Introduction to IDL

5 – Graphics

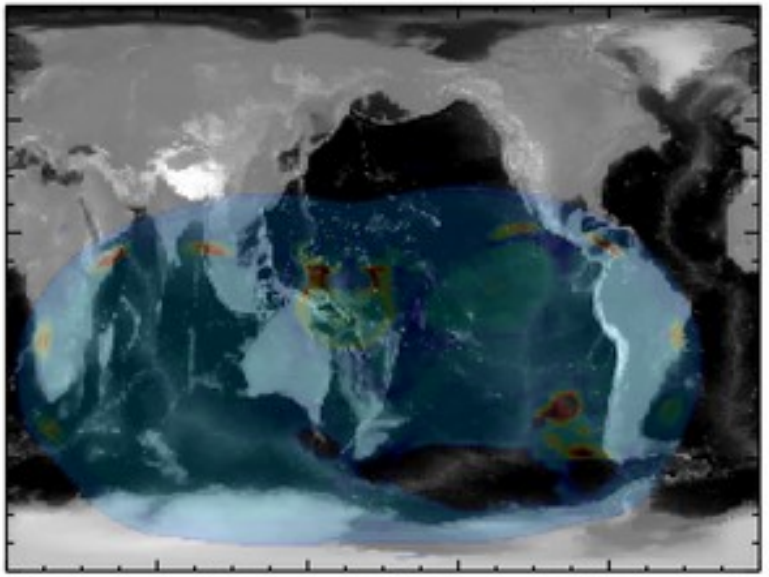
Paulo Penteado
pp.penteado@gmail.com
http://www.ppenteado.net

IDL Graphics

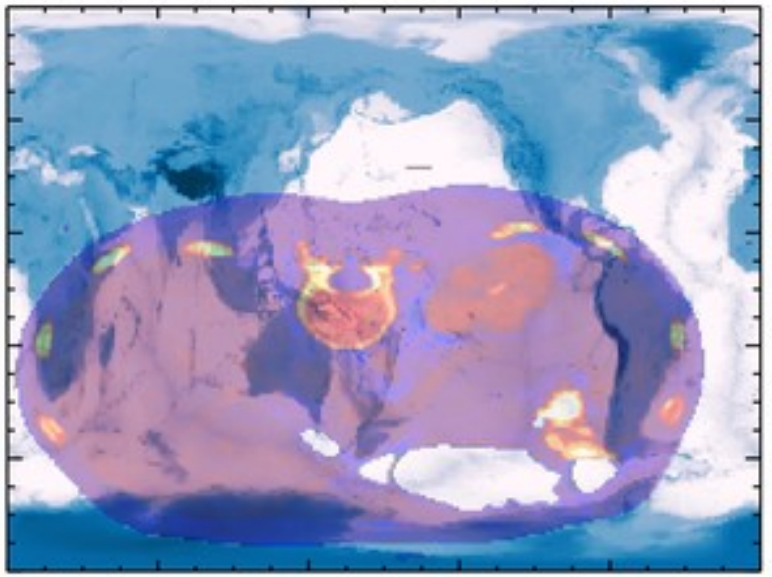
- Besides being oriented towards scientific computing and array processing, IDL always had an extensive and powerful library for graphics.
- Graphics can be static or interactive.
- Graphic creation is platform-independent.

Multi-Transparent Images

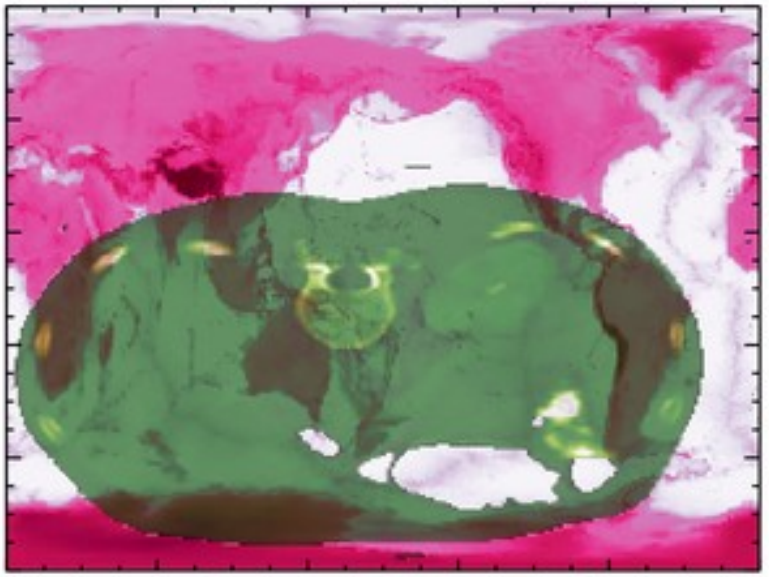
80% Transparent



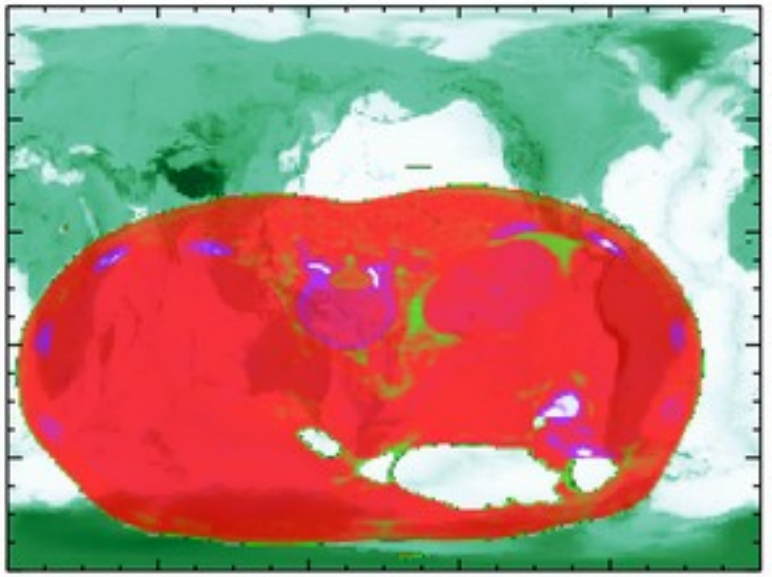
60% Transparent



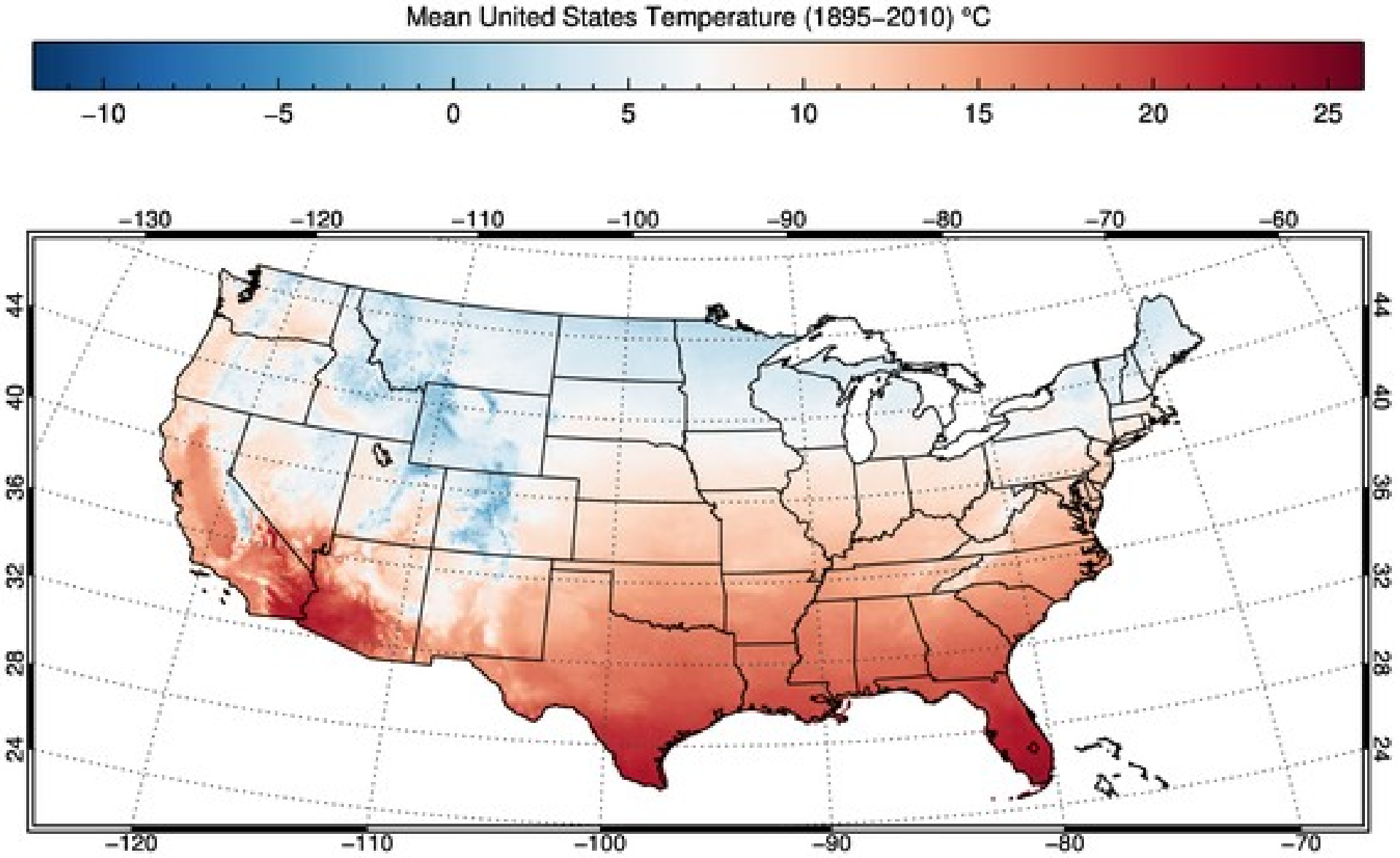
40% Transparent



20% Transparent

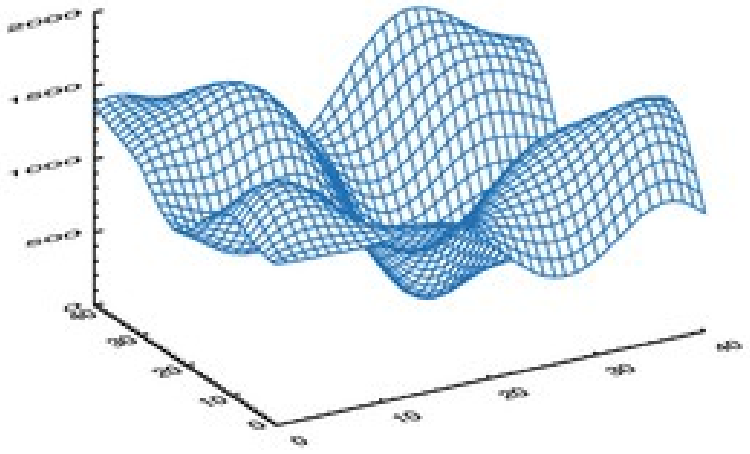


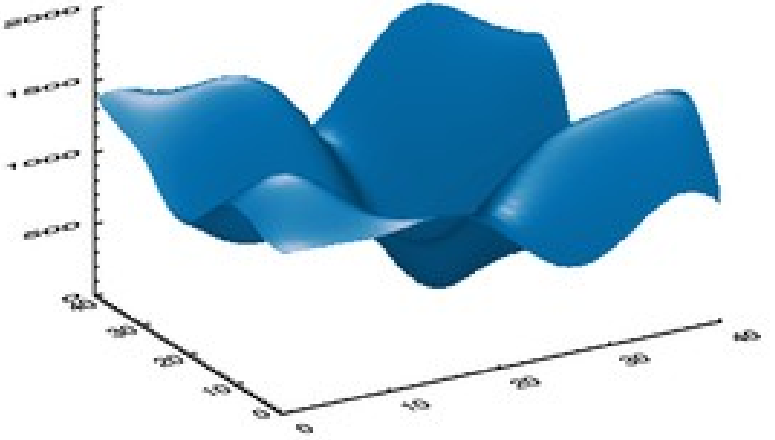
http://www.idlcoyote.com/gallery/

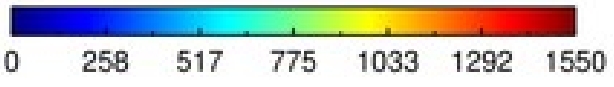


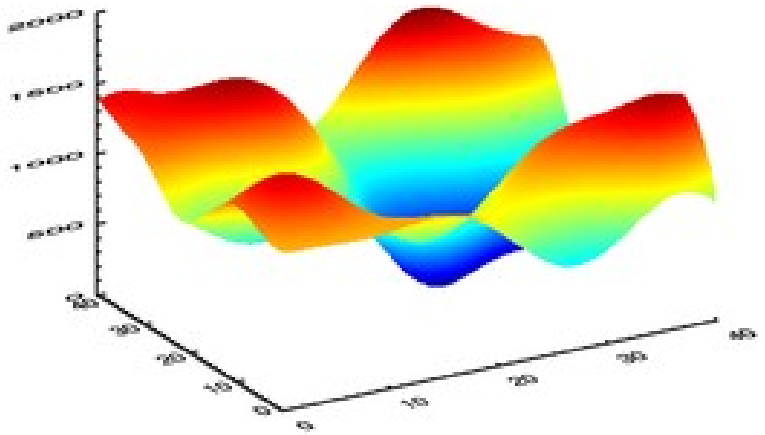
http://www.idlcoyote.com/gallery/

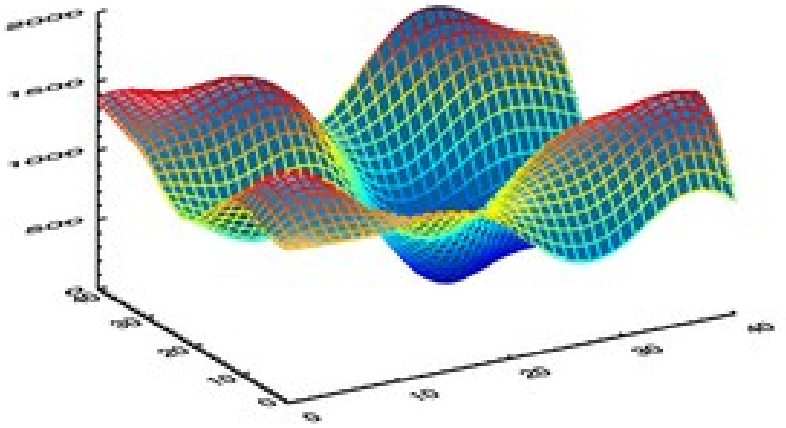
Example Surface Plots



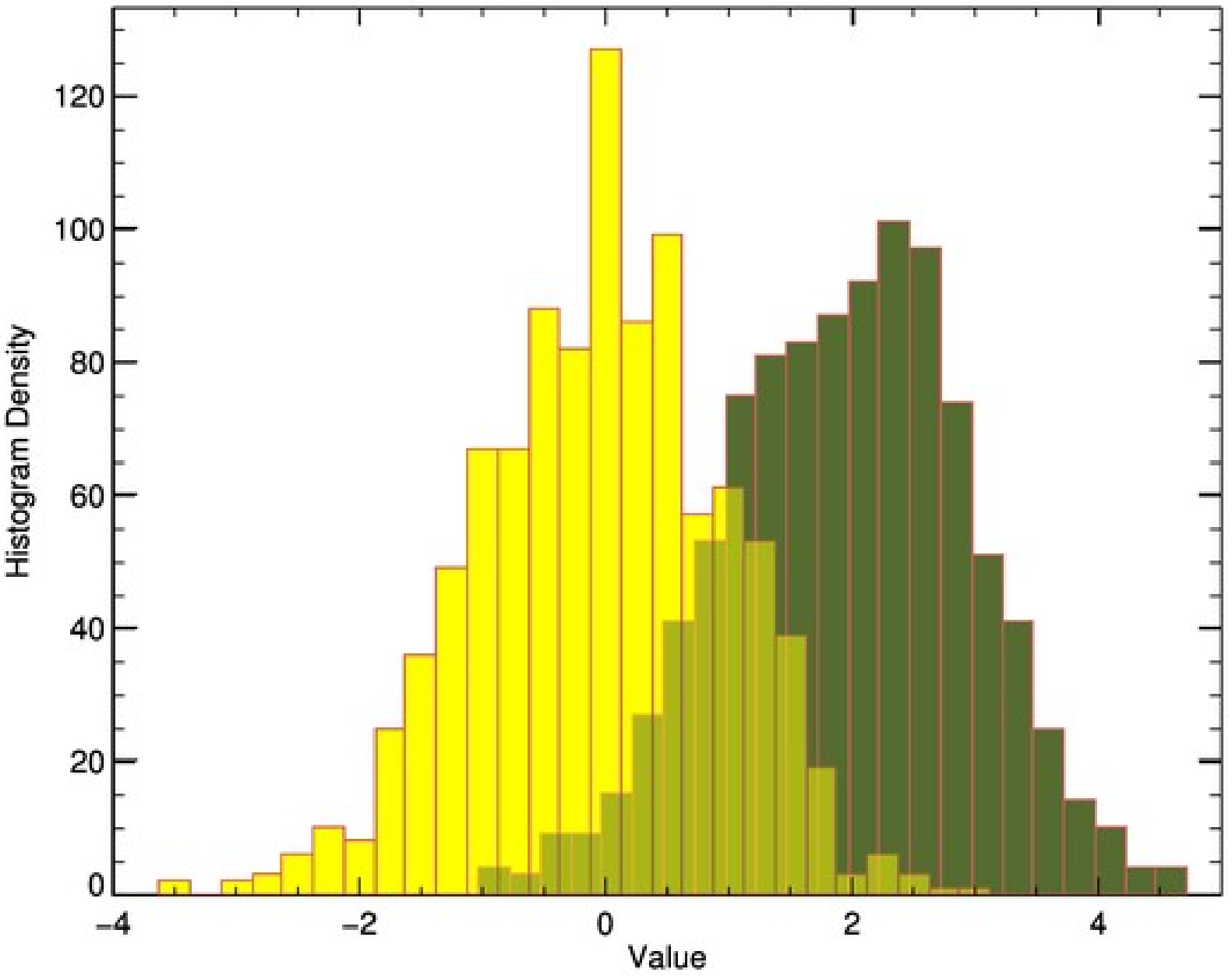




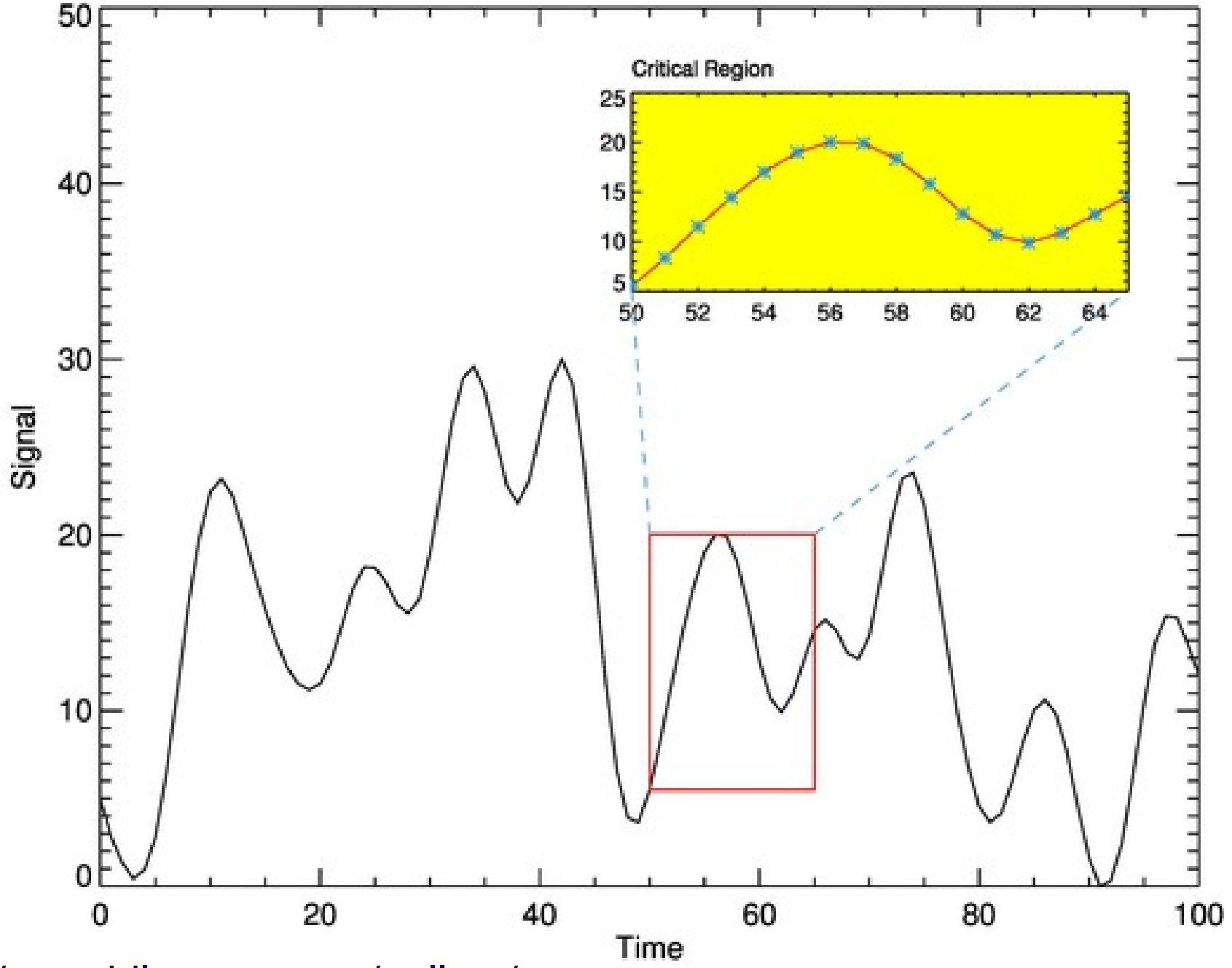




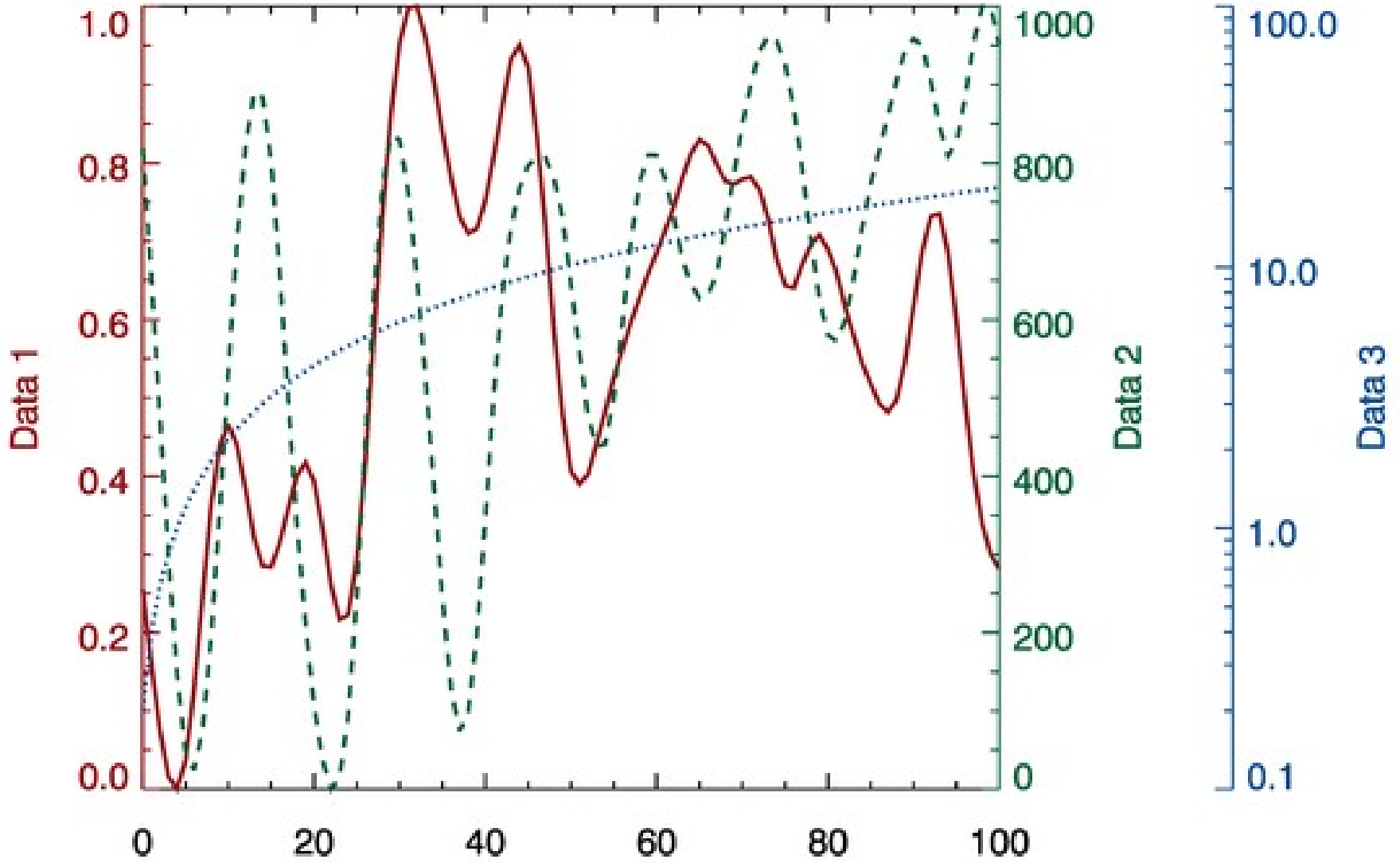
http://www.idlcoyote.com/gallery/



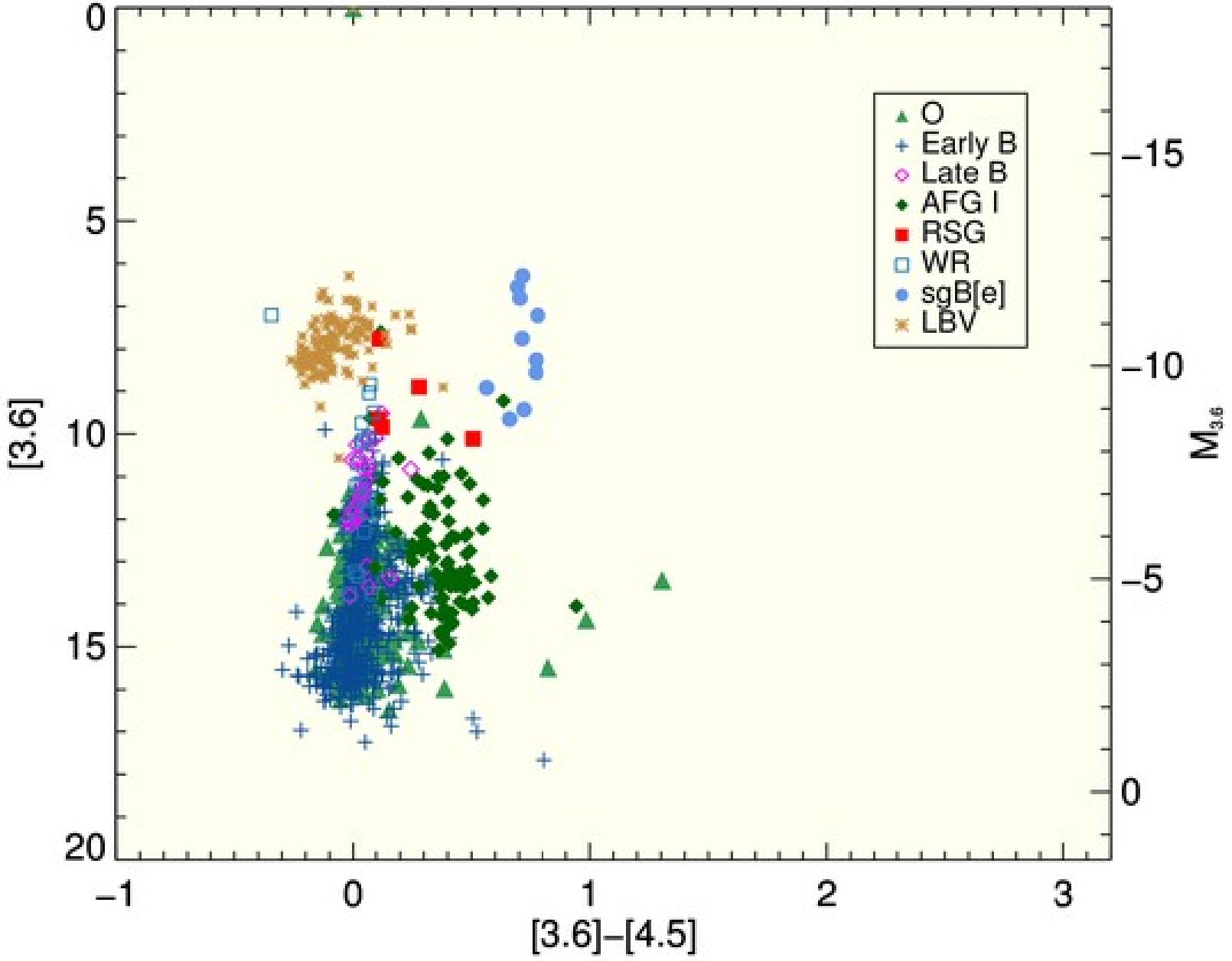
http://www.idlcoyote.com/gallery/



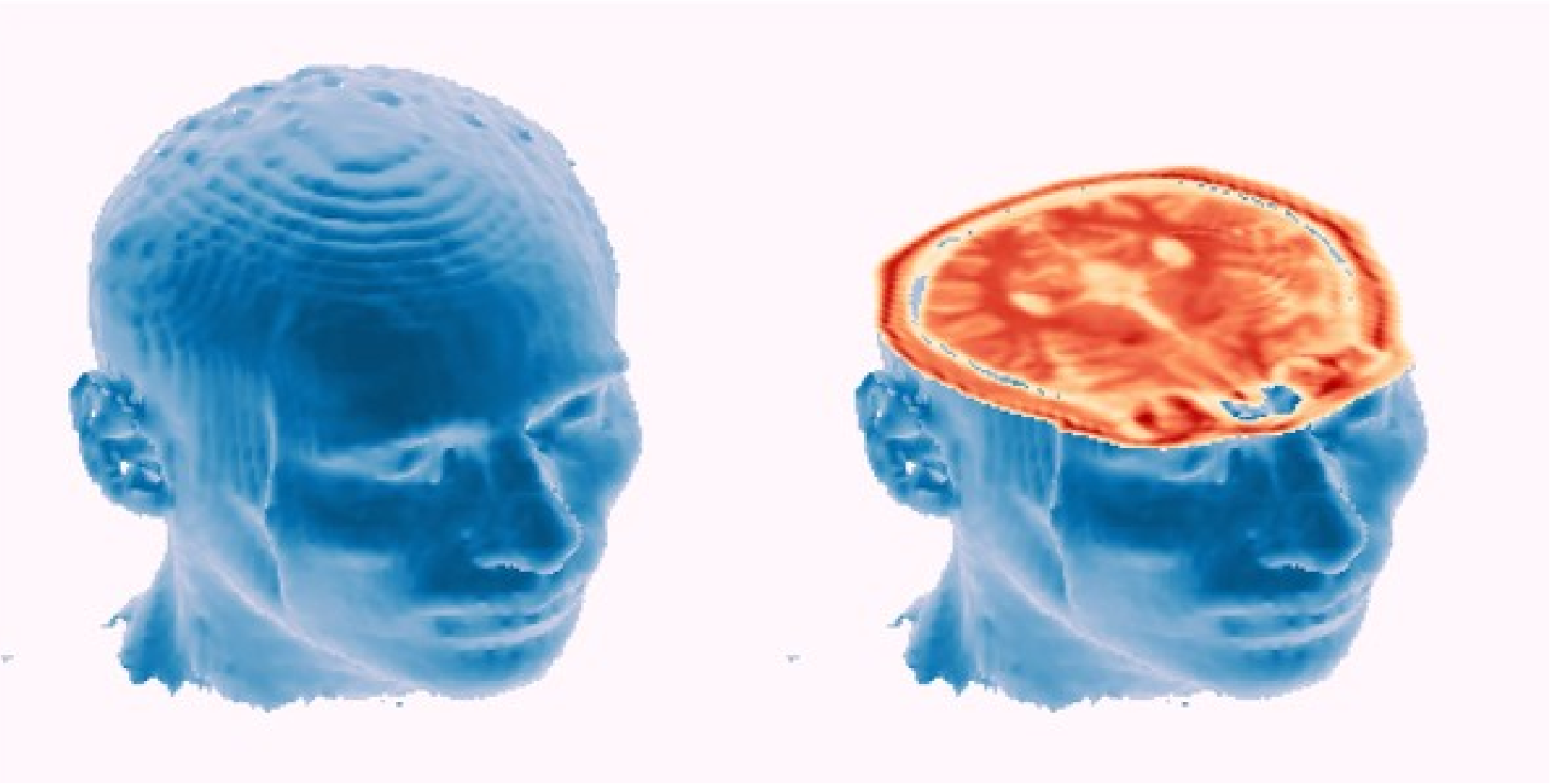
http://www.idlcoyote.com/gallery/



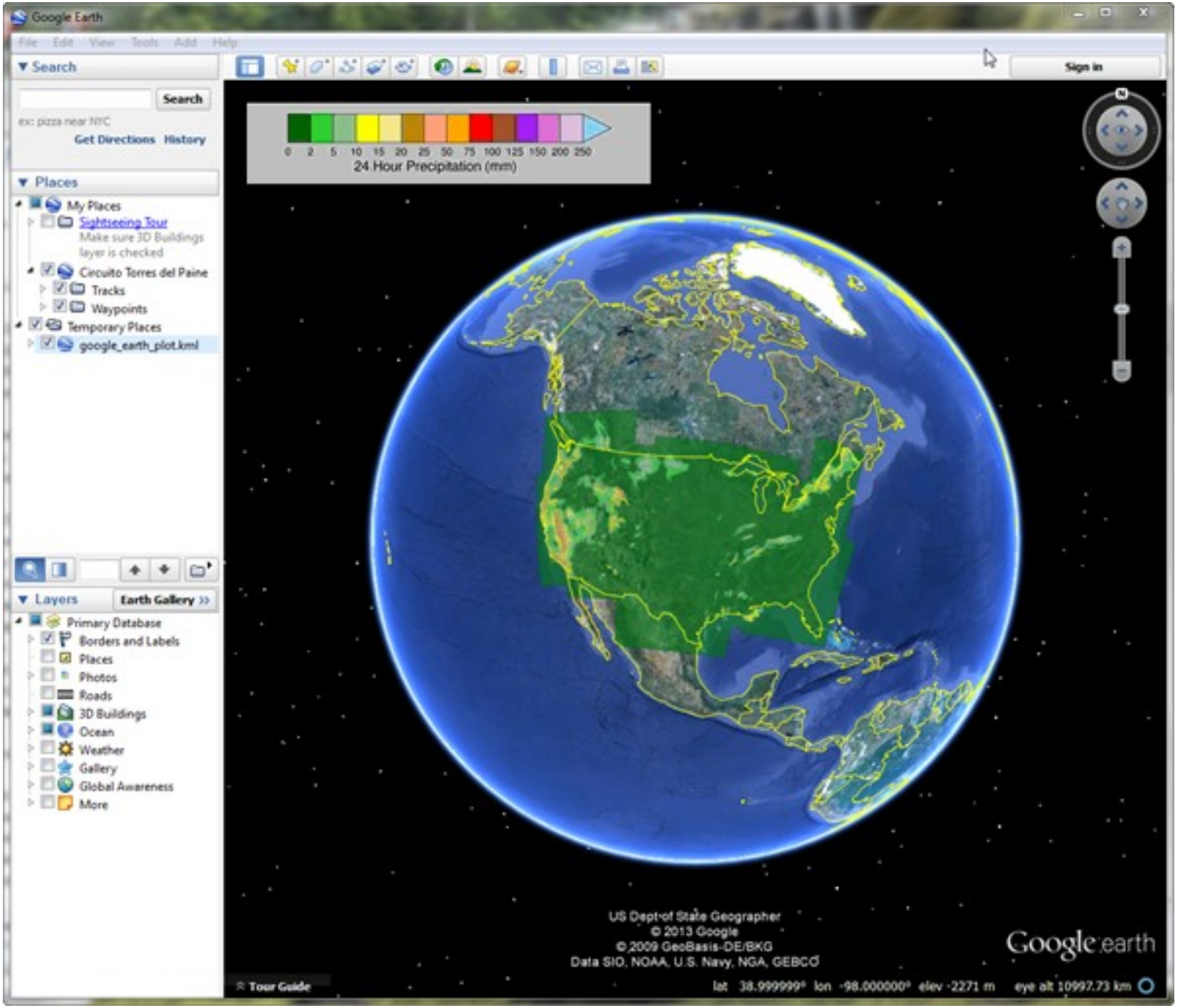
http://www.idlcoyote.com/gallery/



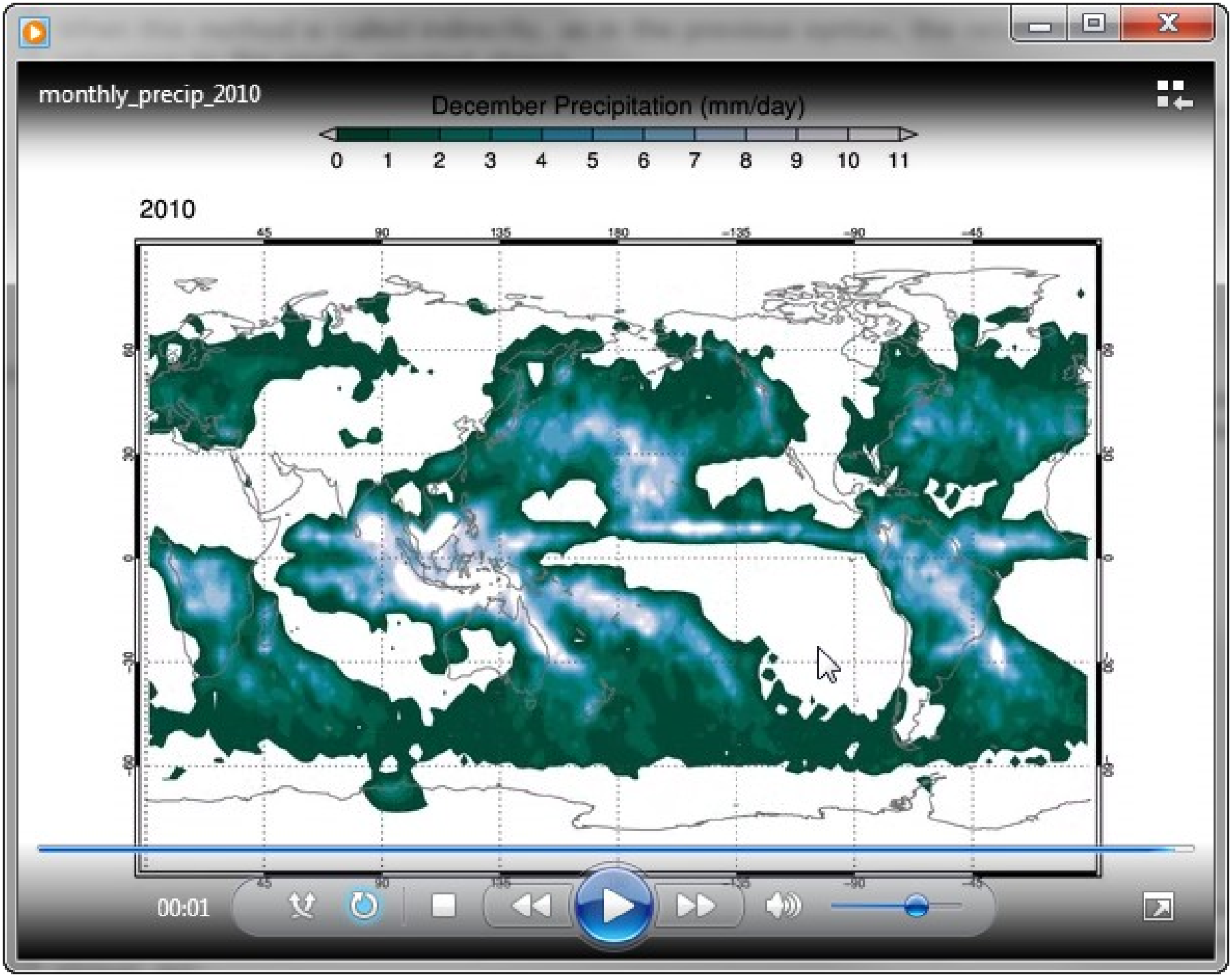
http://www.idlcoyote.com/gallery/



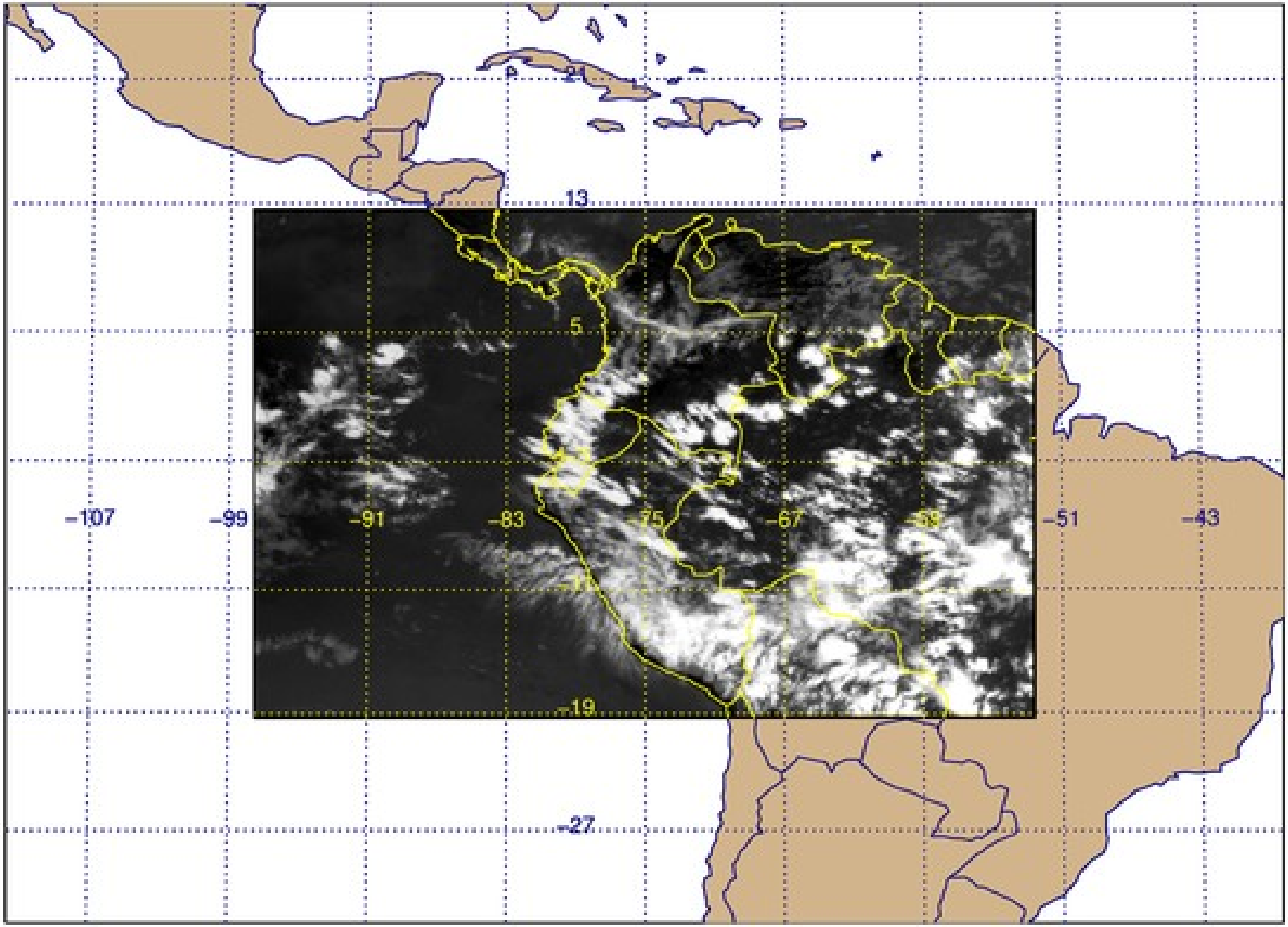
http://www.idlcoyote.com/gallery/



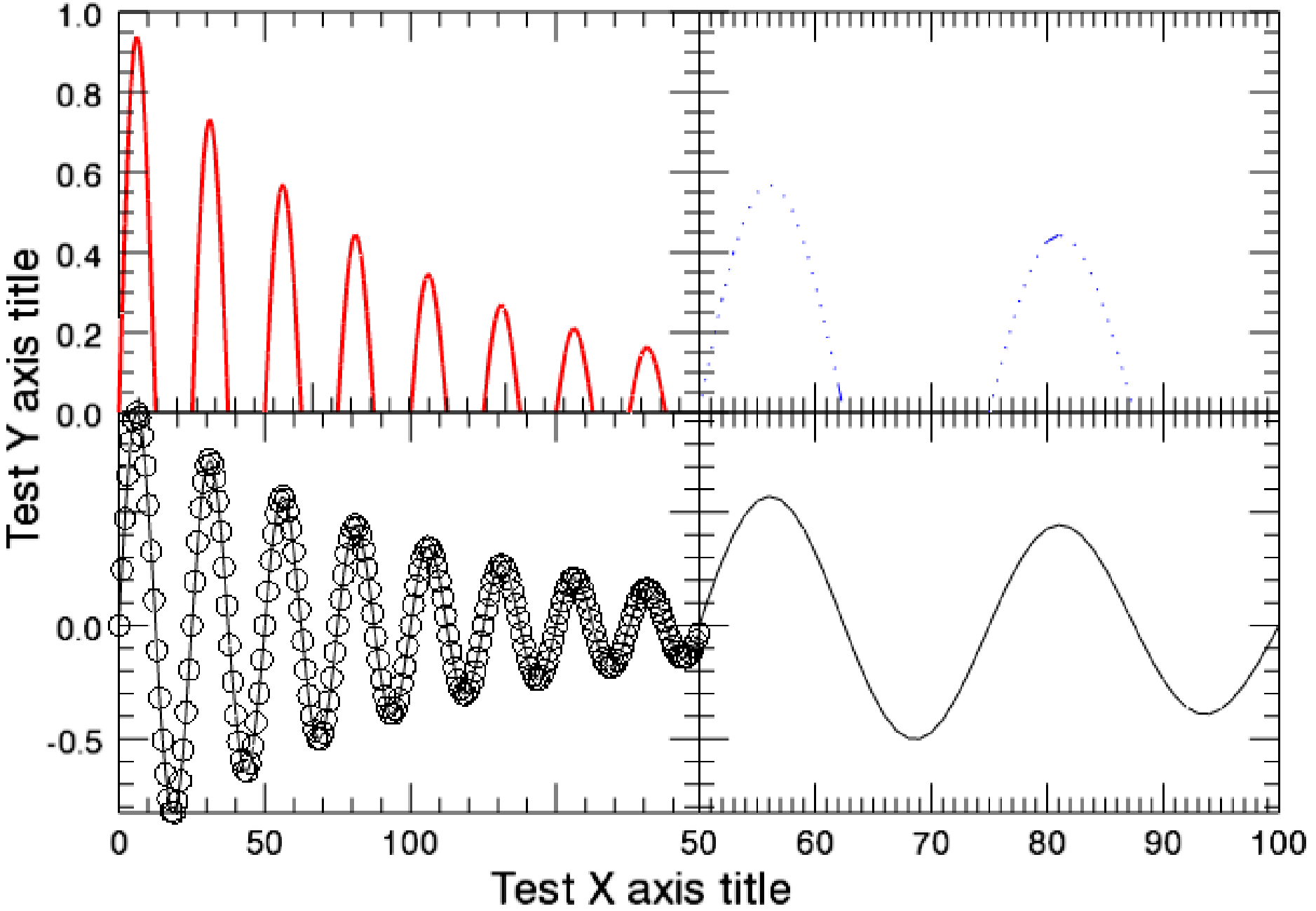
http://www.idlcoyote.com/gallery/



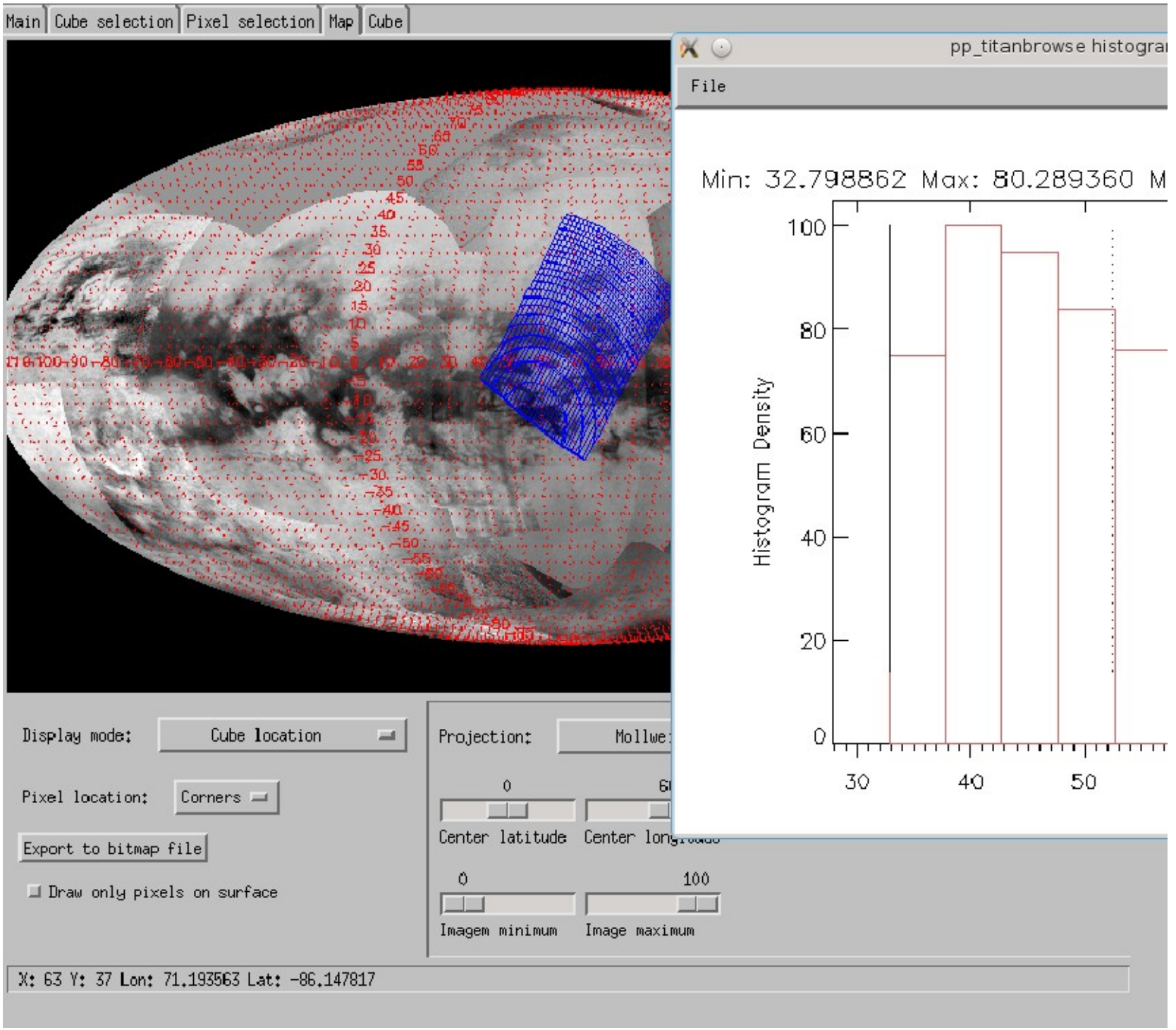
http://www.idlcoyote.com/gallery/



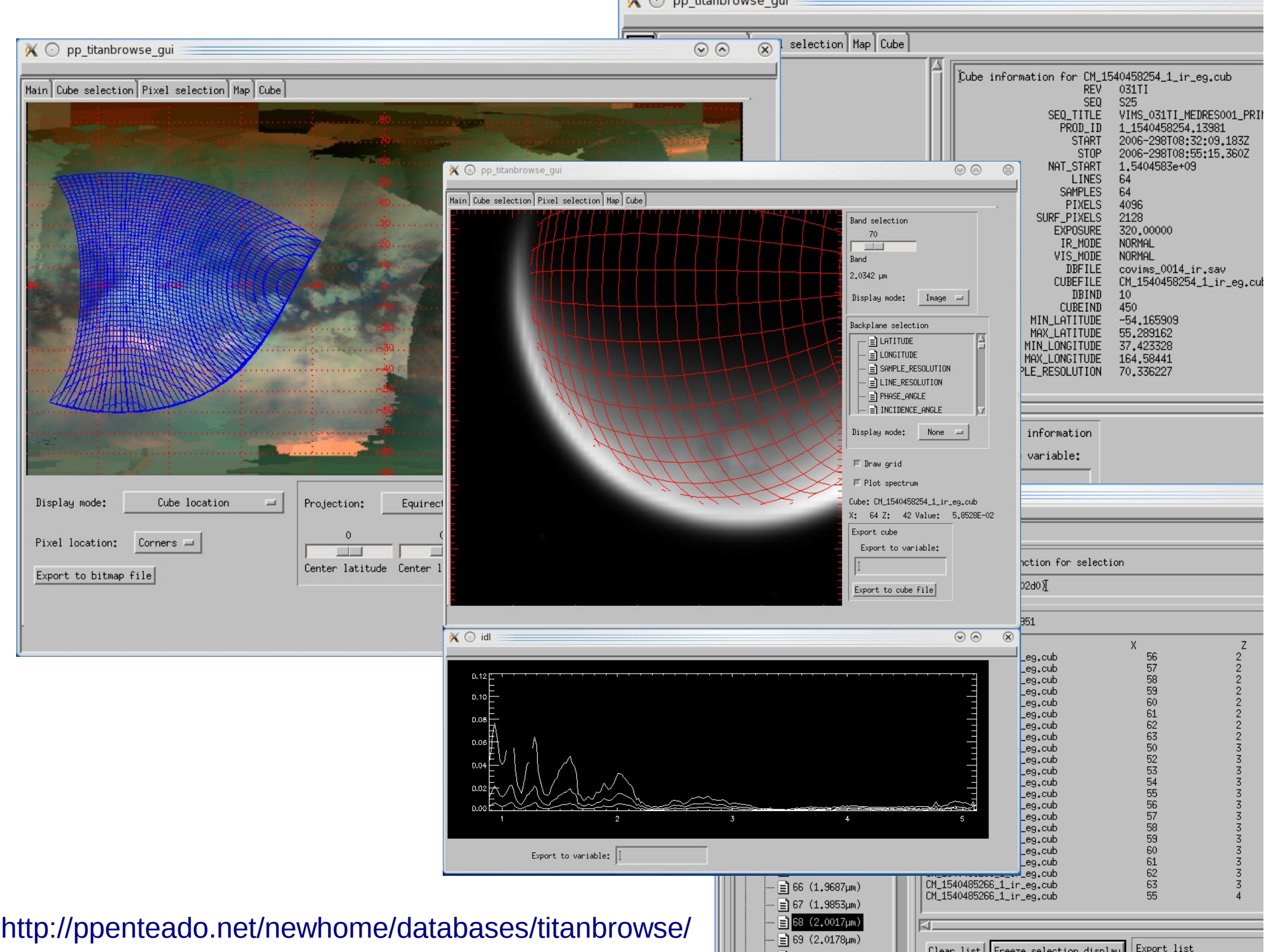
http://www.idlcoyote.com/gallery/



http://www.ppenteado.net/idl/pp_lib/doc/pp_multiplot__define.html



http://ppenteado.net/newhome/databases/titanbrowse/



http://ppenteado.net/newhome/databases/titanbrowse/

IDL Graphics Systems

- Direct Graphics (“Traditional Graphics”)
- Object Graphics
- iTools
- Function Graphics
- Coyote Graphics*
  - Part of the Coyote Library, not IDL's standard library.
  - http://www.idlcoyote.com/documents/programs.php

IDL Graphics Systems

- Direct Graphics (“Traditional Graphics”)
  - The oldest system – the only known to many users.
  - Graphics are static: they are “printed” into the graphics device (screen / file).
  - Any interactivity depends on the user programming, to redraw graphics as needed.
  - Device-dependent.

IDL Graphics Systems

- Object Graphics
  - A highly complex system.
  - Graphics elements (lines, symbols, surfaces, etc.) are editable objects.
  - Allows more complex graphics (particularly in 3D).
  - **A lot of work** to use.

IDL Graphics Systems

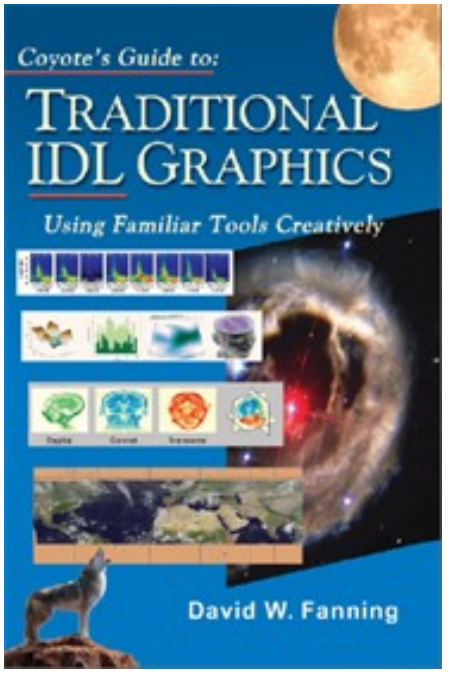
- iTools
  - NOT made by Apple.
  - An easy to use interface to object graphics.
  - Easy to make editable, savable, interactive graphics.
  - Graphics can be edited interactively or programmatically.
  - Some less common operations require knowledge beyond what the documentation supports.
- http://www.exelisvis.com/docs/iTools.html
- http://modernidl.idldev.com/
- https://groups.google.com/forum/#!forum/comp.lang.idl-pvwave

IDL Graphics Systems

- Function Graphics (“New Graphics”)
  - A new (IDL 8.0) interface for object graphics.
  - Built on top of iTools.
  - Where new development currently occurs (though some features are usable through iTools).
  - Provides new tools with more resources to make editing graphic objects easier.
  - New functionality for more modern annotation (new, easier to use, symbols, fonts, colors, styles, etc.)
  - Very extensive gallery of examples and documentation: http://www.exelisvis.com/docs/visualize.html

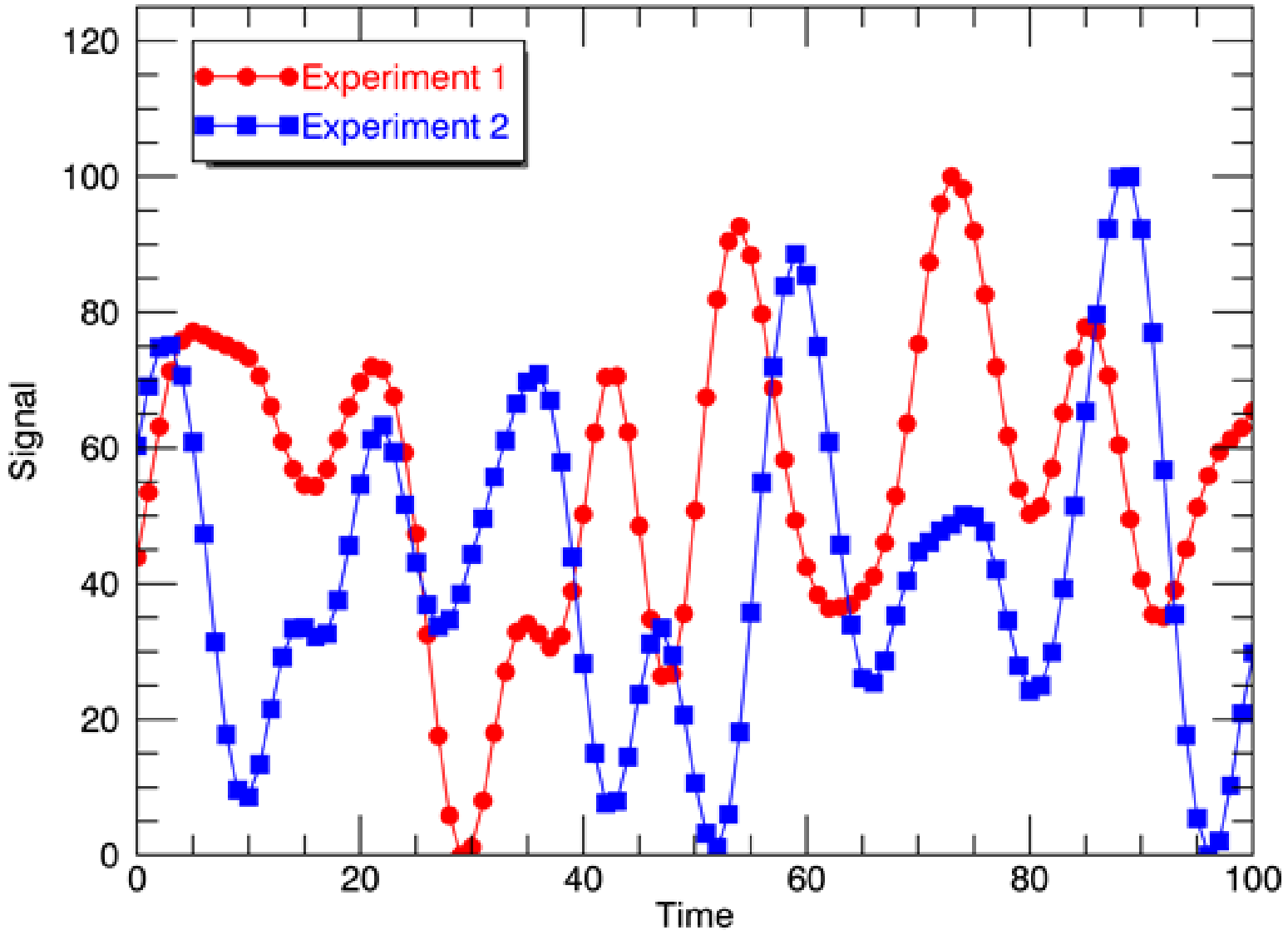
IDL Graphics Systems

- Coyote Graphics
  - Made by David Fanning (part of Coyote Library)
  - A more modern and powerful interface to direct graphics.
  - Makes it easier to create interactive, device-independent, complex graphics.



http://www.idlcoyote.com/graphics_tips/coyote_graphics.php

Plots



http://www.idlcoyote.com/gallery/

Plots

- Direct graphics

```
IDL> plot,x,y,title='plot title',xtitle='x title',$
ytitle='y title',linestyle=1
```

- iTools

```
IDL> iplot,x,y,title='plot title',xtitle='x title',$
ytitle='y title',color='red',linestyle='dotted'
```

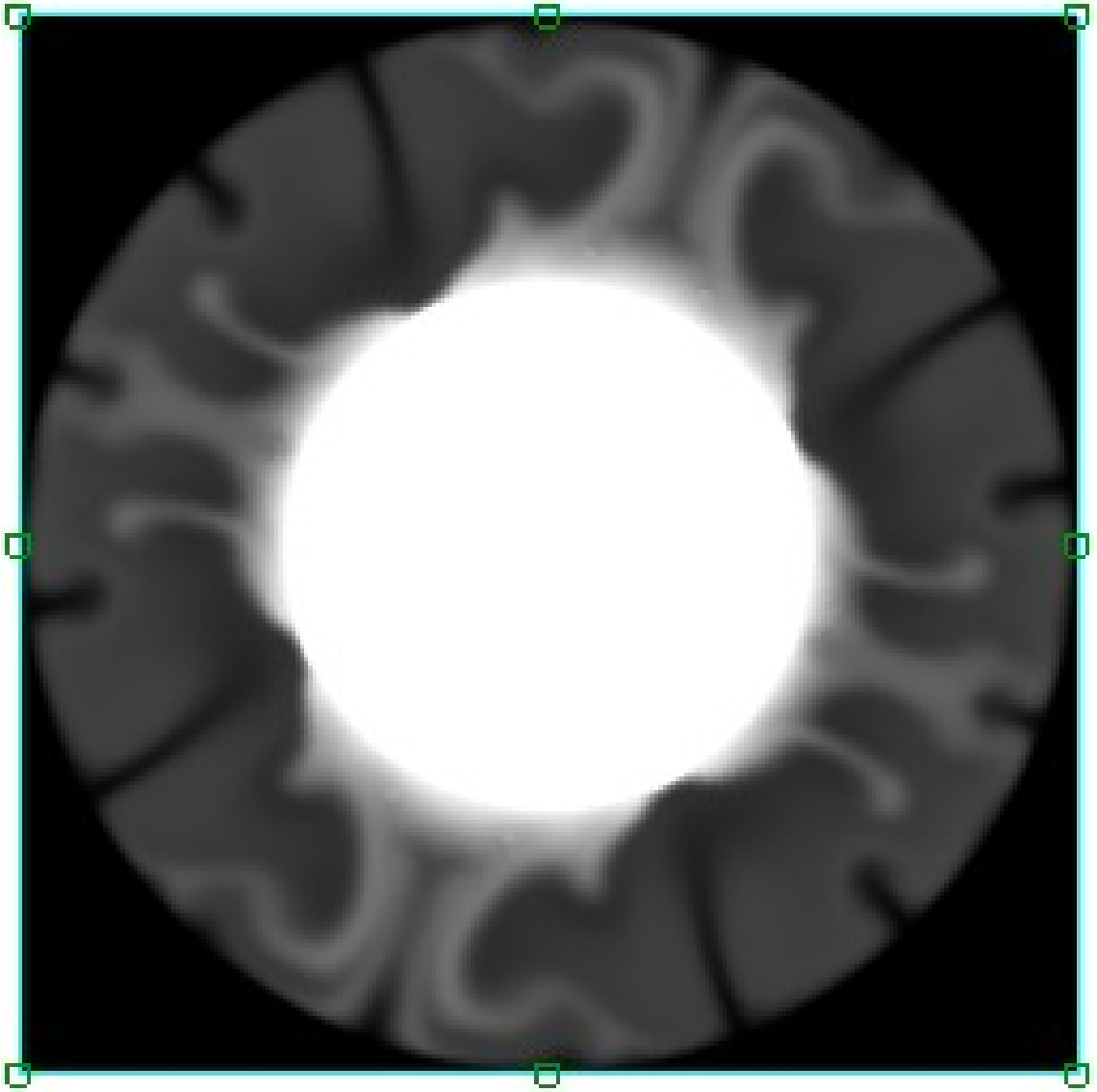
- Function Graphics

```
IDL> p=plot(x,y,title='plot title',xtitle='x title',$
ytitle='y title',color='red',linestyle='dotted')
```

- Coyote Graphics

```
IDL> cgplot,x,y,title='plot title',xtitle='x
title',ytitle='y title',color='red',linestyle=1
```

Images



http://www.exelisvis.com/docs/images_gallery.html

Images

- Direct graphics

```
IDL> tvscl,z
```

- iTools

```
IDL> iimage,z,xtitle='x',ytitle='y',rgb_table=3,/insert_colorbar
```

- Function Graphics

```
IDL> im=image(z,xtitle='x',ytitle='y',rgb_table=3)
IDL> c=colorbar()
```

- Coyote Graphics

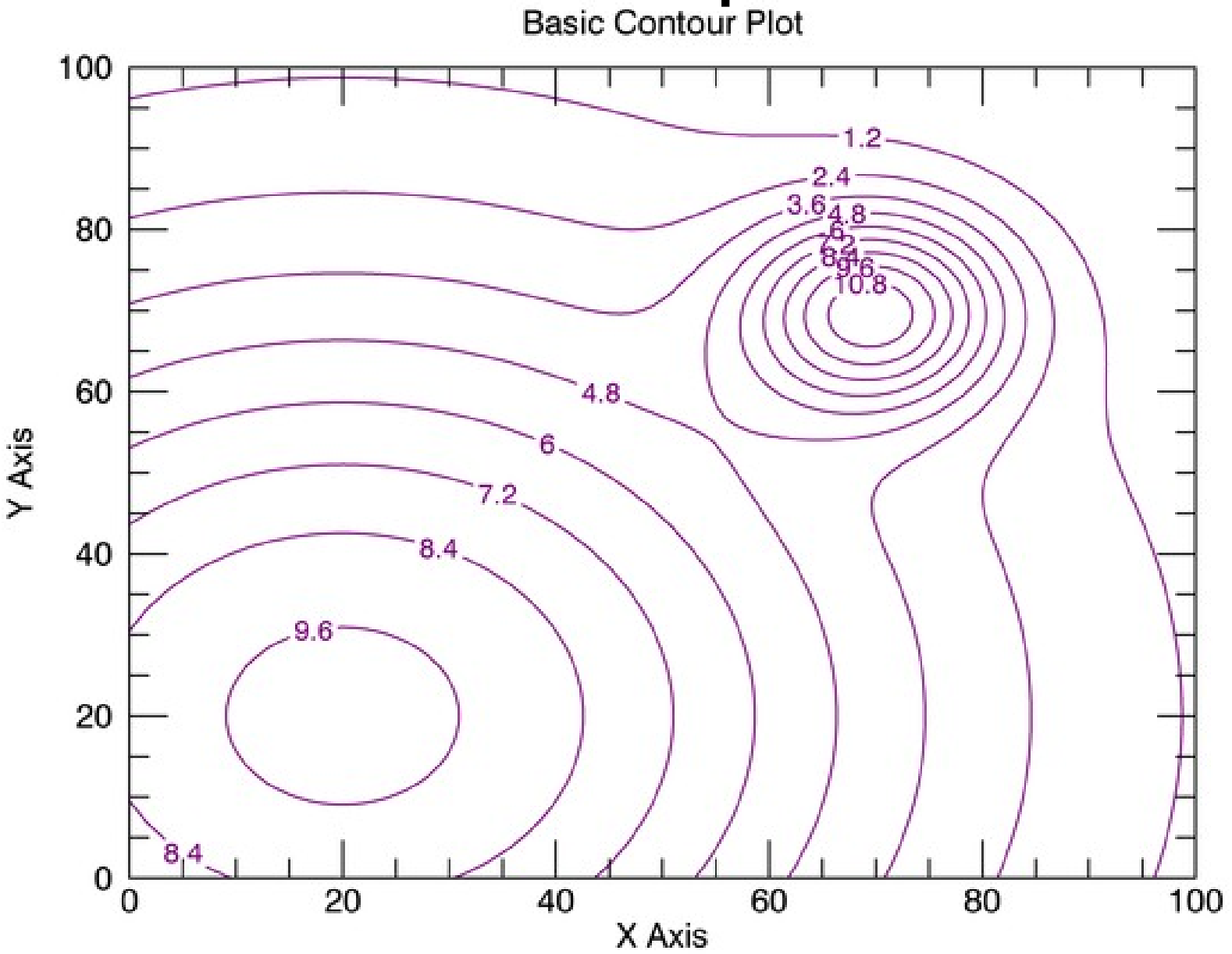
```
IDL> cgimage,z,xtitle='x',ytitle='y'
```

Images

- Images may also be 3D arrays: 2 spatial dimensions, 1 color dimension.
- The color dimension has size 3 (RGB color) or 4 (RGBA color, for transparency).

```
IDL> dataFilePath =$
FILEPATH('marsglobe.jpg',SUBDIR=['examples','data'])
IDL> im=read_image(datafilepath)
IDL> help,im
IM              BYTE      = Array[3, 400, 400]
IDL> i=image(im)
```

Contour plots



http://www.idlcoyote.com/gallery/

Contour plots

- Direct graphics

```
IDL> contour,z,xtitle='X',ytitle='Y',thick=2.
```

- iTools

```
IDL> icontour,z,xtitle='X',ytitle='Y',thick=2.,color='red'
```

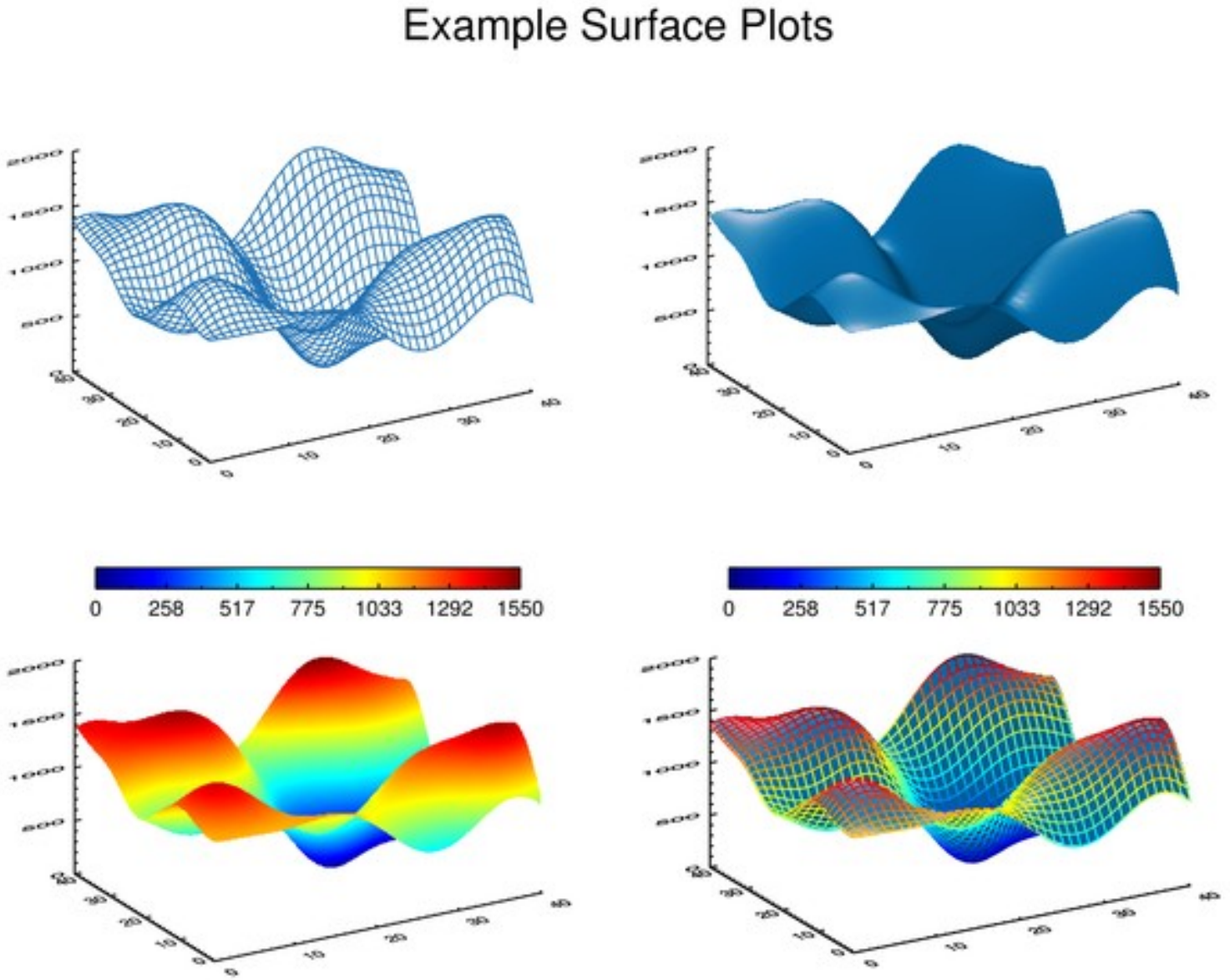
- Function Graphics

```
IDL> c=contour(z,xtitle='X',ytitle='Y',color='red')
```

- Coyote Graphics

```
IDL> cgcontour,z,xtitle='X',ytitle='Y',thick=2.,color='red'
```

Surfaces



http://www.idlcoyote.com/gallery/

Surfaces

- Direct graphics

```
IDL> surface,z,xtitle='x',ytitle='y',ztitle='z'
```

- iTools

```
IDL> isurface,z,xtitle='x',ytitle='y',ztitle='z',color='blue'
```

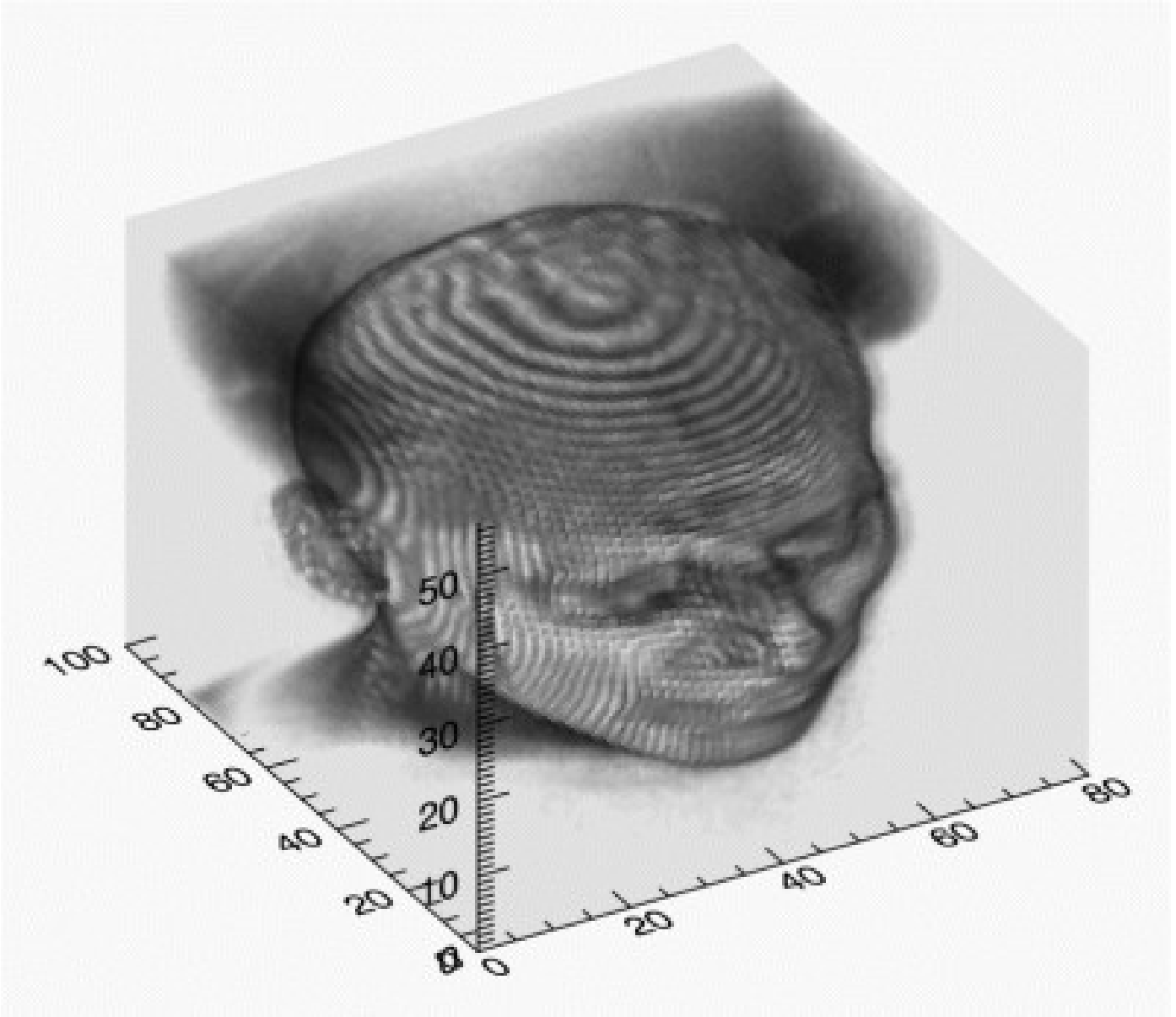
- Function Graphics

```
IDL> s=surface(z,xtitle='x',ytitle='y',ztitle='z',color='blue')
```

- Coyote Graphics

```
IDL> cgsurface,z,xtitle='x',ytitle='y',ztitle='z',color='blue'
```

Volumes



http://www.exelisvis.com/docs/Volume.html

Volumes

First, read some volume data (3D array of densities)

```
IDL> file = FILEPATH('head.dat', SUBDIRECTORY = ['examples', 'data'])
IDL> data = READ_BINARY(file, DATA_DIMS = [80, 100, 57])
```

- Object graphics

```
IDL> xvolume,data
```

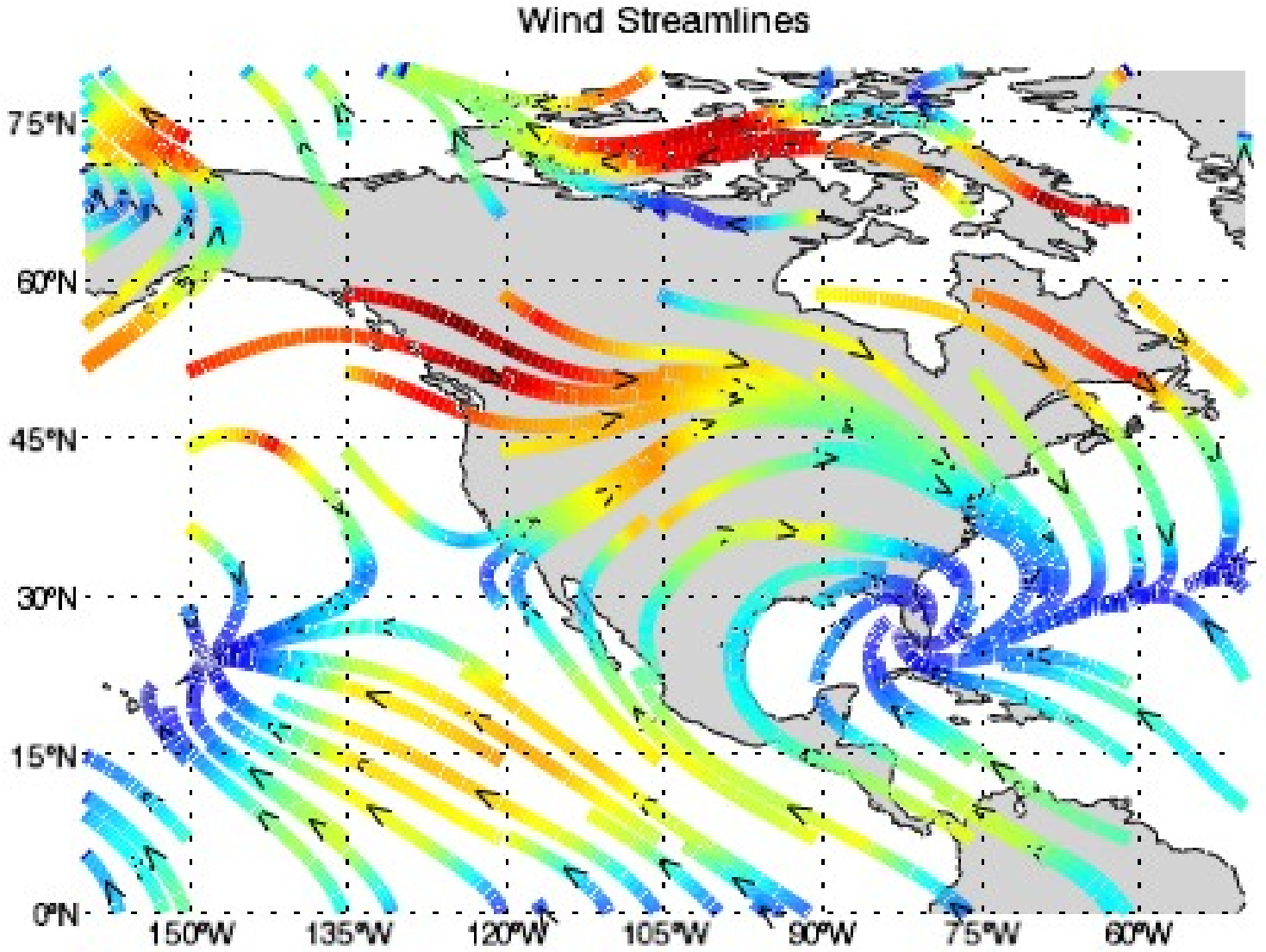
- iTools

```
IDL> ivolume,data
```

- Function Graphics

```
IDL> v=volume(data)
```

Vector fields



http://www.exelisvis.com/docs/vectors_gallery.html

Vector fields

- First, make up some data (2D wind velocities and 2D positions):

```
IDL> n = 21
IDL> x = 10*DINDGEN(n)/(n-1) - 5
IDL> y = 10*DINDGEN(n)/(n-1) - 5
IDL> xx = REBIN(x, n, n)
IDL> yy = REBIN(TRANSPOSE(y), n, n)
IDL> mu = 1
IDL> xdot = mu*(xx - xx^3/3 - yy)
IDL> ydot = xx/mu
```

- iTools

```
IDL> ivector,uu,vv,xx,yy
```

- Function Graphics

```
IDL> ivector,uu,vv,xx,yy
```

Saving graphics

- Direct Graphics
  - When you put something into a direct graphics window, it is “printed” on that window, over what was already there.
  - There is no editing, no saving from a window into something else.
  - To save bitmaps, the window contents can be obtained and saved into an image:

```
IDL> contour,dist(200)
IDL> im=tvrd()
IDL> write_image,'test.png','png',im
```

Saving Graphics

- Direct Graphics
  - To save vector graphics, everything has to be drawn, from the beginning, into a vector device (ps / eps), not the screen:

```
IDL> old_device=!d.name
IDL> print,old_device
X
IDL> set_plot,'ps'
IDL> device,filename='test.eps',/encapsulated,/color
IDL> plot,dindgen(20),color=cgcolor('blue')
IDL> device,/close
IDL> set_plot,old_device
```

Saving Graphics

- iTools
  - Besides clicking on File->Save, one can do:

```
IDL> iplot,/test
IDL> isave,'test.pdf'
```

- Function graphics
  - Besides clicking on the save button, one can do:

```
IDL> p=surface(/test)
IDL> p.save,'test.jpg',resolution=300
```

Saving Graphics

- Coyote Graphics
  - Despite being based on direct graphics, Coyote Graphics allow to edit the window's contents (erase elements, change window size), and save into files – even a Coyote format, to recreate the graphics window.
  - How? Coyote Graphics keeps track of all the commands that went into making the graphic, and can replay them to redraw things.

Saving Graphics

- Coyote Graphics

```
IDL>   cgLoadCT, 33, NColors=8, Bottom=1
IDL>   data = cgDemoData(2)
IDL>   levels = cgConLevels(data, NLEVELS=8)
IDL>   cgContour, data, Levels=levels, $
IDL>     C_Colors=Indgen(8)+1, /Fill, /Outline, $
IDL>     Position=[0.1, 0.1, 0.9, 0.825], /Window
IDL>   cgColorbar, NColors=7, Bottom=1,
OOB_High=8B, $
IDL>     /Discrete, Range=[Min(levels),
Max(levels)], /AddCmd
IDL>   cgControl, Output='filledplot.png',
IM_Width=600
```

http://www.idlcoyote.com/cg_tips/cgoutput.php

Color in Direct Graphics

- It's complicated.
- The way to specify colors depends on the device, and on the device options.
- The only way to do it without getting crazy: cgcolor, from the Coyote Library
- http://www.idlcoyote.com/idldoc/cg/cgcolor.html

```
IDL> plot,x,y,color=cgcolor('red')
```

Multiple Graphics

- Direct Graphics

```
IDL> !p.multi=[0,2,3]
```

- iTools

```
IDL> iplot,dindgen(10),view_grid=[2,3]
IDL> iplot,dindgen(10),/view_next
```

- Function Graphics

```
IDL> p=plot(dindgen(10),layout=[2,3,1])
IDL> p2=plot(dindgen(10),layout=[2,3,2],/current)
```

- Coyote Graphics

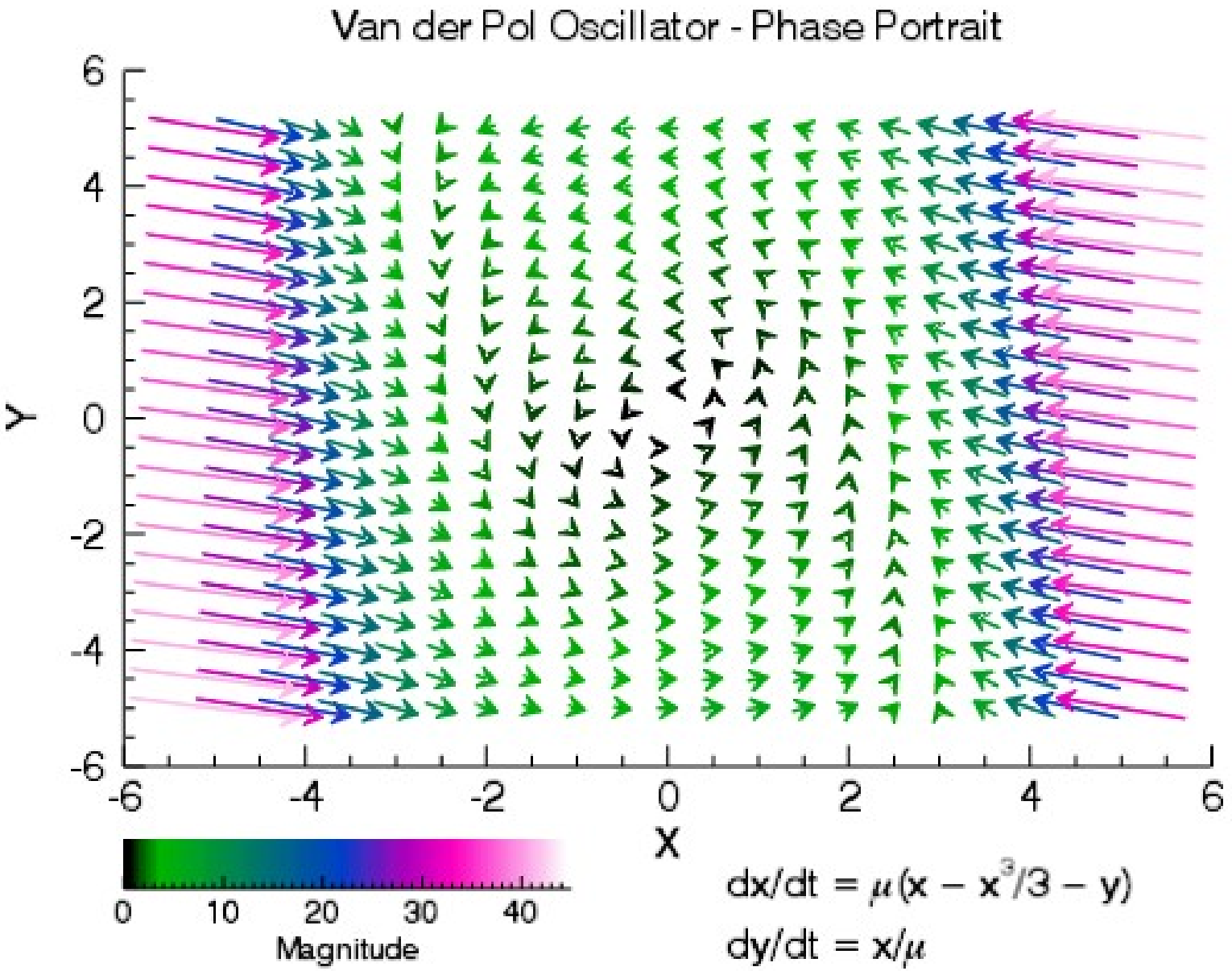
```
IDL> !p.multi=[0,2,3]
```

Editing function graphics

```
n = 21
x = 10*DINDGEN(n)/(n-1) - 5
y = 10*DINDGEN(n)/(n-1) - 5
xx = REBIN(x, n, n)
yy = REBIN(TRANSPOSE(y), n, n)
mu = 1
xdot = mu*(xx - xx^3/3 - yy)
ydot = xx/mu

v = VECTOR(xdot, ydot, x, y, $
AUTO_COLOR=1, AUTO_RANGE=[0,45], $
RGB_TABLE=10, $
POSITION=[0.10,0.22,0.95,0.9], $
XTITLE='X', YTITLE='Y', $
TITLE='Van der Pol Oscillator - Phase Portrait')

; Change some properties.
v.ARROW_THICK = 2
v.LENGTH_SCALE = 2

; Add a colorbar and text annotations.
c = COLORBAR(TARGET=v, $
POSITION=[0.10,0.1,0.45,0.15], $
TITLE='Magnitude')
t = TEXT(0.57, 0.09, 'dx/dt = $\mu(x - x^3/3 - y)$')
t = TEXT(0.57, 0.03, 'dy/dt = $x/\mu$')
```

http://www.exelisvis.com/docs/VECTOR.html

Editing function graphics



http://www.exelisvis.com/docs/VECTOR.html

Overplotting

- Direct Graphics

```
IDL> plot,x,y
IDL> oplot,x,y2
```

- iTools

```
IDL> iplot,x,y,name='y',/insert_legend
IDL> iplot,x,y2,/over,color='red',name='y2',/insert_legend
```

- Function Graphics

```
IDL> p=plot(x,y,name='y')
IDL> p2=plot(x,y2,name='y2',color='red',/over)
IDL> l=legend(target=[p,p2])
```

- Coyote Graphics

```
IDL> cgplot,x,y
IDL> cgoplot,x,y2,color='red'
IDL> cglegend,colors=['black','red'],titles=['y','y2']
```

Annotations

- Direct Graphics

```
IDL> plot,x,y
IDL> xyouts,4,0.1,'my label',/data
```

- iTools

```
IDL> iplot,x,y,name='y',/insert_legend
IDL> itext,'my label',4,0.1,/data
```

- Function Graphics

```
IDL> p=plot(x,y,name='y')
IDL> t=text(4,0.1,'my label',/data)
```

- Coyote Graphics

```
IDL> cgplot,x,y
IDL> cgtext,4,0.1,'my label',/data
```

Without the ***/data***, the coordinates would have been normal coordinates – 0 to 1.

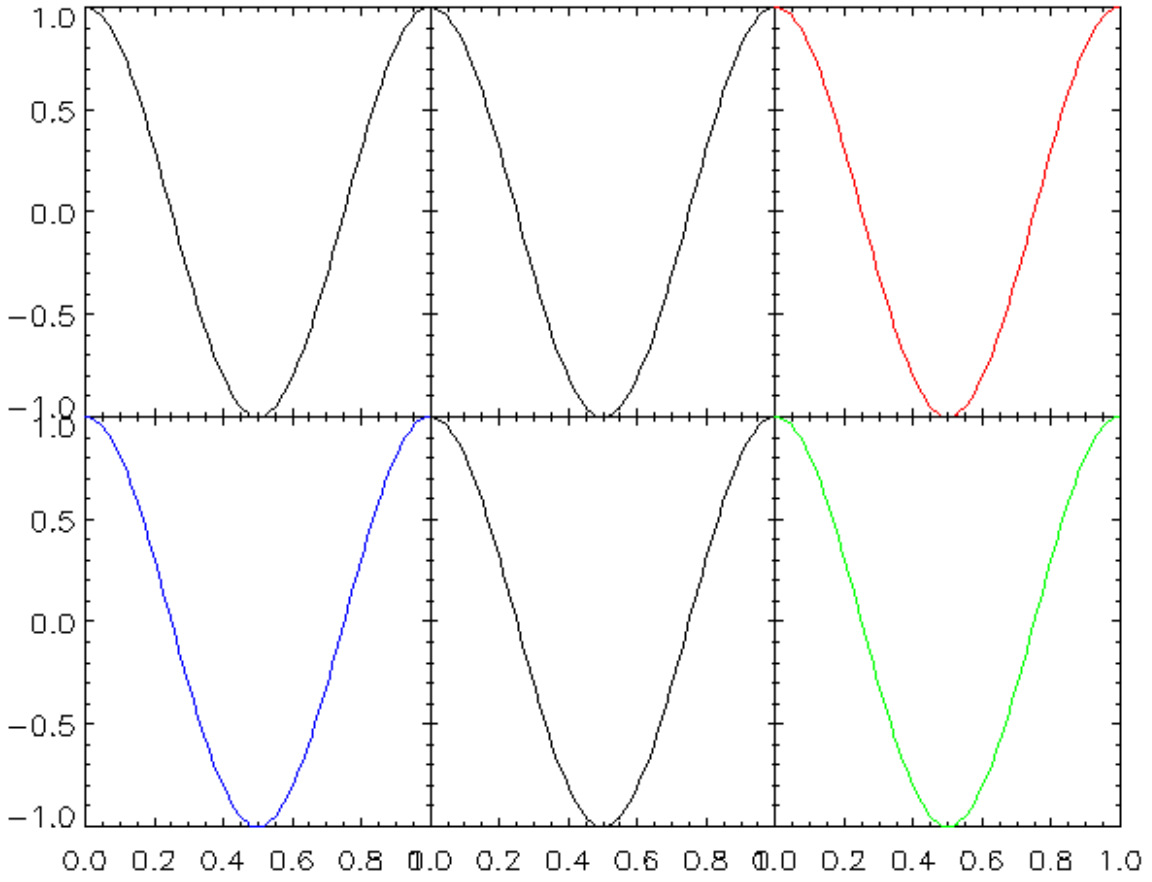
Shared axes

- Direct Graphics / Coyote Graphics (with multiplot from IDL Astro, http://idlastro.gsfc.nasa.gov/contents.html)

```
IDL> multiplot,[3,2]
IDL> plot,x,y
IDL> cgplot,x,y
IDL> multiplot
IDL> cgplot,x,y
IDL> multiplot
IDL> cgplot,x,y,color=cgcolor('red')
IDL> multiplot
IDL> cgplot,x,y,color=cgcolor('blue')
IDL> multiplot
IDL> cgplot,x,y,color=cgcolor('green')
```

Shared axes

- Direct Graphics / Coyote Graphics (with multiplot from IDL Astro, http://idlastro.gsfc.nasa.gov/contents.html)



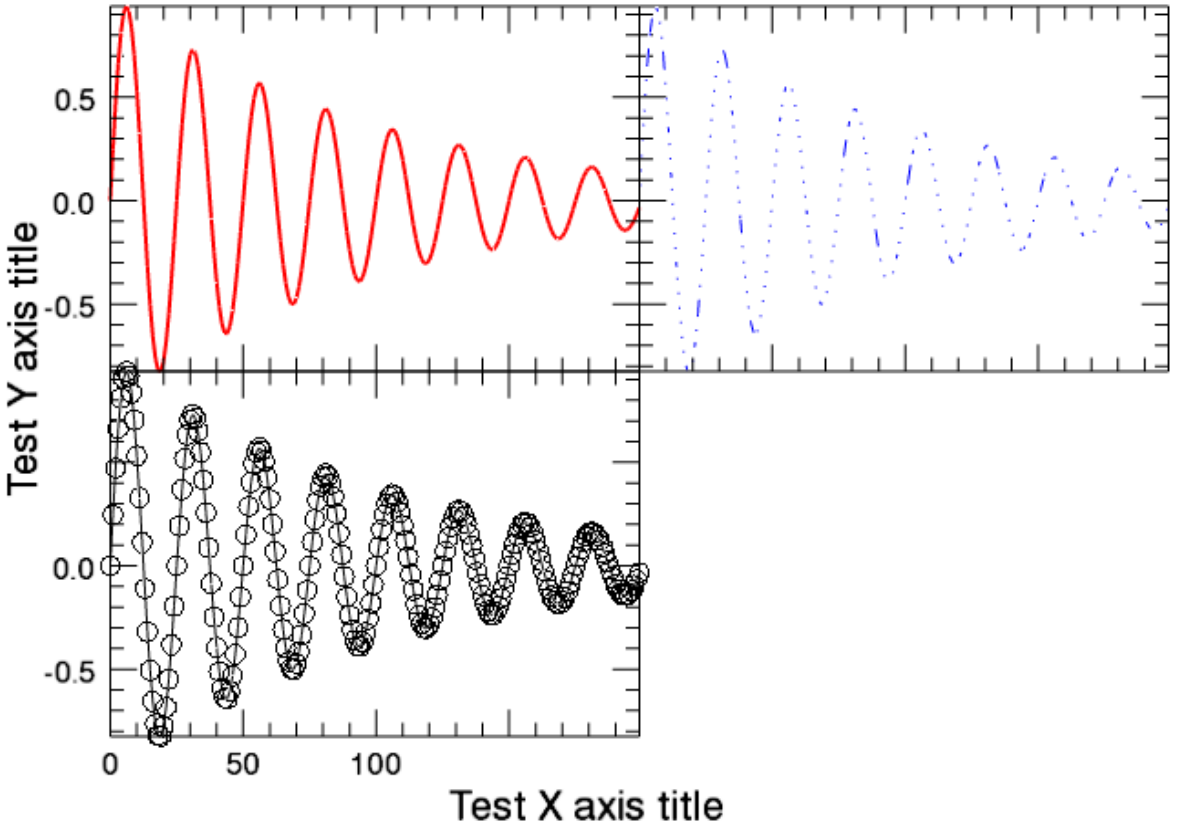
Shared axes

- Function Graphics (with pp_multiplot from pp_lib, http://ppenteado.net/idl/pp_lib/doc/index.html)

```
m=pp_multiplot(multi_layout=[2,2],global_xtitle='Test X$
axis title',global_ytitle='Test Y axis title')

;Since multi_index was not provided, this will occupy the
first free location in the grid:
p0=m.plot(/test,color='red',thick=2.)

;Second location, since multi_index was omitted
p1=m.plot(/test,color='blue',linestyle=1)

p2=m.plot(/test,multi_index=2,symbol='circle')
;Third location, explicitly set with multi_index
```

Shared axes

- Function Graphics (with pp_lib, http://ppenteado.net/idl/pp_lib/doc/index.html)



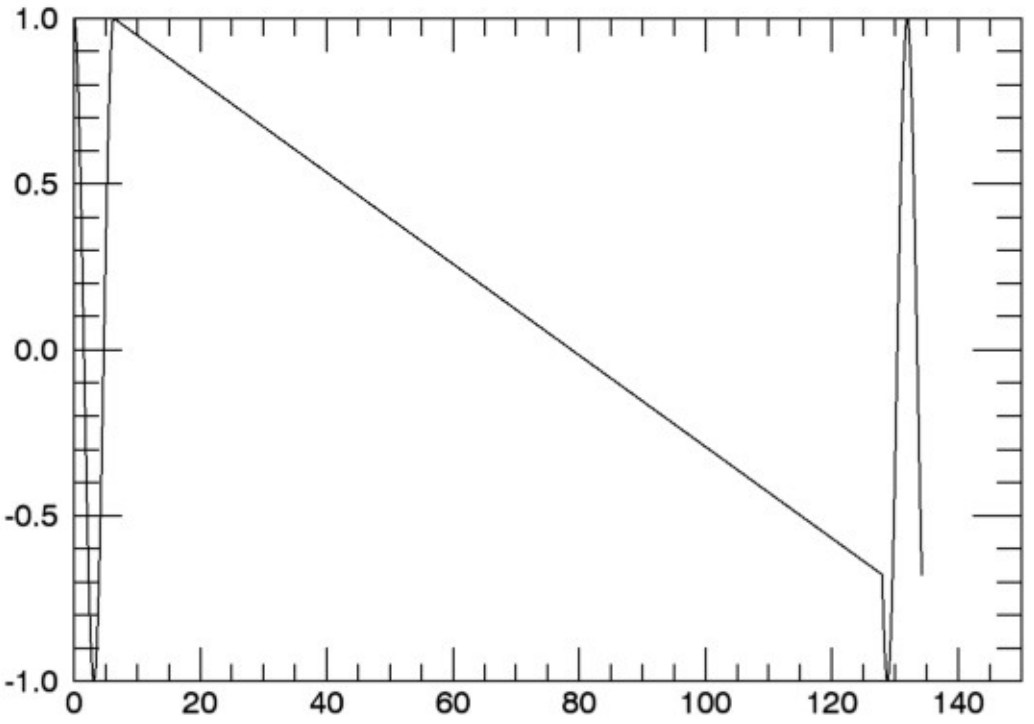
- Ranges can be edited afterwards, and all plots stay synchronized with range changes.

Segmented axes

- Function Graphics (with pp_lib, http://ppenteado.net/idl/pp_lib/doc/index.html)

Problem: Data with big gaps in the x axis

```
IDL> x=dindgen(100)*2d0*!
dpi/99d0
IDL> x2=x+127.98
IDL> iplot,[x,x2],cos([x,x2])
```



Segmented axes

- Function Graphics (with pp_lib, http://ppenteado.net/idl/pp_lib/doc/index.html)

Solution: multiple plots with shared axes and little blanks between them

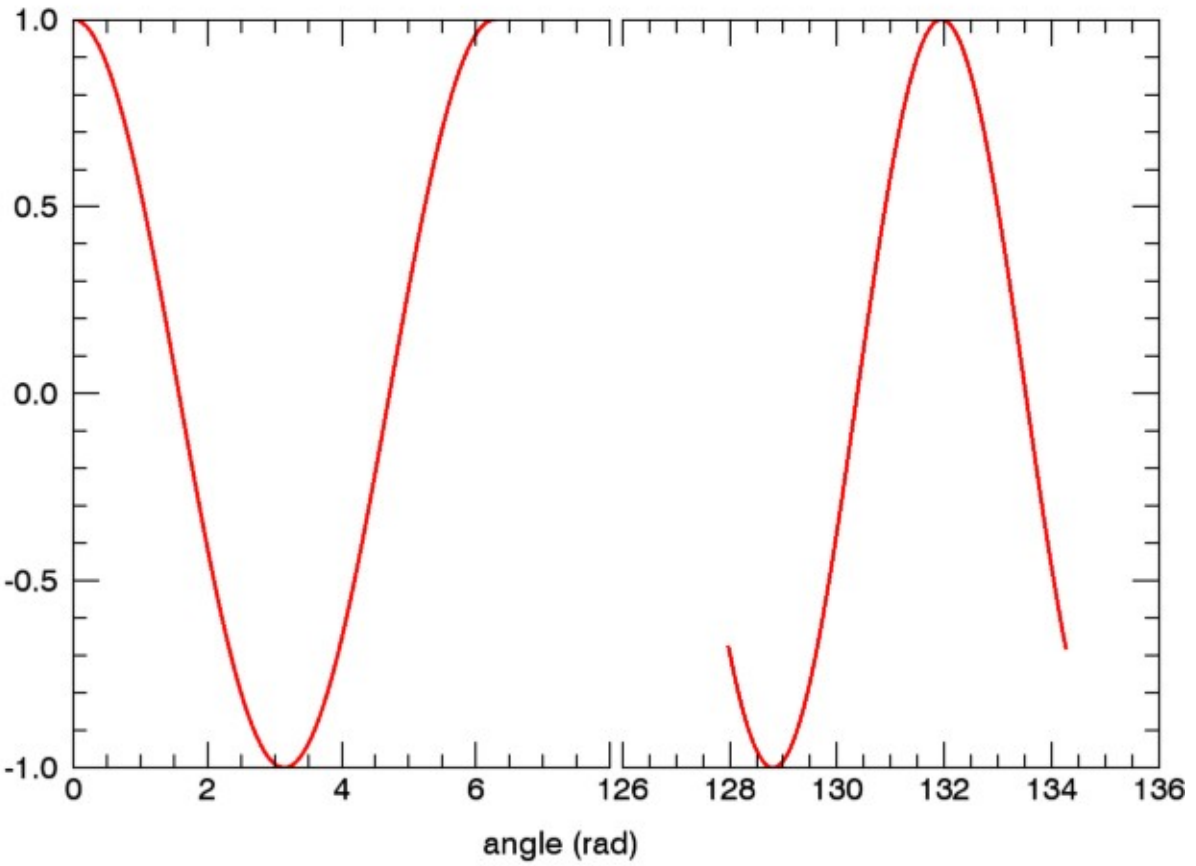
```
IDL> m=pp_multiplot(multi_layout=[2,1],$
xgap=0.01,global_xtitle='angle (rad)')
IDL> p0=m.plot(x,cos(x),color='red',thick=2.)
IDL> p1=m.plot(x2,cos(x2),color='red',thick=2.)
IDL> p1['axis1'].hide=1
IDL> p0['axis3'].hide=1
```

Segmented axes

- Function Graphics (with pp_lib, http://ppenteado.net/idl/pp_lib/doc/index.html)

Solution: multiple plots with shared axes and little blanks between them

```
IDL> m=pp_multiplot(multi_
xgap=0.01,global_xtitle='a
IDL> p0=m.plot(x,cos(x),co
IDL> p1=m.plot(x2,cos(x2),
IDL> p1['axis1'].hide=1
IDL> p0['axis3'].hide=1
```



Some references

- http://www.idlcoyote.com/gallery/
- http://www.idlcoyote.com/graphics_tips/coyote_graphics.php
- http://www.idlcoyote.com/books/index.php
- http://modernidl.idldev.com/
- https://groups.google.com/forum/#!forum/comp.lang.idl-pvwave
- http://www.exelisvis.com/docs/routines-1.html
- http://www.exelisvis.com/docs/visualize.html