contour_map(a,b,c) – NCL function; to plot a variable and project it on a map *(compare it with gsn_contour)*
  - gsn_panel(a,b,c,d) – NCL function ; makes it easy to

sse1_mme.ncl

- Almost same as sse1.ncl, but makes an MME after spatially correcting model fields using observed EOFs

sse2_mme.ncl

- Different from sse1_mme.ncl in that the MME is not a simple MME, but a weighted MME
- The weights are found by multiple linear regression of observed PCs and model PCs
- Uses NCL built-in function reg_multlin(y,X,opt)
  - See `http://ncl.ucar.edu/Document/Functions/Built-in/reg_multlin.shtml`