Introduction to Transformations, Derived variables, Building 'JNL' and go files for automation

FERRET Course Day - 3

17th March 2021

R U V N Satish

Outline of the Talk

✓Derived Variables

- ❑ How to define a Derived Variable ?
- ❑ Sample scripts defining few key Derived Variables
- ❑ Hands on Session on Derived Variables

✓Transformations

- ❑ What is a Transformation ?
- ❑ List of currently available Transformations
- ❑ Examples of Important Transformations
- ❑ Hands on Session on Important Transformations

✓Building JNL and go files

- ❑ What is a JNL file ? Applications ?
- ❑ How to pass arguments to JNL file ?
- ❑ Automation of go files
- ❑ Hands on Session on Building JNL and go files

DATA SETS For Hands on Session

Dataset	Description
etopo120	relief of the earth's surface at 120-minute resolution
etopo60	relief of the earth's sur face at 60-minute resolution
levitus_climatology	subset of the Climatological Atlas of the World Oceans by Sydney Levitus (Note: the updated World Ocean Atlas, 1994, is also available with Ferret)
coads_climatology	12-month climatology derived from 1946–1989 of the Comprehensive Ocean/Atmosphere Data Set
monthly_navy_winds	Monthly averaged Naval Fleet Numerical Oceanography Center global marine winds (1982–1990)
esku_heat_budget	Esbensen-Kushnir 4×5 degree monthly climatology of the global ocean heat budget (25 variables)

Derived variables

- Using the LET command new variables may be created "from thin air" as **abstract expressions** or created from combinations of known variables as arbitrary **expressions**.

Yes? Let diff = airt – sst

- One should note that the new variable being created from existing variables should be of same grid size.

- If not then re-gridding need to done for setting them on the same platform for obtaining the variables.

- If no arguments are supplied to the variables then the new variables creates hold the same grid definition of the existing variables in the file.

Example:

Yes? Let diff = temp[z=0] – temp[z=5] !only creates one level for z

Yes? Let diff = temp[z=0] – temp !creates diff for all z levels in the file.

Operators & Expressions in Ferret

Valid operators are

+, −, *, /, ^ (exponentiate), AND, OR, GT, GE, LT, LE, EQ, NE

For instance the exponentiate operator can compute the square root of a variable as var^0.5

The expressions may also contain a syntax of:
IF condition THEN expression_1 ELSE expression_2

Examples of Expressions

i) ***temp ^ 2***
temperature squared

ii) ***temp - temp[Z=@AVE]***
for the range of Z in the current context, the temperature deviations from the vertical average

iii) ***COS(Y)***
the cosine of the Y coordinate of the underlying grid (by default, the y-axis is implied by the other variables in the expression)

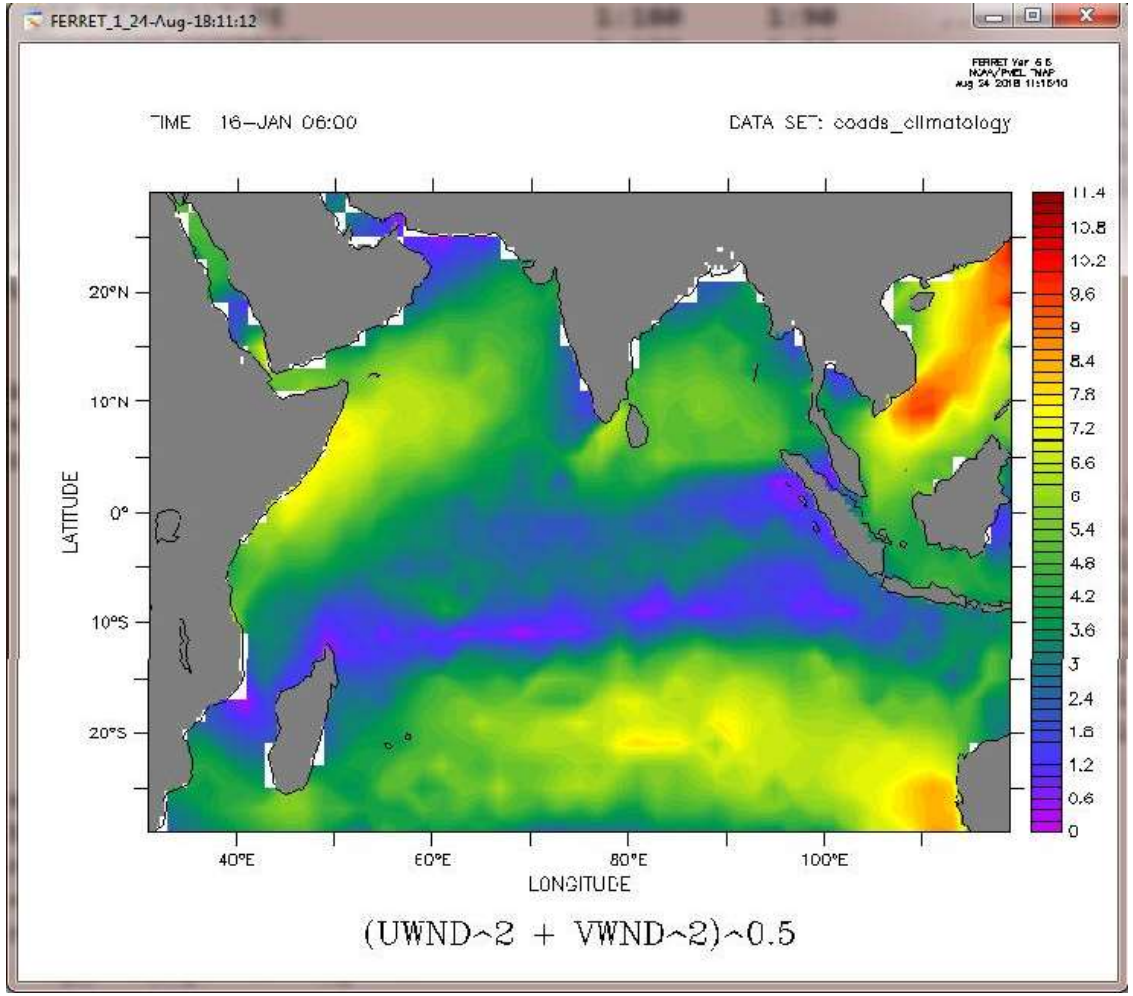
iv) ***IF (vwnd GT vwnd[D=monthly_navy_winds]) THEN vwnd ELSE 0***
use the meridional velocity from the current data set wherever it exceeds the value in data set monthly_navy_winds, zero elsewhere.

Examples of derived variables

- Let us derive a magnitude from individual u and v components.
  - Magnitude is obtained by sqrt($u^2 + v^2$)
  - Let us define the new variable as “mag” and assign the expression to this variable.
- Eg:-

```
yes? Use coads_climatology
yes? Let mag = (uwnd^2 + vwnd^2)^0.5
yes? Set window 1
! Let us compare this with original wspd variable in built
yes? Fill mag[l=1];go fland; go land
yes? Set window 2
yes? Fill wspd[l=1];go fland; go land
```

- We observe slight differences, which can be attributed to interpolation.



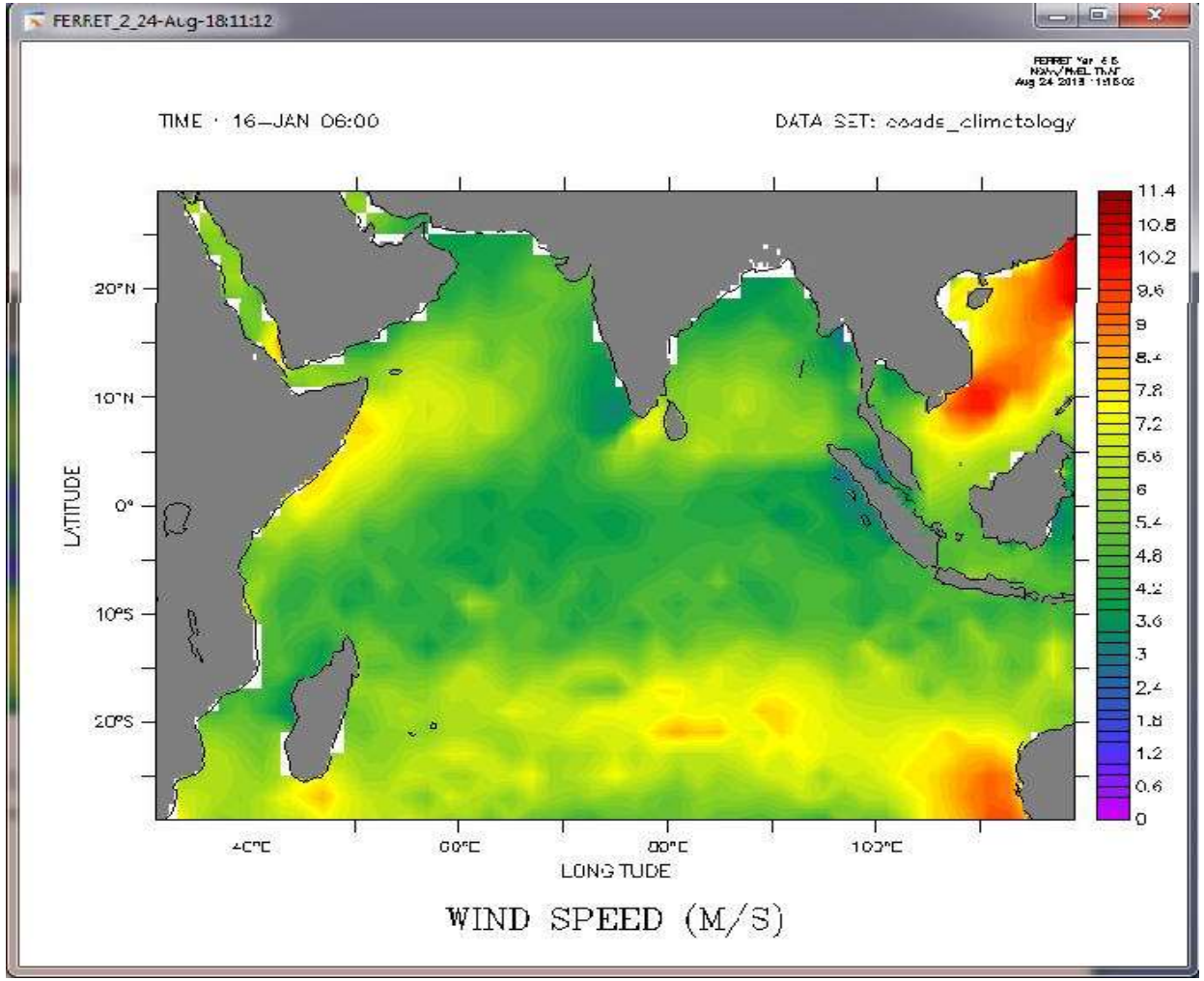
Yes? Set window 1

! Let us compare this with original wspd variable in built

yes? Fill mag[l=1];go fland; go land

yes? Set window 2

yes? Fill wspd[l=1];go fland; go land



Another example .....

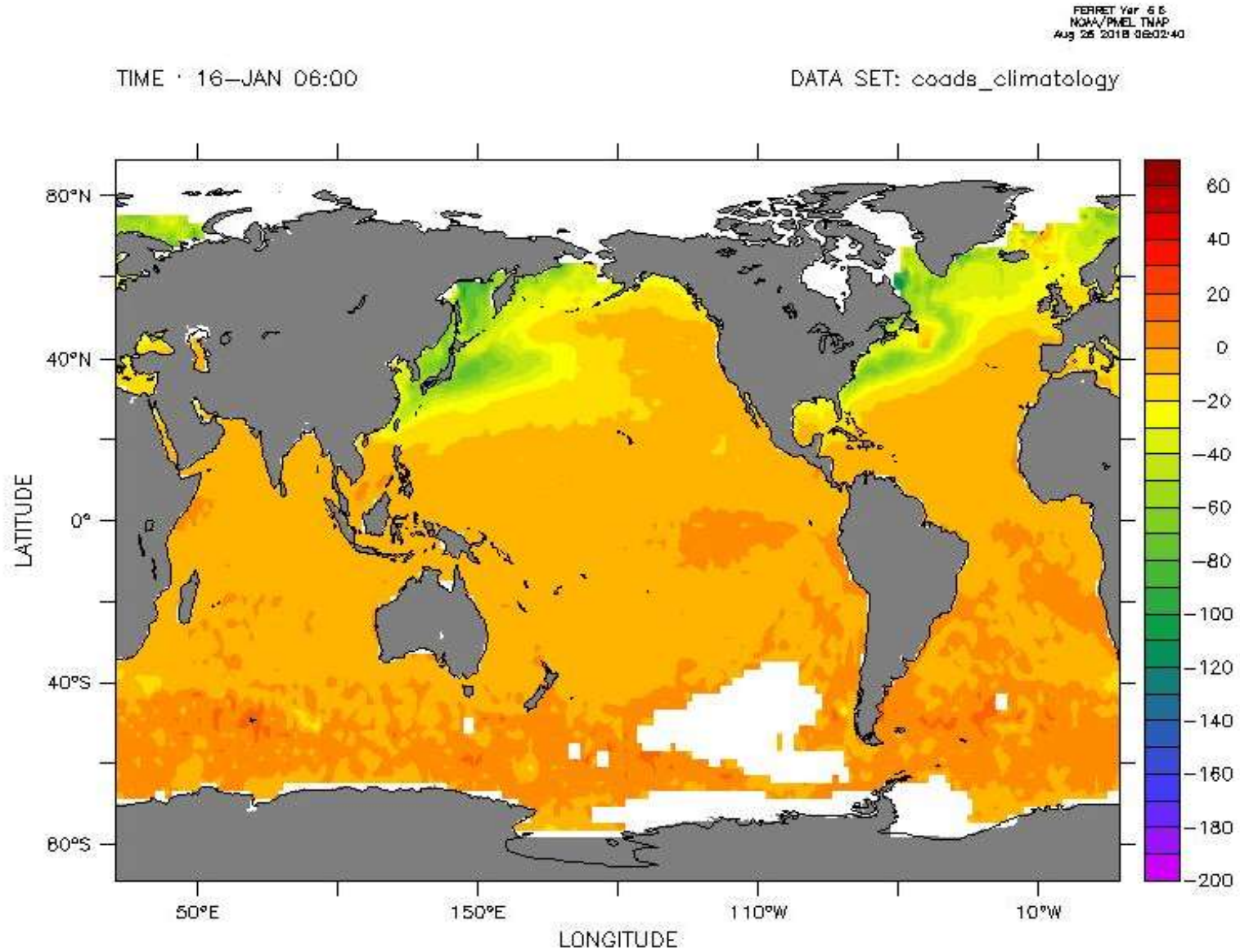
- Let us derive sensible heat using wind speed, air temperature and sea surface temperature.
  - We use the following formula for deriving the sensible heat
    - Sensible Heat = k * *(air temp –sea surface temp)* * wind speed
  - Let us define the new variable as “sens_heat” and assign the expression to this variable.
- Eg:- **yes? Use coads_climatology**

  yes? Let kappa = 1.004

  yes? Let sens_heat = kappa * (airt – sst) * wspd

  ! Let us plot and see how the sensible heat looks like

  yes? Fill sens_heat[l=1]; go fland; go land
- We observe the figure for the entire global ocean as we have not set any bounds.



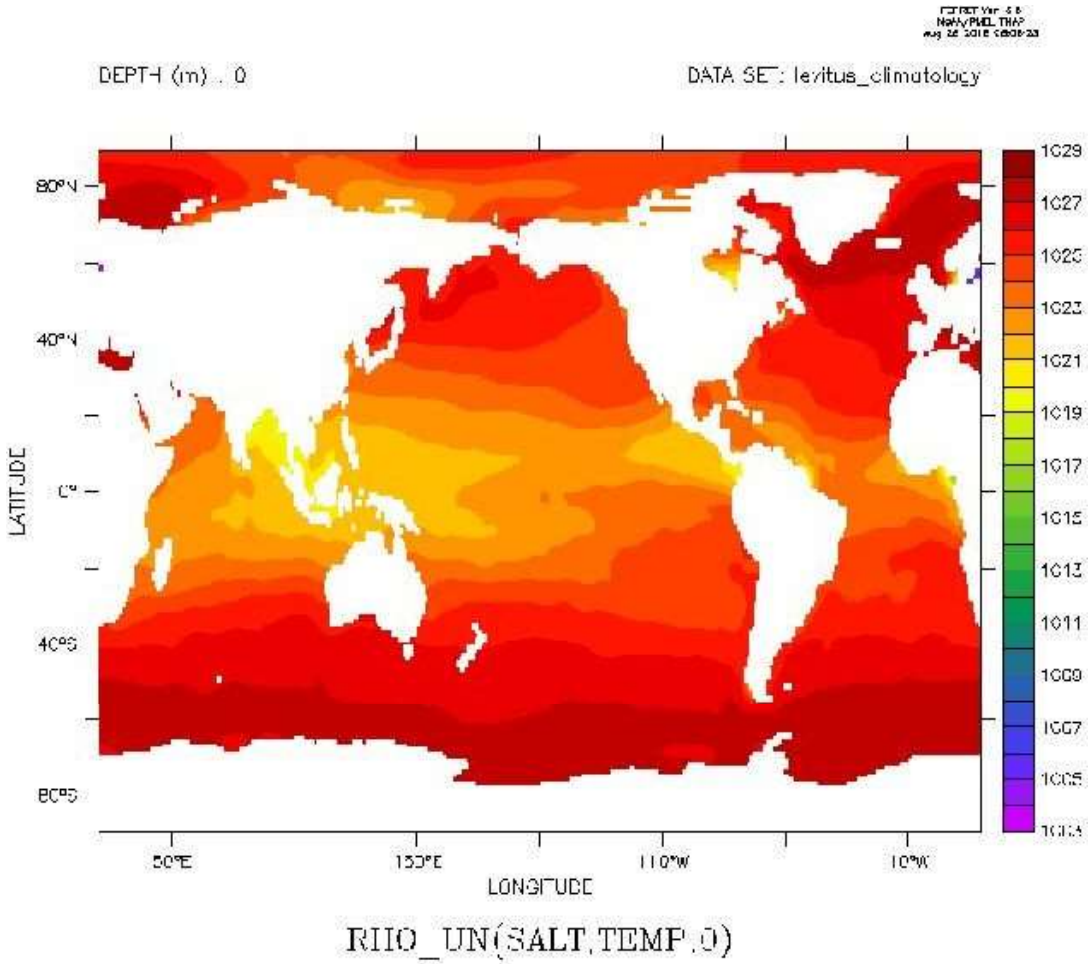
KAPPA * (AIRT − SST) * WSPD

Using built in functions

- There are many built in functions provided with in ferret which one can use for deriving a new variables.
- The list of them can be obtained using
  - **Yes? Show functions**
    - EXP(X) - exponential e(X)
    - LOG(X) - base 10 log(X)
    - MAX(A,B) - point-by-point greater of A and B
    - MIN(A,B) - point-by-point lesser of A and B
    - INT(X) - truncate to integer
    - ABS(X) - absolute value
    - SIN(theta)
- All these can be used for creating new variables

Examples using built in functions

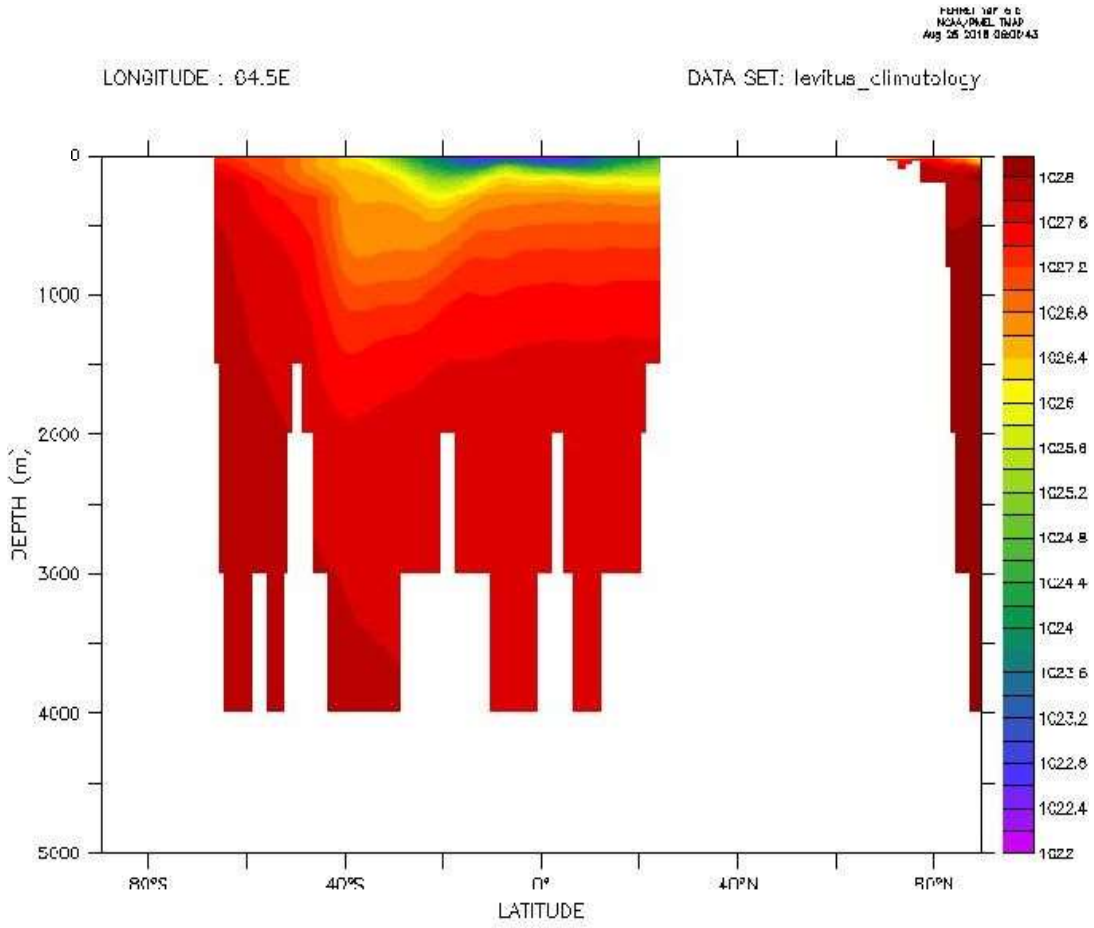
- To derive the density from temperature and salinity we use the built in function called rho_un(s,t,d) as shown below
  - **Yes? Let density = rho_un(salt,temp,0)**
- This derived variable den will have the same dimensions as that of temperature and salinity
  - **Yes? show grid density**
    - GRID GMS1
    - name axis # pts start end
    - XAXLEVITR LONGITUDE 360mr 20.5E 19.5E(379.5)
    - YAXLEVITR LATITUDE 180 r 89.5S 89.5N
    - ZAXLEVITR DEPTH (m) 20 i- 0 5000
    - normal T
- The plotting of density can be done the same way as that of temperature and salinity.



RHO_UN(SALT,TEMP,0)

Yes? fill density[z=0]

Yes? fill density[x=4.5e]



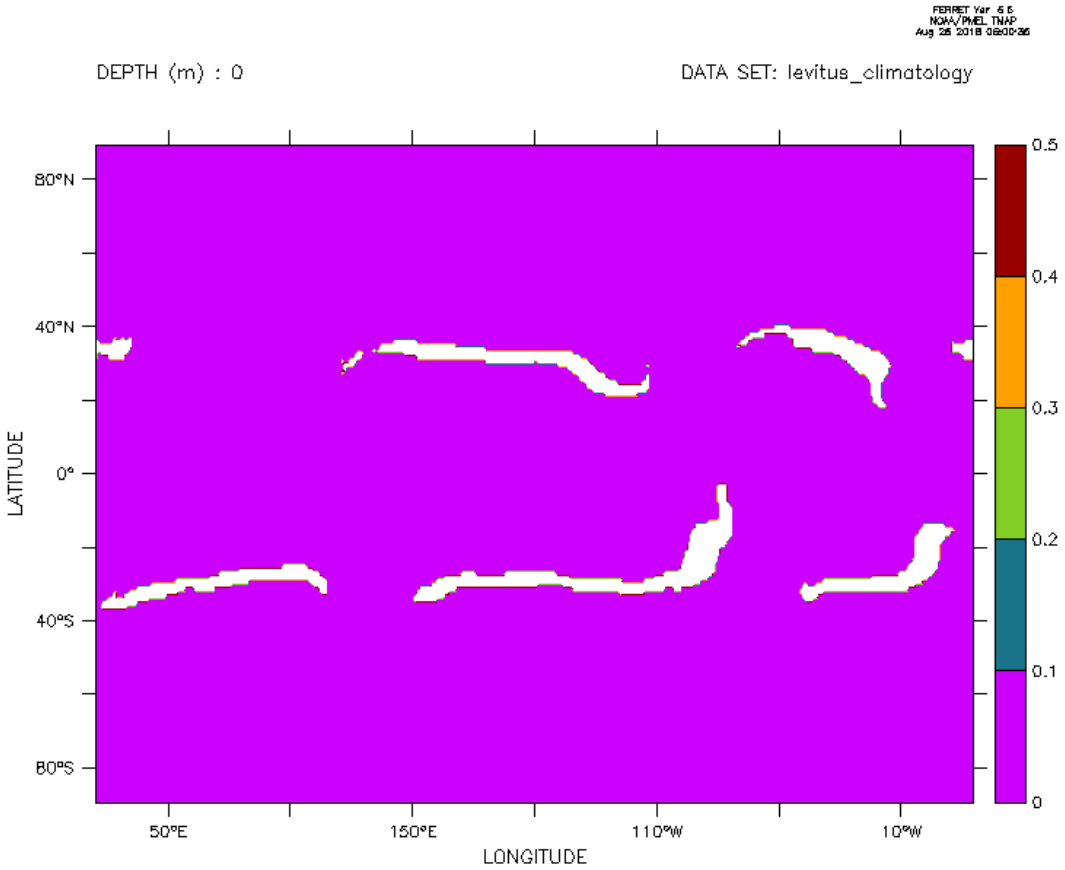
RHO_UN(SALT,TEMP,0)

Using let with “if then else”

- Some conditional assignment can be done and new variable can be creating, by using both let command and “if then else”.

- Generally this can be used for creating mask or eliminating some of the unwanted levels of data in the desired variables.
  - Ex: If we want to capture only region of temperature between 20 and 22 deg C we give.
    - **Yes? let temp_required = if temp gt 20 and temp le 22 then temp**

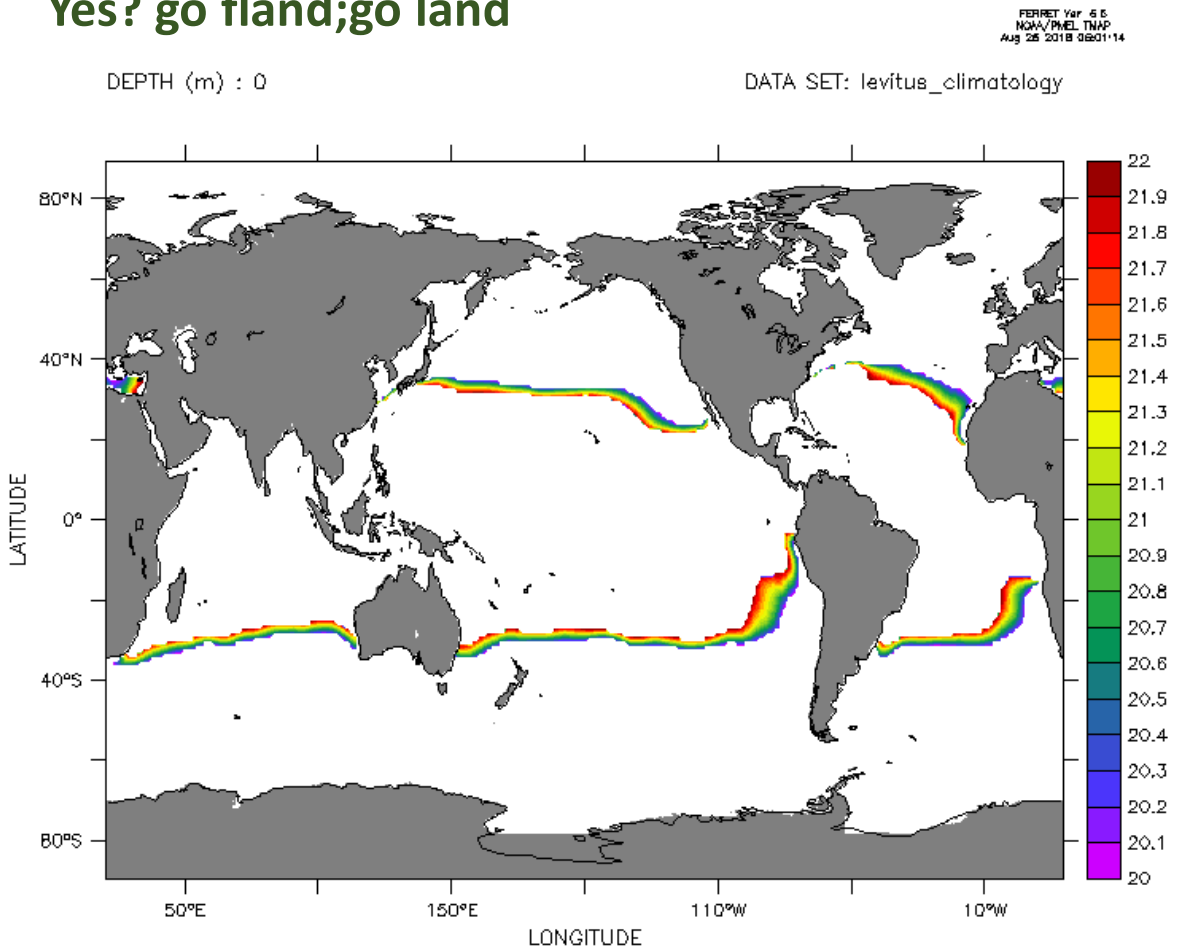
- We can also create mask to be used for eliminating unwanted stuff.
  - **Yes? let mask = if temp gt 20 and temp le 22 then 1 else 0**
  - **Yes? fill/level=(20,22,0.1) temp[l=1,k=1]*mask[l=1,k=1];go fland;go land**



IF TEMP GT 20 AND TEMP LE 22 THEN 1 ELSE 0

Yes? fill mask[l=1,k=1];

Yes? fill/level=(20,22,0.1) temp[l=1,k=1]*mask[l=1,k=1];
Yes? go fland;go land



TEMP[L=1,K=1]*MASK[L=1,K=1]

Let's Start Hands on Derived Variables

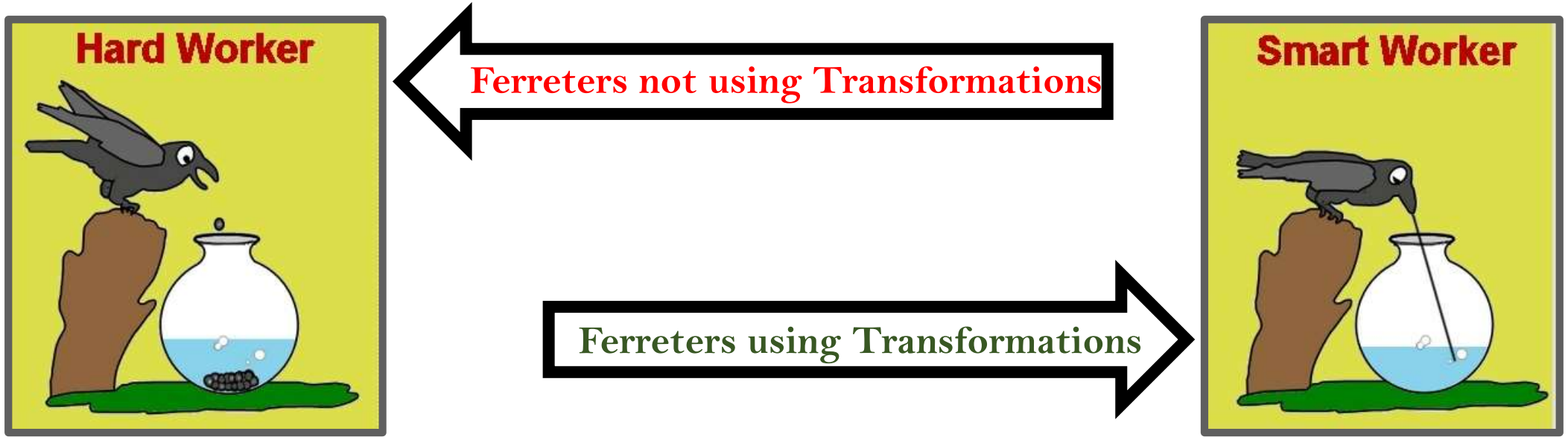
FERRET

TRANSFORMATIONS

What is a Transformation ?

Transformation is an operation performed on a variable along a particular axis and is specified with the syntax "@trn" where "trn" is the name of a transformation.

Using Transformations Complex operations are made easy.



The command **SHOW TRANSFORM** will produce a list of currently available Transformations.

Transform – Description
@SUM - unweighted sum
@RSUM - running unweighted sum
@DIN - definite integral (weighted sum)
@IIN - indefinite integral (weighted running sum)
@AVE - average
@VAR - unweighted variance

@MIN - minimum
@MAX – maximum

@SHF - shift (Default value: 1pt)
@SBX - box car smoothed (Default value: 3pt)
@SBN - binomial smoothed (Default value: 3pt)
@SHN - Hanning smoothed (Default value: 3pt)
@SPZ - Parzen smoothed (Default value: 3pt)
@SWL - Welch smoothed (Default value: 3pt)

@NGD - number of valid points
@NBD - number of bad (in valid) points flagged

@FAV - fill missing values with average (Default value: 3pt)
@FLN - fill missing values by linear interpolation
(Default value: 1pt)
@FNR - fill missing values with nearest point (Default value: 1pt)

@LOC - coordinate of ... (e.g., depth of 20 degrees)
(Default value: 0)
@WEQ - "weighted equal" (integrating kernel)
@CDA - closest distance above
@CDB - closest distance below
@CIA - closest index above
@CIB - closest index below

@DDC - centered derivative
@DDF - forward derivative
@DDB - backward derivative

Examples of Transformations

- If we want to find the depth of 20 degrees isotherms from the temperature data sets.

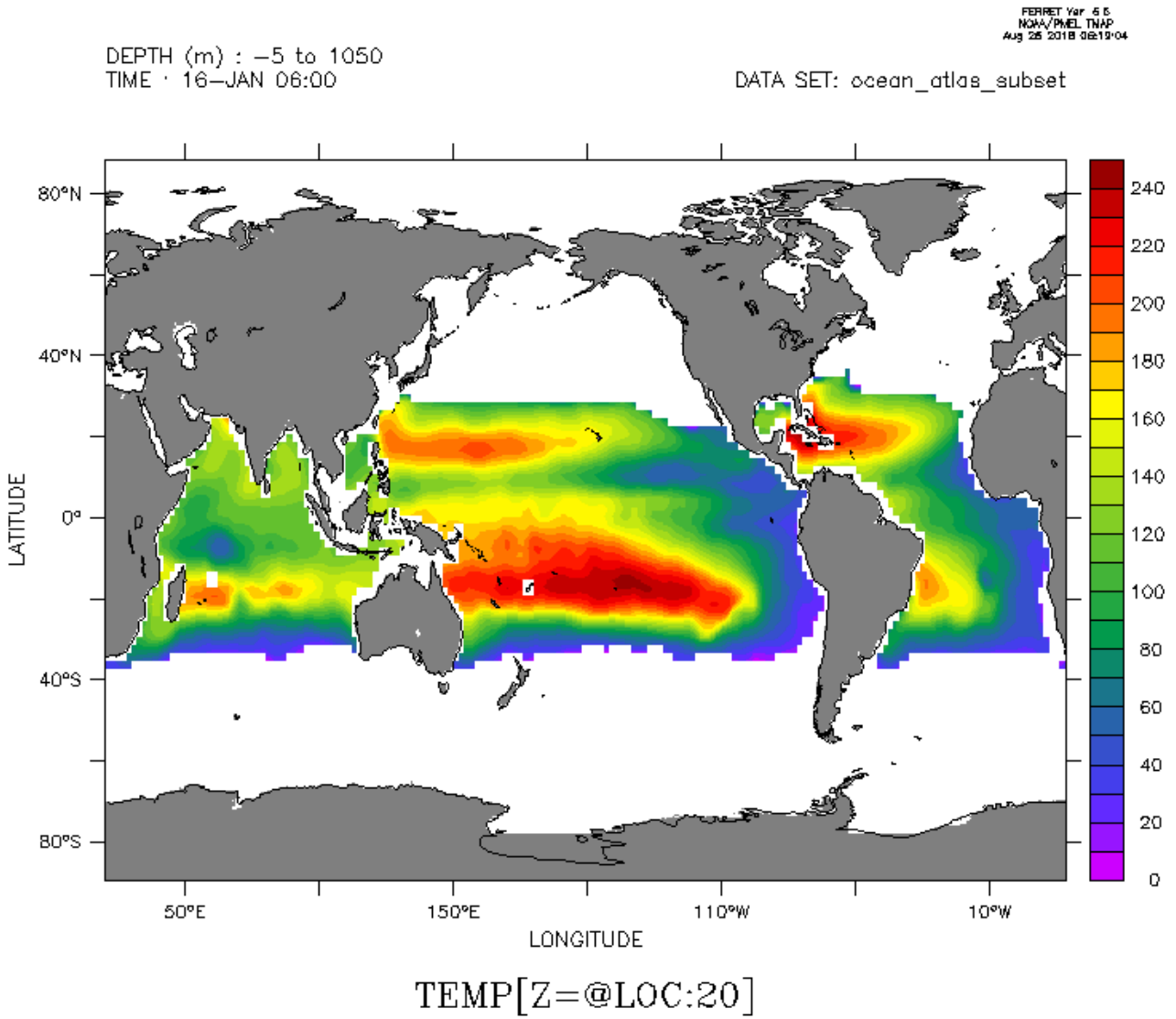
yes? use ocean_atlas_subset

yes? let d20 = temp[z=@loc:20]

yes? fill d20[l=1]

yes? go fland;go land

- This will yield the depth and display the depth of 20 deg isotherm across the global ocean.

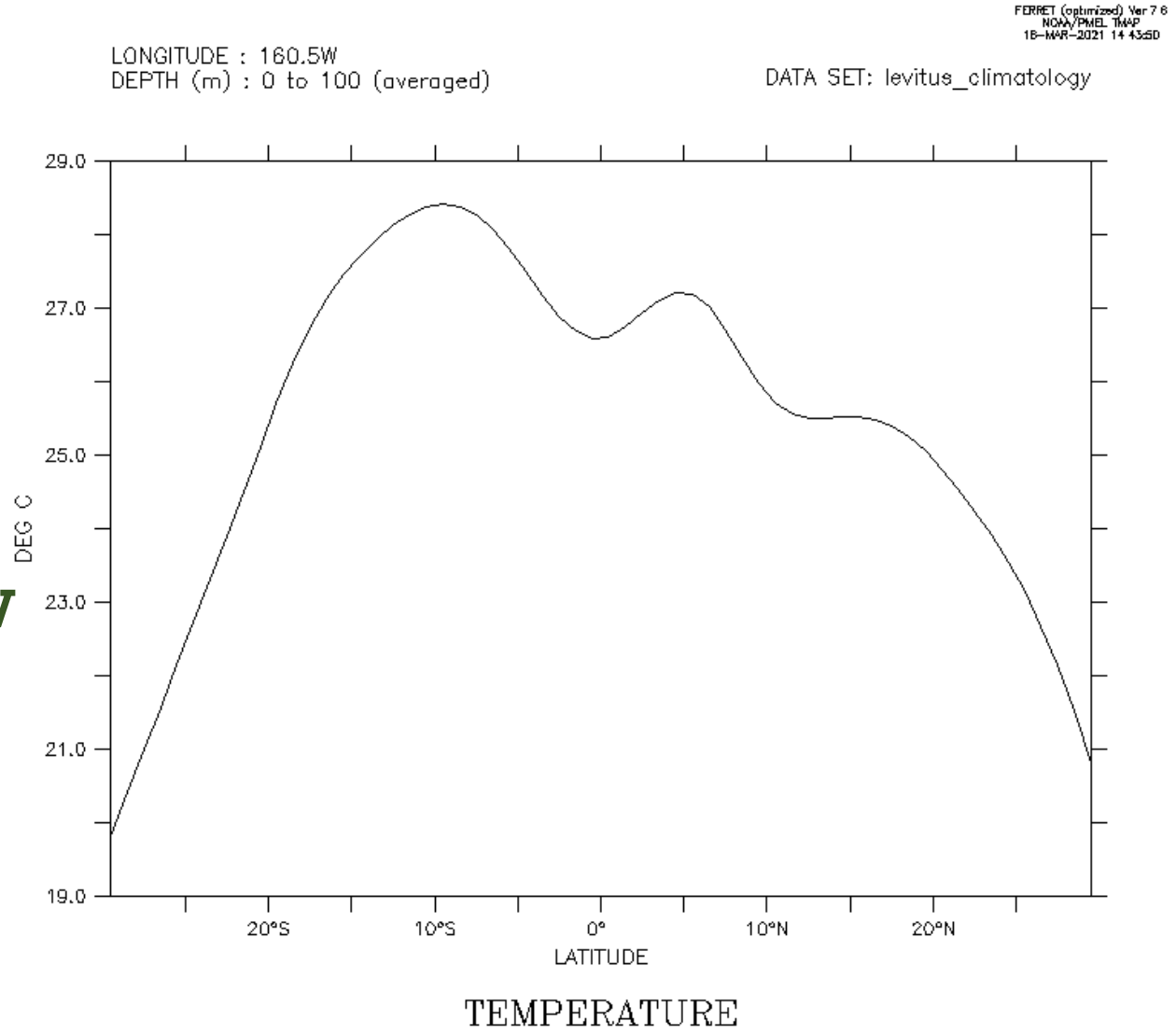


@AVE - average

The transformation @AVE computes the average weighted by grid box size—a single number representing the average of the variable between two end points.

Suppose you want to average Ocean temperatures over a range of depths, say from 0 to 100 meters of depth using levitus climatology data set which has detailed resolution in depth. Lets plot the data along longitude 160 west from latitude 30 south to 30 north.

```
% ferret
yes? USE levitus_climatology
yes? SET REGION/Y=30s:30n/X=160W
yes? PLOT temp[Z=0:100@AVE]
```



Examples continued …

- If we want to find heat content up to depth of 300 mts.

yes? use ocean_atlas_subset

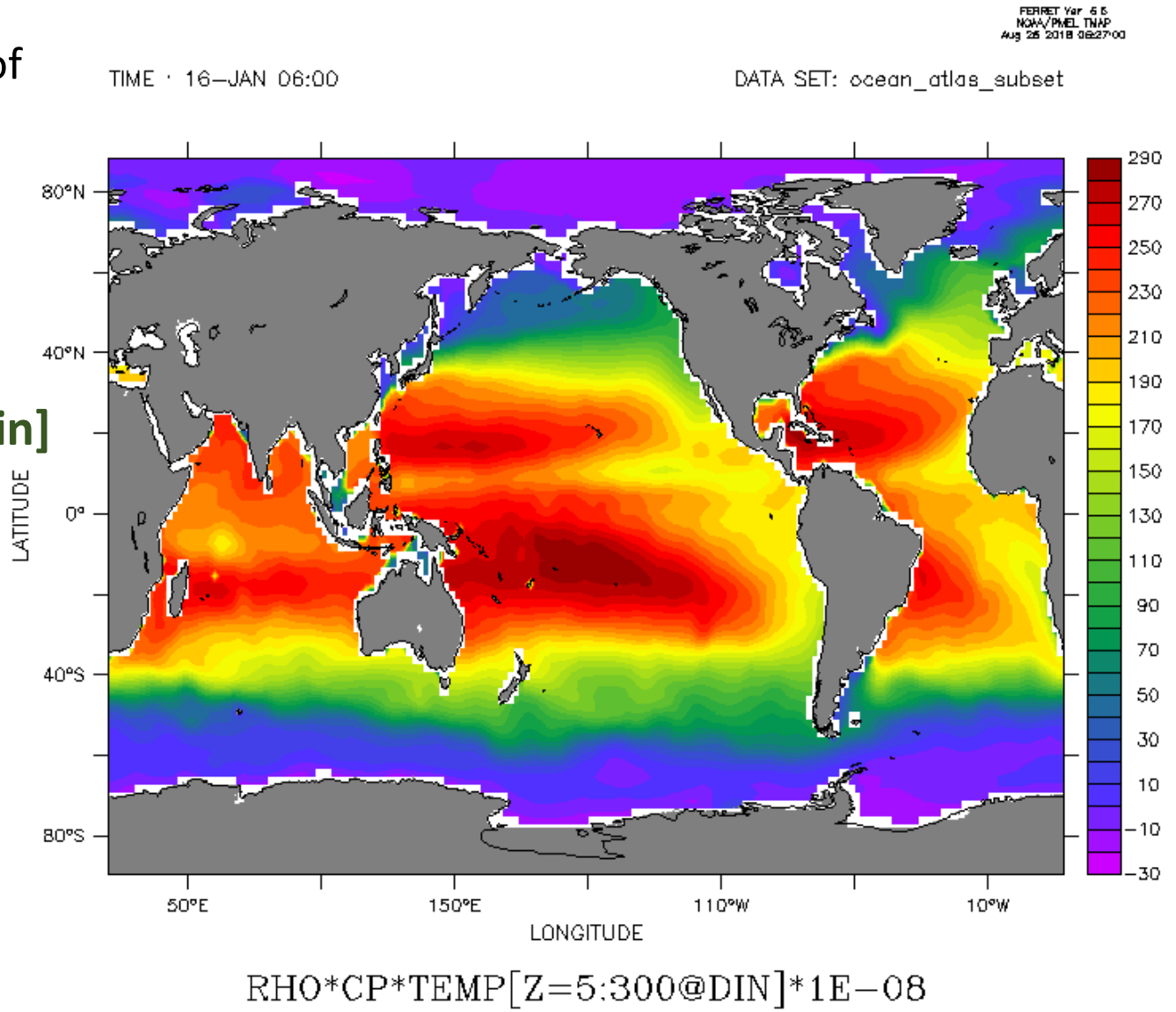
yes? let cp = 4000.5

Yes? Let rho = 1024.5

yes? Let htcnt = rho*cp*temp[z=0:300@din]

yes? Fill htcnt[l=1];go fland;go land

This will yield the heat content up to 300 mts across the global ocean.



Derivative Tranformations

The transformation @DDC computes the derivative with respect to the indicated axis using a centered differencing scheme. The units of the underlying axis are treated as they are with integrations. If the points of the axis are unequally spaced, note that the calculation used is still $(F_{i+1} - F_{i-1}) / (X_{i+1} - X_{i-1})$.

Example:

yes? PLOT/X=160W/Y=0/Z=0 u[L=1:120@DDC]

The transformation @DDF computes the derivative with respect to the indicated axis. A forward differencing scheme is used. The units of the underlying axis are treated as they are with integrations. $(F_{i+1} - F_i) / (X_{i+1} - X_i)$

Example:

yes? PLOT/X=160W/Y=0/Z=0 u[L=1:120@DDF]

The transformation @DDB computes the derivative with respect to the indicated axis. A backward differencing scheme is used. The units of the underlying axis are treated as they are with integrations. $(F_i - F_{i-1}) / (X_i - X_{i-1})$

Example:

yes? PLOT/X=160W/Y=0/Z=0 u[L=1:120@DDB]

Filling Tranformations

The transformation @FAV:n fills holes (values flagged as invalid) in variables with the average value of the surrounding grid points along the indicated axis. The width of the averaging window is the number of points given as an argument to the transformation.

The transformation @FLN:n fills holes in variables with a linear interpolation from the nearest non-missing surrounding point.

The transformation @FNR:n is similar to @FLN:n, except that it replicates the nearest point to the missing value. In the case of points being equally spaced around the missing point, the mean value is used.

Let’s Start Hands on Important Transformations

Building JNLs

From start to finish the sequence of operations needed to obtain results from Ferret is simply:

1) Specify the data set
2) Specify the region
3) Define the desired variable or expression (optional)
4) Request the output

Writing GO tools

A GO tool ("GO script," "journal file," ...) is simply a sequence of Ferret commands stored in a file and executed with the GO command. Writing a simple GO tool requires nothing more than typing normal commands into a file.

Documenting GO tools

Documentation consists primarily of well-chosen comment lines (lines beginning with an exclamation mark). In addition, a line of this form should be included:

```
! Description:  [one-line summary of your GO tool]
```

How to pass arguments to JNL file ?

Arguments (parameters) may be passed to GO tools on the command line. There is an upper limit of nine arguments allowed. For example,

```
yes? GO land red
```

passes the string "red" into the GO file named land.jnl. Inside the GO tool the argument string "red" is substituted for the string "$1" wherever it occurs. The "1" signifies that this is the first argument—similar logic can be applied to $1,... $9 or $0 where $0 is replaced by the name of the GO tool itself. Similarly "$*" is replaced by all the arguments, 1–9 as a single string, separated by spaces.

As Ferret performs the substitution of $1 (or other) arguments it offers a number of string processing and error processing options. For example, *without* these options, if a user failed to supply an argument to "GO land" then Ferret would not know what to substitute for $1 and it would have to issue an error message. A default value can be supplied by the GO tool writer using the syntax

```
$1%string%
```

for example,

```
$1%black%
```

Ferret Automation

Ferret may be ran in batch mode using the command-line switches -gif, -batch, -script.
For example to run a script which produces gif images and takes arguments:

```
% ferret -gif -script my_ferret_script.jnl 5 2 1
```

2) In addition, you can use redirection to pass a script to Ferret. for example

```
% ferret < my_ferret_script.jnl
```

and also redirect output

```
% ferret < my_ferret_script.jnl > script_output.txt
```

You can also place commands to ferret within a shell script,

```
ferret << STOP
plot/i=1:10 i
list/i=1:10 i*3
STOP
```

In short, Ferret can be treated as other executables, redirecting input and its output.

COMMON COMMANDS

Command	Description
USE	Names the data set to be analyzed (alias for "SET DATA")
SHOW DATA	Produces a summary of a variable
SHOW GRID	Examines the coordinates of a grid
SET REGION	Sets the region to be analyzed
LIST	Produces a listing of data
PLOT	Produces a plot
CONTOUR	Produces a line contour plot
FILL	Produces a color filled contour plot
SHADE	Produces a shaded-area plot
VECTOR	Produces a vector arrow plot
POLYGON	Plots polygonal regions
DEFINE	Define new axes, grids, and symbols
STATISTICS	Produces summary statistics about variables and expressions
LET	Defines a new variable
SAVE	Saves data in NetCDF format
GO	Executes Ferret commands contained in a file

Demonstration files

Name	Description
tutorial	Brief tour through Ferret capabilities
topographic_relief_demo	Global topography
coads_demo	View of global climate using the Comprehensive Ocean-Atmosphere Data Set
levitus_demo	T-S relationships using Sydney Levitus' climatological Atlas of the World Oceans
fnoc_demo	Naval Fleet Numerical Oceanography Center data
vector_demo	Vector plots
wire_frame_demo	3D wire frame representation
custom_contour_demo	Customized contour plots
viewports_demo	Output to viewports
multi_variable_demo	Multiple variables with multiple dependent axes
objective_analysis_demo	Interpolating scattered data to grids
mp_demo	Map projections demo
log_plot_demo	Log plots using PPLUS in Ferret
depth_to_density_demo	Contour with a user-defined variable as an axis
file_reading_demo	Reading an ASCII file
regridding_demo	Tutorial on regridding data
mathematics_demo	Abstract function calculation
statistics_demo	Probability distributions
spirograph_demo	For-fun plots from abstract functions
splash_demo	For-fun mathematical color shaded plots
symbol_demo	How to use symbols for plot layouts
sigma_coordinate_demo	How to work with sigma coordinates

Let's Start Hands on Building JNL and Go Files


thank·you