Introduction to IDL
Optimization & Statistical Techniques

Week 12
Rachel McCrary
4-14-2009
Goals for today!

- Build your own color table
- Rebin/Reform + avoiding loops
- Regression
- EOFs
- Power Spectra
- file_search
- setting your path.
Creating your own color tables

- As nice as the IDL predefined color tables are ... they are sometimes seriously lacking. For example IDL does not have a good color table for plotting anomalies (with white in the center, cool colors on one side and warm colors on the other).

- Fortunately, you can create your own color tables in IDL!

- This fairly advanced stuff ...
color tables

• The basic concept:
  • Identify the RGB indices (or color triples) for the colors you want to put in your color table.
  • Create an \((x, 3)\) integer array that contains these color triples
  • Create a new integer array that is \((256, 3)\) and put the color triples into the order you want them in your color table
  • use the tvlct command to save the color table into something IDL can use!
  • See: colors_anomalies.pro
The rebin function enlarges or shrinks an array by an integer multiple or factor of the current dimensions:

/sample or sample = 1 causes the nearest neighbor sampling to be used for enlarging or shrinking.

```
IDL> arr = [10, 30, 50]
IDL> print, rebin(arr, 9, /sample)
10 10 10 30 30 30 50 50 50
```

If the sample keyword is not set (sample = 0) then linear interpolation is used for enlarging

```
IDL> print, rebin(arr, 9)
10 16 23 30 36 43 50 50 50
```

rebin does not extrapolate past the end of the array
rebin

- When shrinking an array, rebin uses neighborhood averaging.
- Remember must enlarge/shrink by a factor of multiple of the current dimension.

IDL> arr = indgen(9)
IDL> print, arr
   0    1    2    3    4    5    6    7    8
IDL> print, rebin(arr, 3)
   1    4    7
IDL> print, rebin(arr, 4)
% REBIN: Result dimensions must be integer factor of original dimensions
% Execution halted at: $MAIN$ 31 /Users/rachel/idl_course_week4/rmv_season.pro
• *rebin* is also useful when you wish to create multidimensional grid arrays.

• Eg. create a grid of values that increase along the column dimension:

```idl
IDL> v = indgen(5)
IDL> print, v
   0  1  2  3  4
IDL> x = rebin(v, 5, 3)
IDL> print, x
   0  1  2  3  4
   0  1  2  3  4
   0  1  2  3  4
```
The reform function changes the dimensions of an array without changing the total number of elements.

```idl
IDL> v = indgen(5)
IDL> print, v
   0 1 2 3 4
IDL> y = reform(v, 1, 5)
IDL> help, y
Y           INT       = Array[1, 5]
IDL> print, y
   0
   1
   2
   3
   4
```
to create a grid of values that increase along the row dimension, the 1-D array must be reordered by calling reform, then rebin can be used to create the grid

IDL> v = indgen(5)
IDL> y = reform(v, 1, 5)
IDL> help, y
Y       INT             = Array[1, 5]
IDL> print, y
  0
  1
  2
  3
  4
IDL> y = rebin(y, 3, 5)
IDL> print, y
  0   0   0
  1   1   1
  2   2   2
  3   3   3
  4   4   4
the rebin function can be used in conjunction with reform to create index arrays with multiple dimensions

IDL> arr = indgen(3)
IDL> print, rebin(arr, 3,3)

0 1 2
0 1 2
0 1 2
0 1 2
IDL> print, rebin(reform(arr, 1, 3), 3, 3)

0       0       0
1       1       1
2       2       2
```idl
IDL> print, rebin(reform(arr, 1, 1, 3), 3, 3, 3)
```

```
0  0  0  0
0  0  0  0
0  0  0  0

1  1  1  1
1  1  1  1
1  1  1  1

2  2  2  2
2  2  2  2
2  2  2  2
```
Use of rebin/reform

- You can use rebin/reform to “optimize” removing the seasonal cycle from a timeseries of monthly data.

- see: rmv_season.pro

- more advanced stuff can be found here:

regression

- The regression function performs a multiple linear regression fit and returns a column vector of coefficients. (x and y can be 1-D vectors, or multiple dimensional vectors)


- regress accepts a number of keywords which are very useful

\[ y = m x + b \]
regression keywords

• X - independent variable, y - dependent variable

• result - regression coefficient

• chisq - set keyword to named variable that will contain the value of the unreduced chi-square goodness-of-fit statistic.

• const - set keyword to named variable that will contain the constant term of the fit (b).

• correlation - set keyword to named variable that will contain the linear correlation coefficients.

• /double

• ftest - set this keyword to a named variable that will contain the F-value for the goodness-of-fit test.

• yfit - set this keyword to a named variable that will contain the vector of calculated y values
regression example

• Calculate the linear trend in a timeseries of global mean temperature anomalies: temp_trend.pro

• Calculate the regression/correlation coefficients for global mean temp vs. sun spot activity: temp_vs_sunspot.pro
EOFs

1. Decompose matrix $A$ such that:
   - $A = U\Sigma V^T$

2. Do this using eigenanalysis, or using the SVD routine in IDL
   - $A$ - space x time array, seasonal cycle removed, weighed by the square root of latitude
   - Columns of $V$ are the EOFs
   - Columns of $U$ are the PCs

3. Standardize the PC time series

4. Regress the original data onto the PC time series
   (PC time series is the independent variable)
SVDC

- The SVDC procedure computes the Singular Value Decomposition (SVD) of a square (n x n) or non-square (n x m) array as the product of orthogonal and diagonal arrays.

- Use SVD to calculate: \( A = U \Sigma V^T \)


- See: eof_example.pro
Steps for Spectral Analysis

- Subdivide the data
- Find the lag-1 autocorrelation of your timeseries and calculate the e-folding timescale $T = -\Delta T/\ln(a)$
- Calculate the red-noise spectrum: $\Phi_R = 2T/(1+T^2W^2)$
- Normalize the area under the red noise spectrum to 1
- Apply window (if you are using one)
- Use FFT to find: $C^2 = A^2 + B^2$ (you want $C^2/2$)
- Apply running mean (if that's how you want to smooth $C^2/2$)
- Normalize the area under the spectrum to 1
- Find degrees of freedom ($N/M^*$)
- Plot, red-noise, spectrum, and 95% confidence line as a function of $k, w, f$
Spectral Analysis

- To calculate the power spectra of a timeseries, you will want to use the FFT procedure in IDL.

- The FFT function returns a result equal to the complex, discrete Fourier transform of Array. The result of this function is a single- or double-precision complex array.

- \[ \text{Result} = \text{FFT} \left( \text{Array} \ [, \ \text{Direction}] \ [, \ \text{DIMENSION} = \text{scalar}] \ [, \ /\text{DOUBLE}] \ [, \ /\text{INVERSE}] \ [, \ /\text{OVERWRITE}] \right) \]

- See `power_spec_example.pro`
The `file_search` function returns a string array containing the names of all files matching the input path specification.

- The file search function returns a string array containing the names of all files matching the input path specification.
- Input path specifications may contain wildcard characters, enabling them to match multiple files.
- `filelist = file_search(path_specification)`

see: `find_files.pro`
Changing your path

- As I mentioned previously, to organize my .pro files, I have created a single directory where I put all of my own idl scripts.
- I then organize all of my files within subdirectories of this main IDL directory.
- I then have a IDL startup file that I run every time I open IDL, which instructs IDL to always look in this directory for .pro files to run.
- I like this because then I can access my .pro files from any directory on my computer, and I don’t have to copy and paste things like my color table procedures into random places.
- I also know where to find all of my code!!!!!!!!
my IDL startup file contains this:

!path = !path + ':' + expand_path('+/Users/rachel/idl')

!path is an IDL system variable, which determines the directories that IDL searches for, and includes files and programs written in the IDL language (.pro)

I am appending my IDL directory to the end of the !path system variable.

the + in front of the directory tells IDL to also look for .pro files within any subdirectories within my IDL directory
Questions???
A histogram is just a fancy way to count: a series of numbers in an input vector is divvied up into bins according to their values and the size of the histogram bin. For each input value that falls into a given bin, a count is added to the bin.


http://www.dfanning.com/tips/histogram_tutorial.html