Data Analysis in Geophysics
ESCI 7205

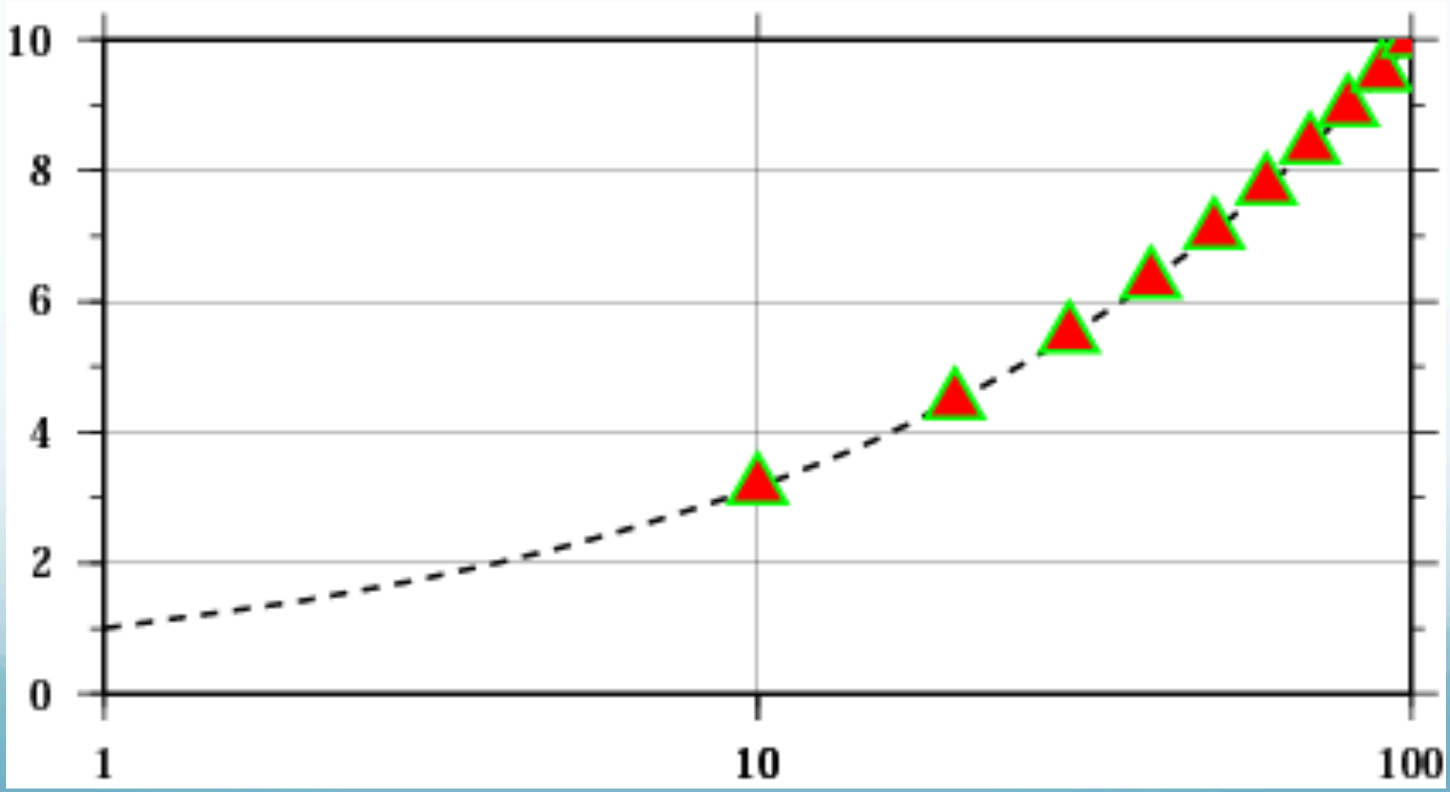
Bob Smalley
Room 103 in 3892 (long building), x-4929

Tu/Th - 13:00-14:30
CERI MAC (or STUDENT) LAB

Lab – 14, 12/8/13

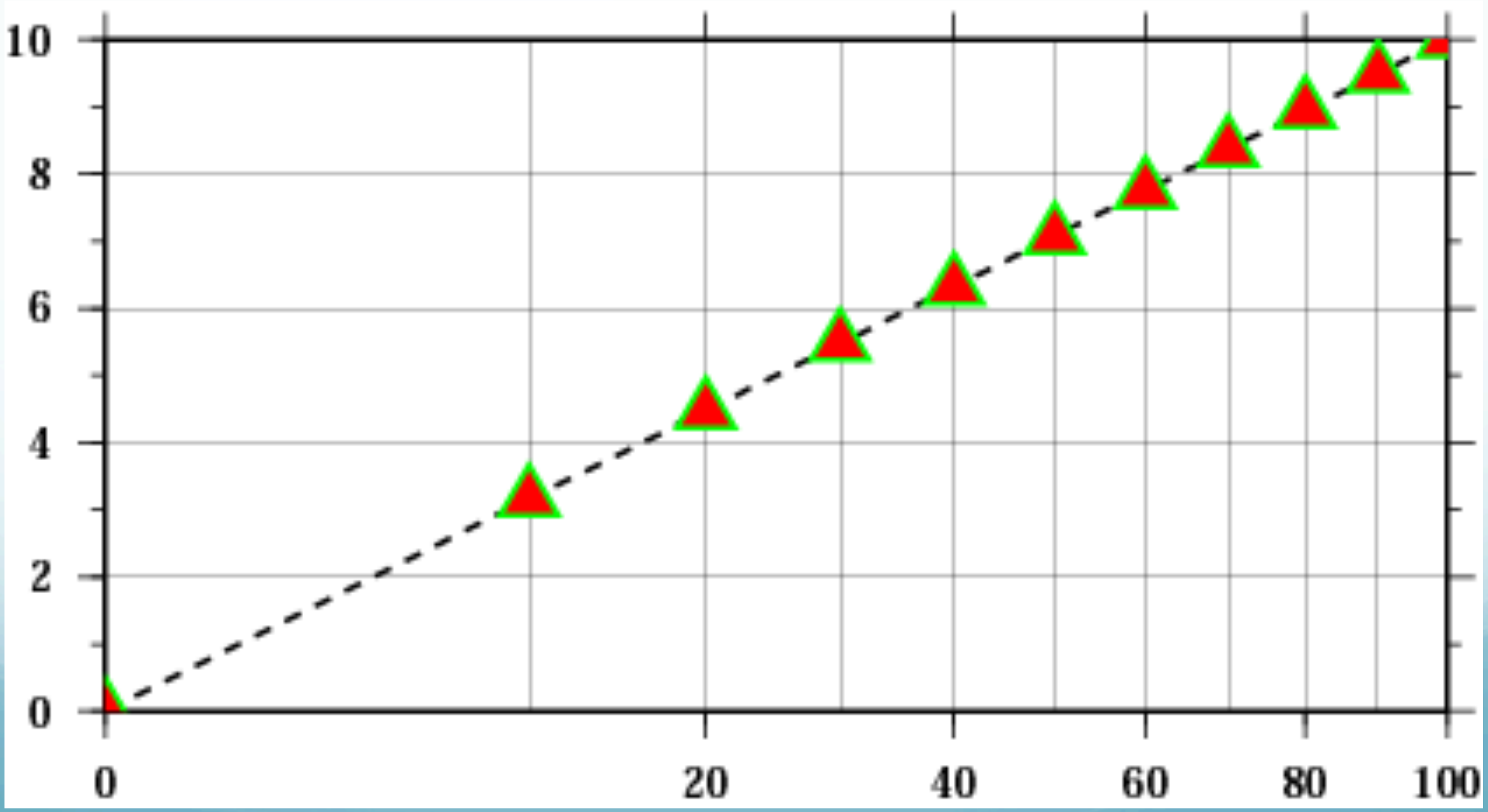
There are two other non-map-projected forms

1) Logarithmic - add `l` (lower case letter L) after scale of axis you want logarithmic `-JX4l/2`



2) Power/exponential – add p and the value of the exponent to scale of axis you want exponential (can scale axes individually)

`-JX4p0.5/2`



Common command options on first, and possibly subsequent, calls

Need on all calls

`-R` Define region for plot – will need on first call and at least "`–R`" on subsequent

`-J` define projection for plot – will need this on all calls if need to define region

Common command options on first, and possibly subsequent, calls

(Generally) Need on first call only

`-B` Borders -- annotation, frame, grid. Only need on first (or a single) call.

`-P` Switch between landscape and portrait modes

`-X` Shift X axis

`-Y` Shift Y axis

Common command options on first, and possibly subsequent, calls.

Need when needed.

`-K` Don't close PostScript (`showpage`), use when more plotting will follow
- need on all but last GMT call

`-O` Don't initialize PostScript, use when appending to pre-existing file
- need on all but first GMT call

- use both `–K` and `–O` when putting a large number of GMT call outputs together

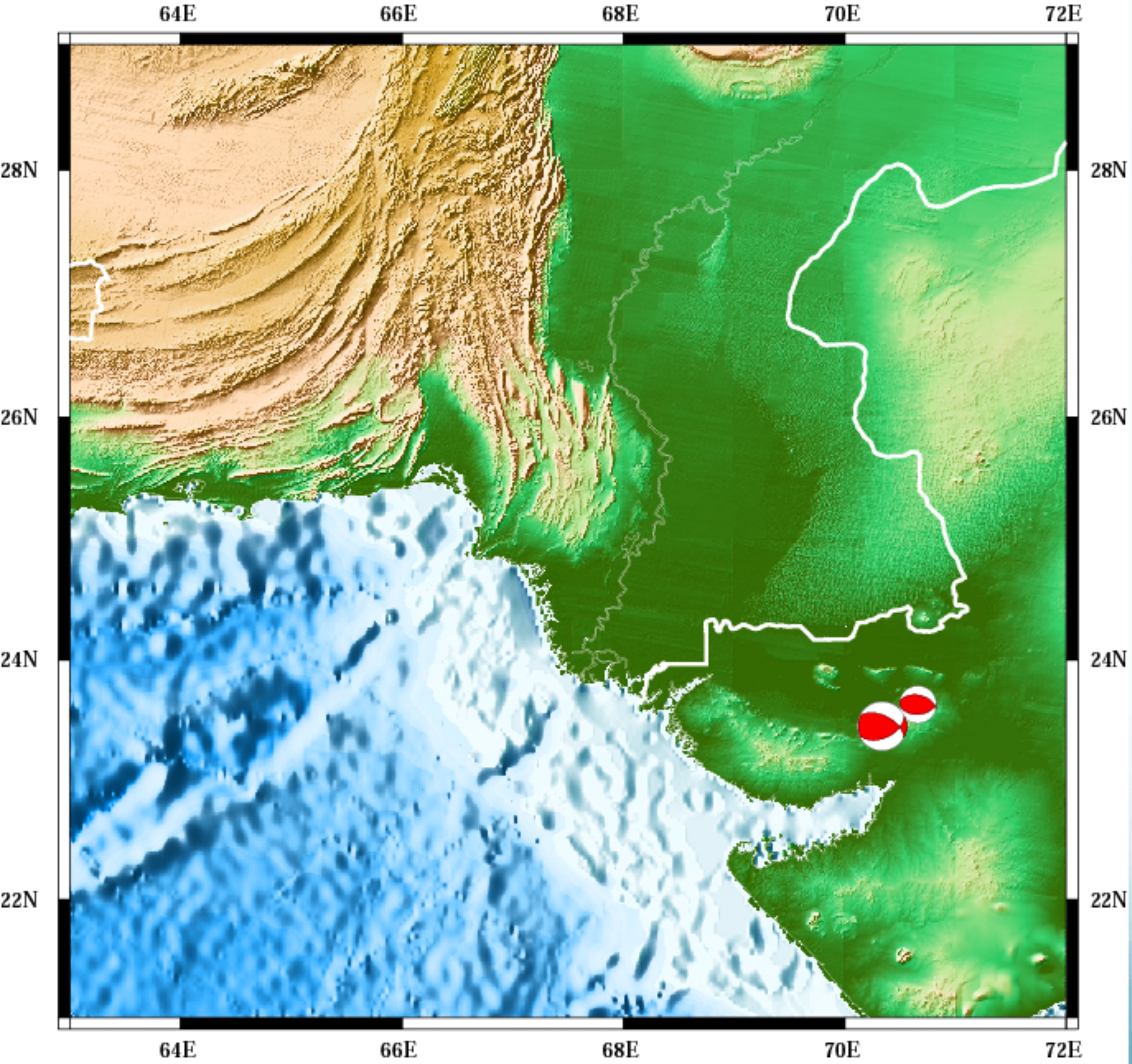
Common command options on first, and possibly subsequent, calls.

Need when needed.

`-V` Verbose (prints out stuff to standard error for user).

`-H` Header records (tells GMT to skip first `H` lines of ascii input file)

How about making pretty MAPS?



Map projections available in GMT

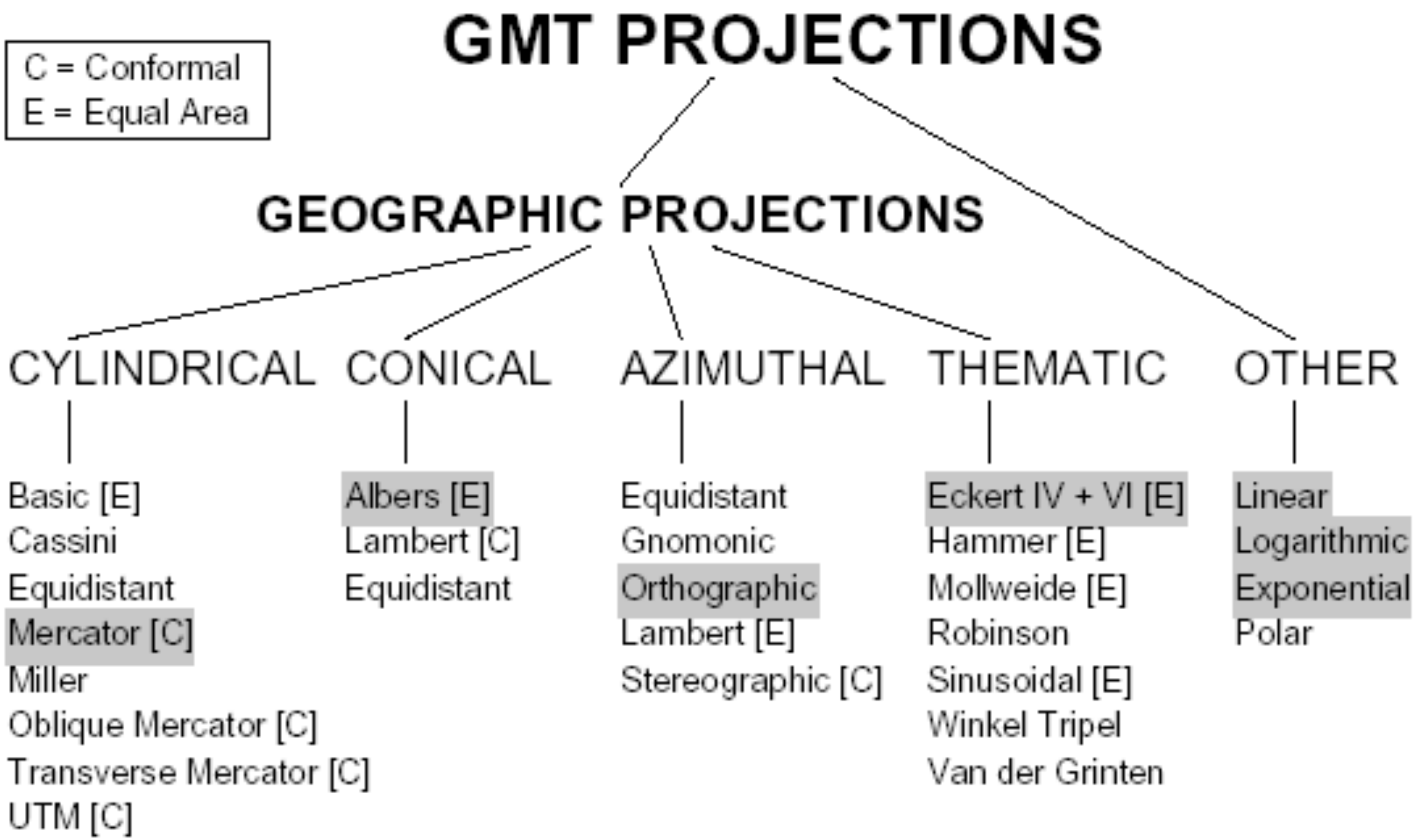


Figure 1.9: The 25 projections available in **GMT**.

List of "standard" command line options.
The –J option sets the "projection"
One has to look at the man page for each one as "different things vary"

STANDARDIZED COMMAND LINE OPTIONS	
–B*xinfo*[/*yinfo*[/*zinfo*]][*WESNZwesnz+*][:.*title*:] Tickmarks. Each *info* is [**a**]*tick*[**m**|**c**][**f***tick*[**m**|**c**]][**g***tick*[**m**|**c**]][**l**|**p**][:"label":][:,"unit":]	
–H[*n_headers*]	ASCII tables have header record[*s*]
–J (upper case for width, lower case for scale)	Map projection (see below)
–**JA**lon_0/lat_0/*width*	Lambert azimuthal equal area
–**JB**lon_0/lat_0/lat_1/lat_2/*width*	Albers conic equal area
–**JC**lon_0/lat_0/*width*	Cassini cylindrical
–**JD**lon_0/lat_0/lat_1/lat_2/*width*	Equidistant conic
–**JE**lon_0/lat_0/*width*	Azimuthal equidistant
–**JF**lon_0/lat_0/*horizon*/*width*	Azimuthal Gnomonic
–**JG**lon_0/lat_0/*width*	Azimuthal orthographic
–**JH**lon_0/*width*	Hammer equal area
–**JI**lon_0/*width*	Sinusoidal equal area
–**JJ**lon_0/*width*	Miller cylindrical
–**JKf**lon_0/*width*	Eckert IV equal area
–**JKs**lon_0/*width*	Eckert VI equal area
–**JL**lon_0/lat_0/lat_1/lat_2/*width*	Lambert conic conformal
–**JM***width* or –**JM**lon_0/lat_0/*width*	Mercator cylindrical
–**JN**lon_0/*width*	Robinson
–**JOa**lon_0/lat_0/*az*/*width*	Oblique Mercator, 1: origin and azimuth
–**JOb**lon_0/lat_0/lon_1/lat_1/*width*	Oblique Mercator, 2: two points
–**JOc**lon_0/lat_0/lon_p/lat_p/*width*	Oblique Mercator, 3: origin and pole
–**JP**[**a**]*width*[/*origin*]	Polar [azimuthal] (θ, r) (or cylindrical)
–**JQ**lon_0/*width*	Equidistant cylindrical (Plate Carrée)
–**JR**lon_0/*width*	Winkel Tripel
–**JS**lon_0/lat_0/*width*	General stereographic
–**JT**lon_0/*width*	Transverse Mercator
–**JU***zone*/*width*	Universal Transverse Mercator (UTM)
–**JV**lon_0/*width*	Van der Grinten
–**JW**lon_0/*width*	Mollweide
–**JX***width*[**l**|**p**][/*height*[**l**|**p**]][**d**]	Linear, $\log_{10}$, and x^a–y^b (exponential)
–**JY**lon_0/lat_s/*width*	General cylindrical equal area
–K	Append more PS later

```
pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 > map1.ps
```

pscosat makes the basic “background map”.

It “knows” about coastlines and is used to plot them.

(unfortunately GMT is particularly dumb about topography and – following the UNIX philosophy – leaves the finding and installation of topographic data to the user.)

```
pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 > map1.ps
```

"All" gmt programs plot "maps" through the projection command line option or switch **-J** (even the x-y plot).

```
pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 > map1.ps
```

All projections give you two selections for specifying the scale

(note GMT takes the mapmakers attitude that a map has to have a predetermined/known scale – assuming you want the map to nicely fill the page does not cut it – a map without an explicitly known or specified scale is simply <u>inconceivable</u>.)

pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 > map1.ps

-Jmparameters

(Mercator).

Specify one of:

-Jmscale or -JMwidth

Give scale along equator

(1:xxxx or UNIT/degree, indicated by lower case m or upper case M.

```
pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 > map1.ps
```

`-Jmlon0/lat0/scale` or

`-JMlon0/lat0/width`

Can also give central meridian, standard latitude and scale along parallel

(`1:xxxx` or `UNIT/degree`, UNIT = number inches or cms).

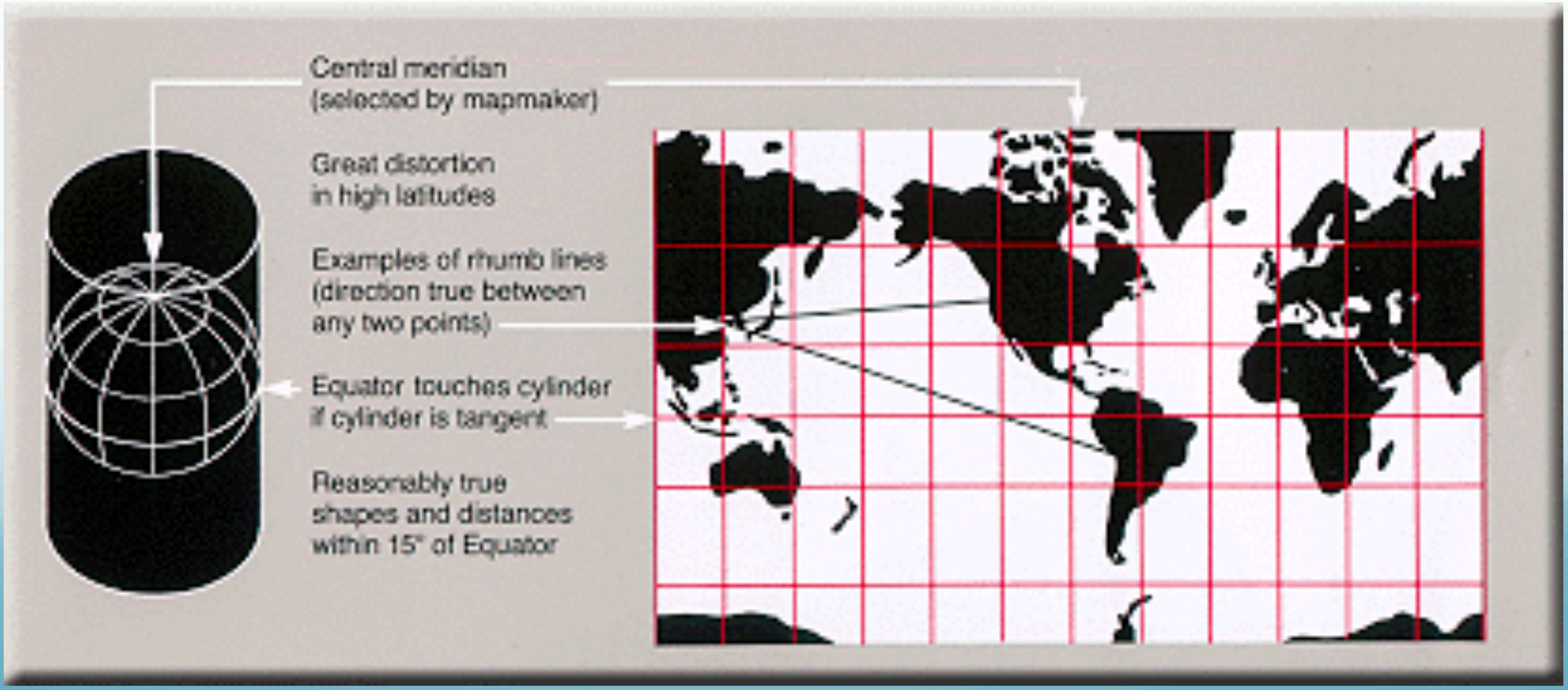
Map Projection:
addresses plotting sphere on a plane

Mercator Projection:

One way to address plotting sphere on a plane (which is whole 'nother subject)

Conformal (maintains shapes but not relative sizes)

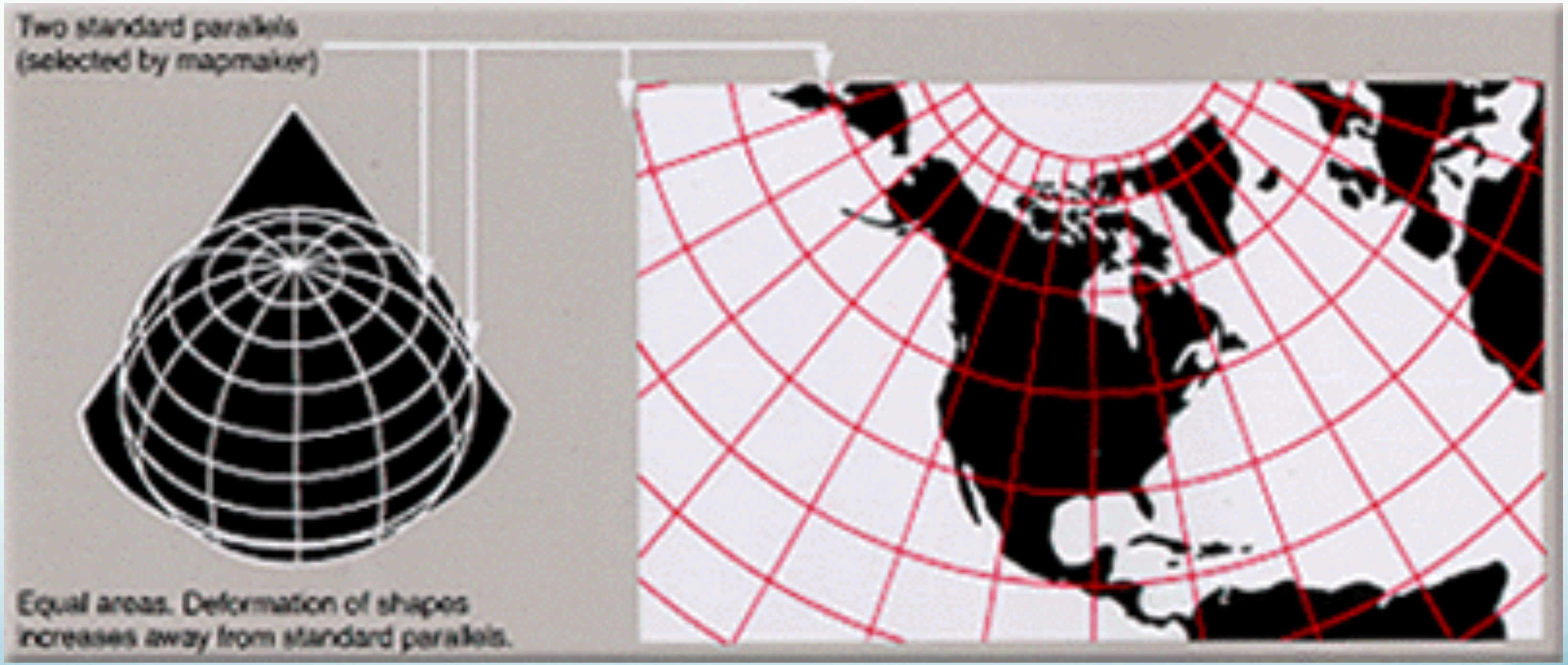
Is a cylindrical projection



Albers

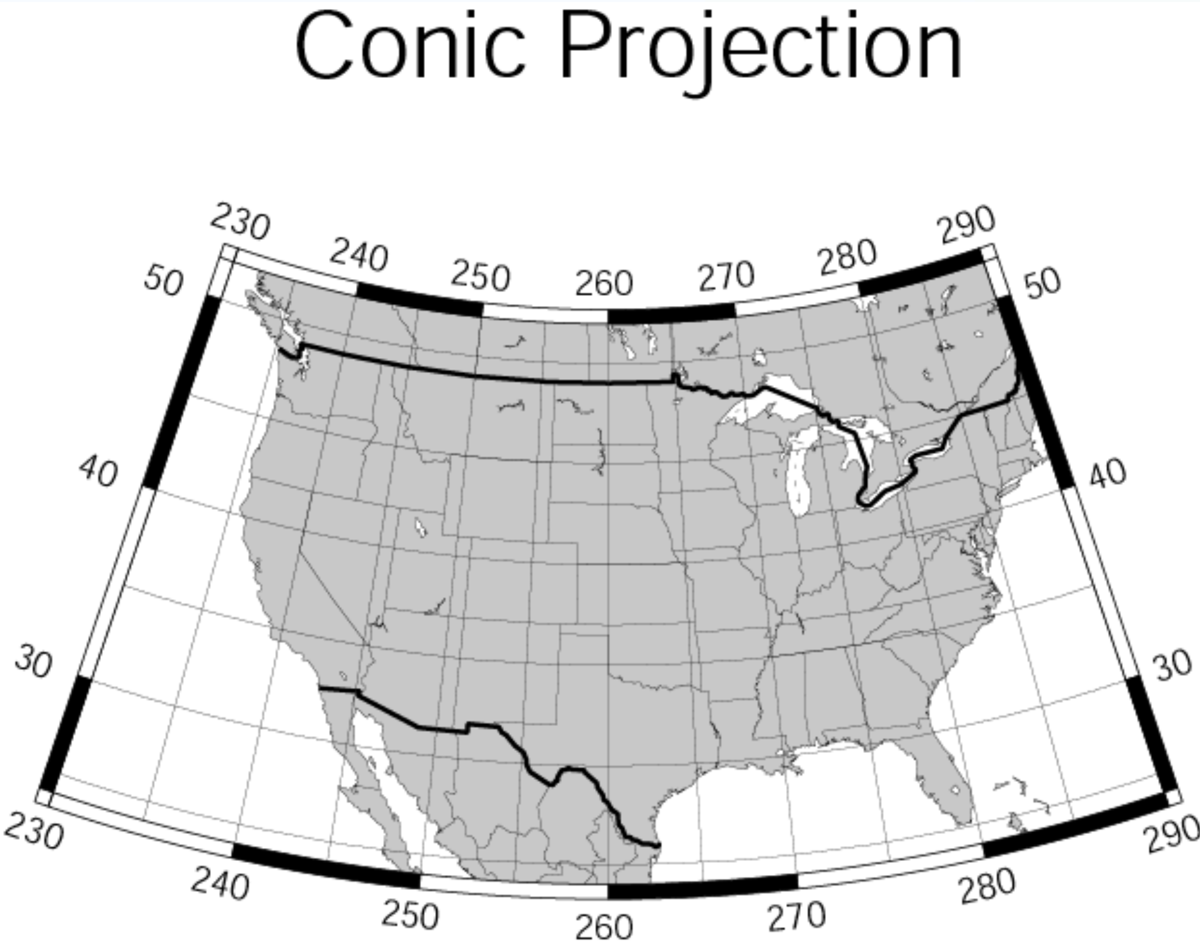
Also conformal (maintains/conserves shape)

Conical projection



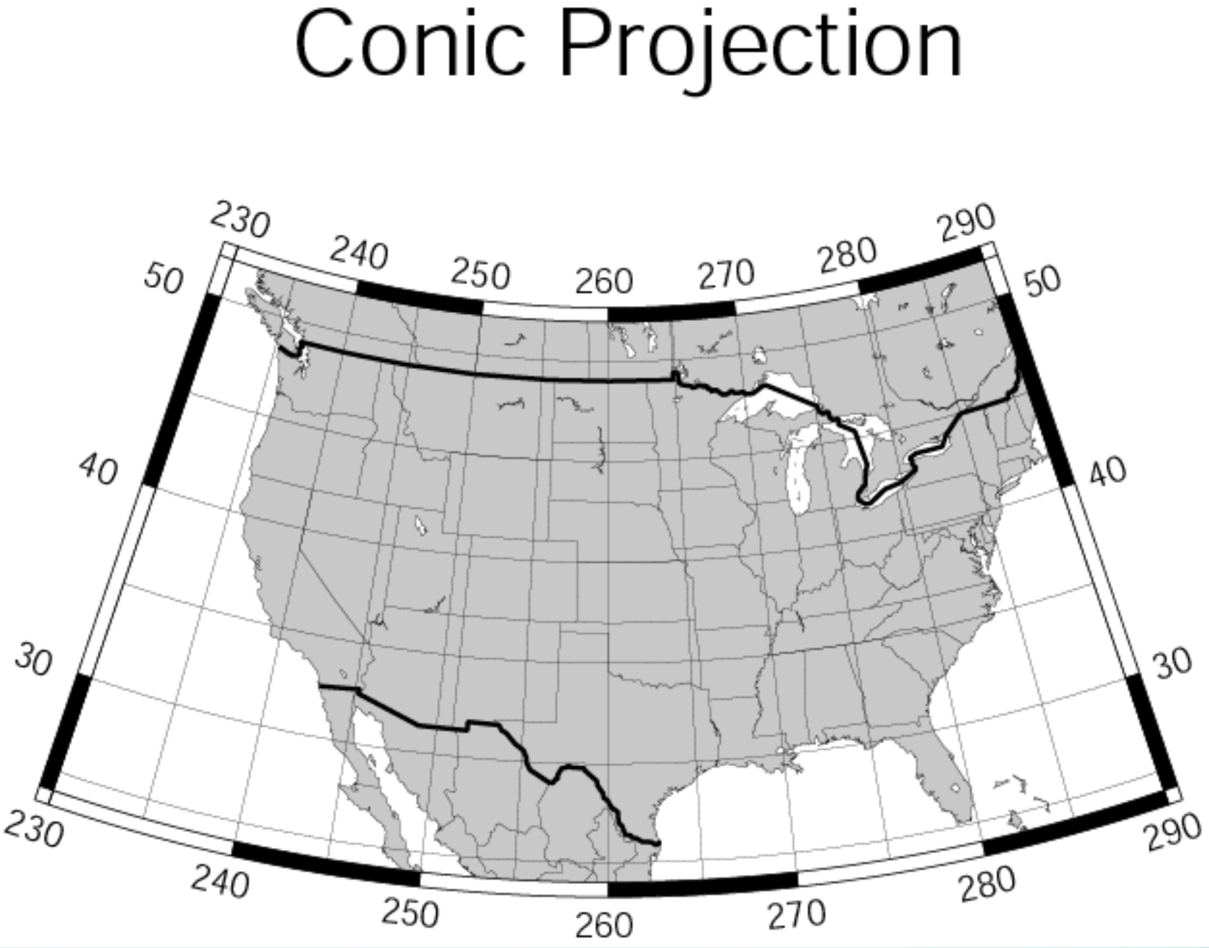
```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

Region, saw before, specified by -R, is a "rectangle" defined by latitude and longitude lines on the spherical earth.



```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

Albers Conic Projection (b/B) – need to know something (center and/or standard parallels).



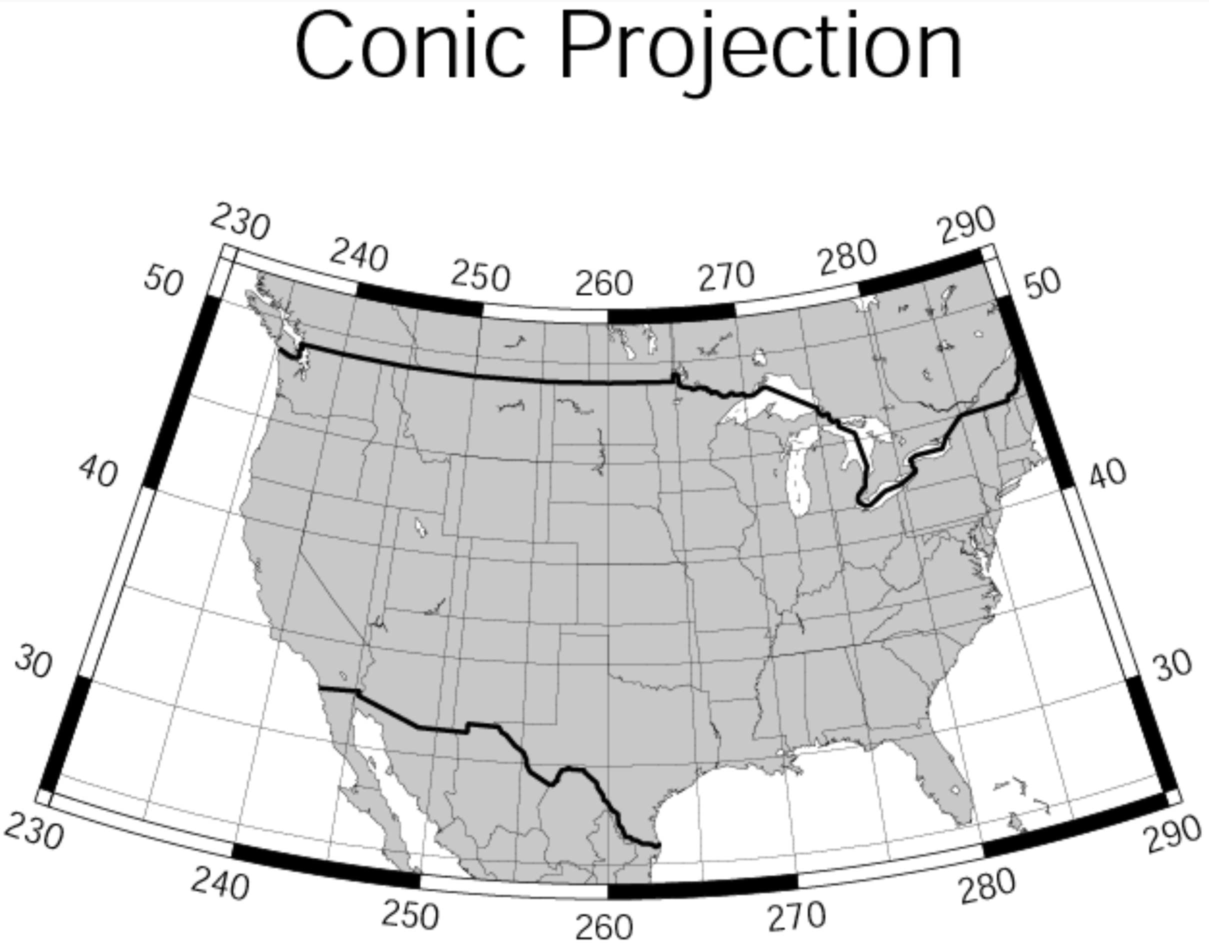
`-Jblon0/lat0/lat1/lat2/scale` or `-JBlon0/lat0/lat1/lat2/width`

Give projection center: lon0/lat0, two standard parallels: lat1/lat2, and scale (1:xxxx or UNIT/degree).

```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

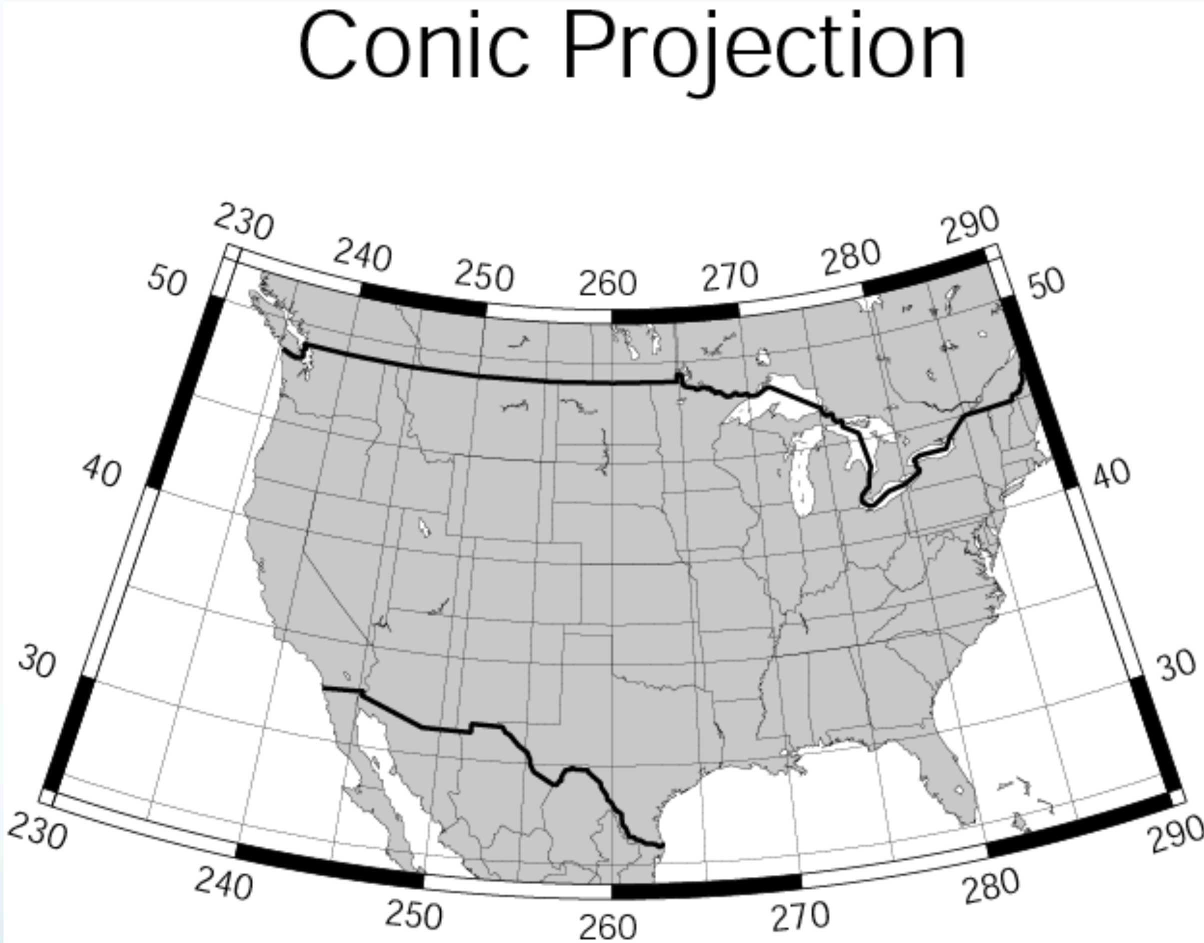
-N for political boundaries

(international, US/Canadian/ Mexican state boundaries "built in"), rivers.



```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

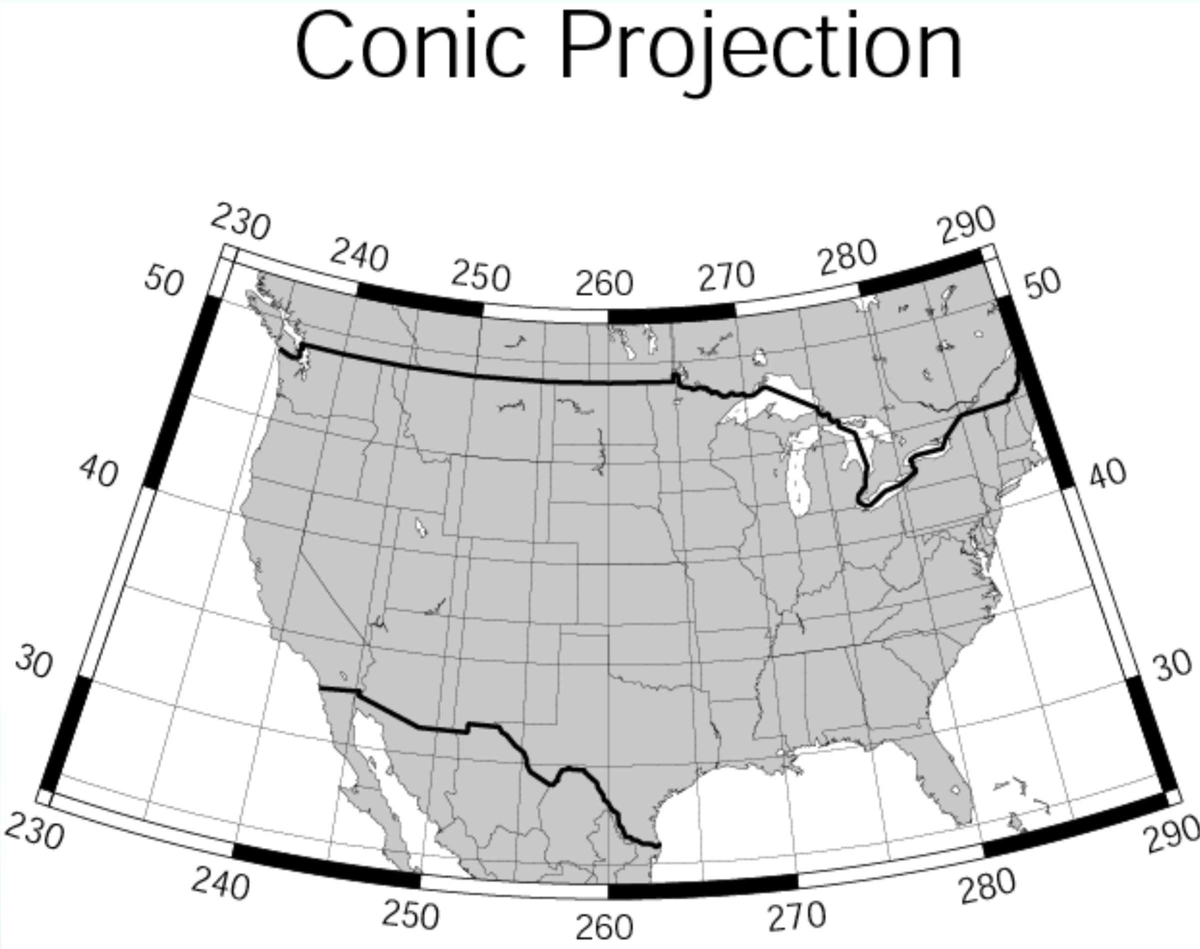
-A to get rid of small water/island features (number gives min size to plot in km^2)



```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

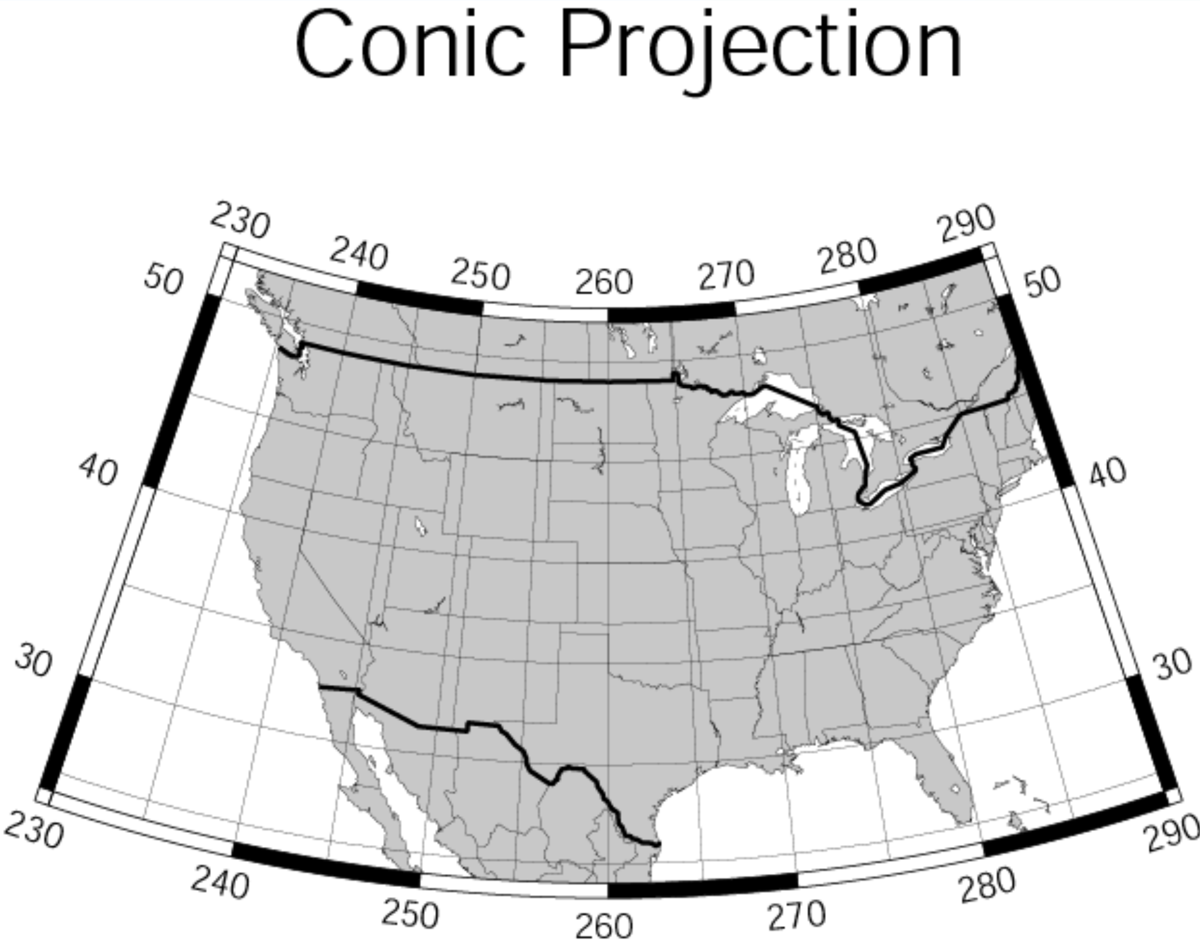
-G fill for land/dry areas, (saw before with RGB, new form with single value for gray scale - 0 black, 255 white).

-G fill for ocean/lakes/wet areas (not used here).



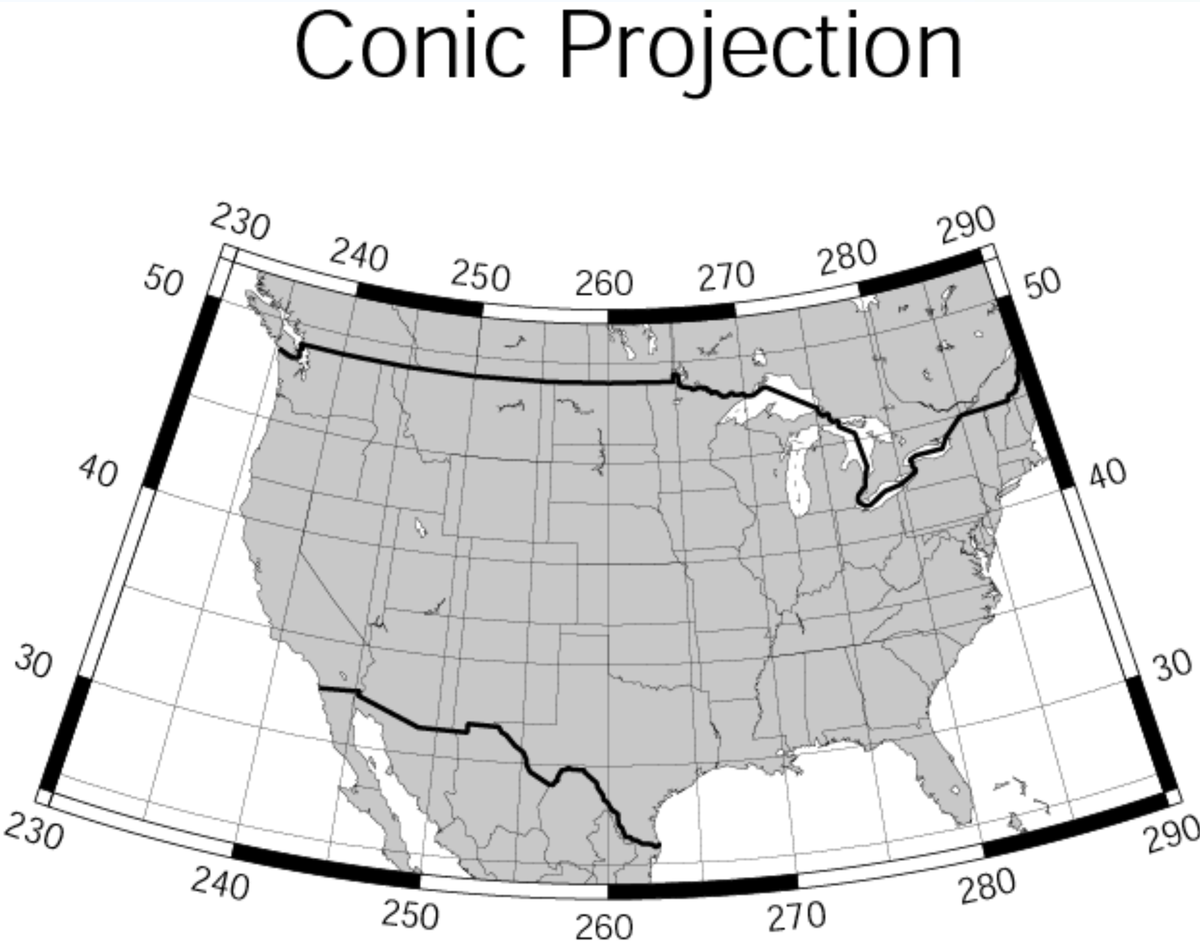
```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

-W for line widths (in points "p").



```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

-P switches from portrait to landscape or vice-versa depending on the default setting.

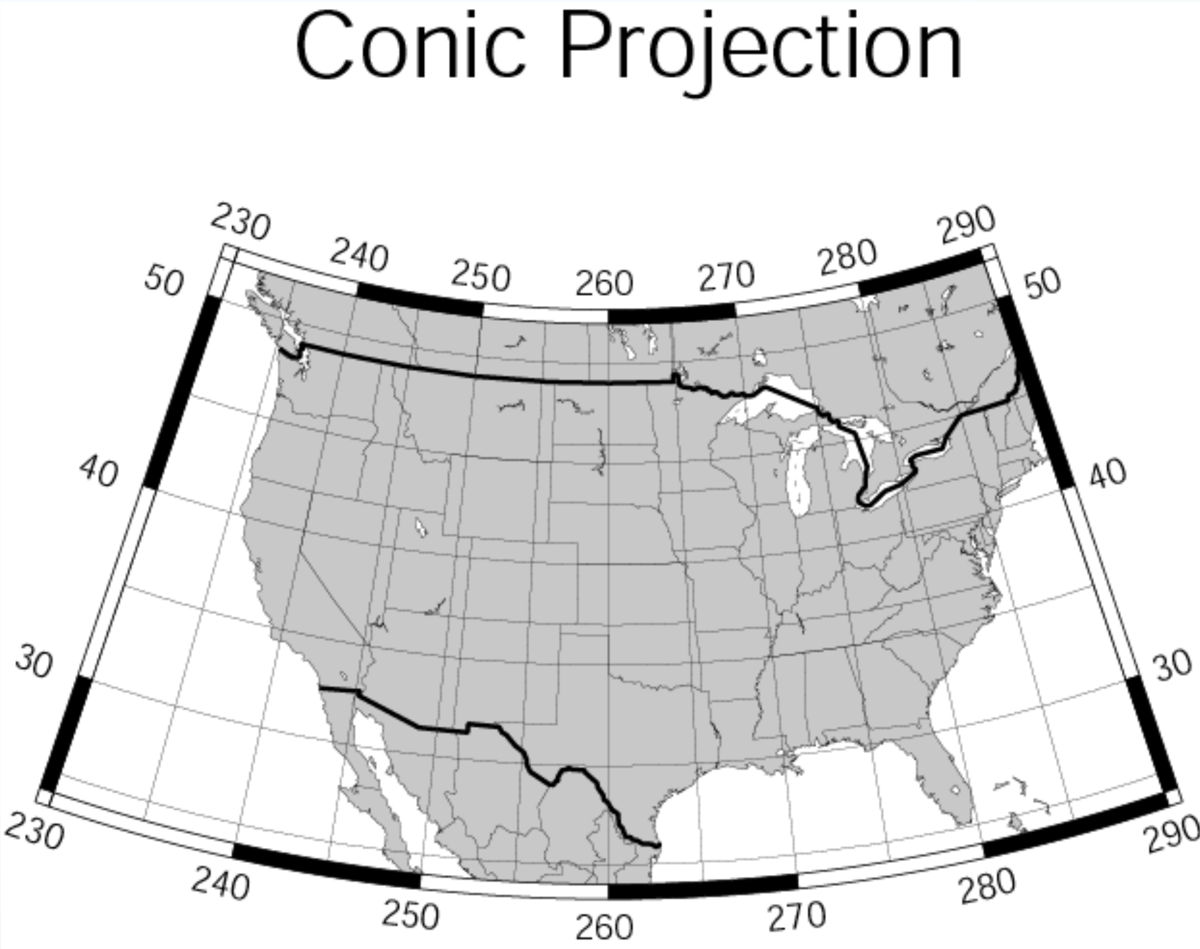


```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic\ Projection":
-N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

Put in file **map.ps** and clobber old version if it exists

(csh and tcsh, not sh or bash)

This is a single line gmt program – quite unusual.



```
pscoast -R0/360/-90/90 -JG280/30/6i -Bg30/g15 -Dc -A5000 \
-G255/255/255 -S150/50/150 -P >! map.ps
```

Other projections –

azimuthal orthographic (projection mimics looking at earth from infinite distance).

```
pscoast -R0/360/-90/90 -JG280/30/6i -Bg30/g15 -Dc -A5000 \
-G255/255/255 -S150/50/150 -P >! map.ps
```

New option

-Dc

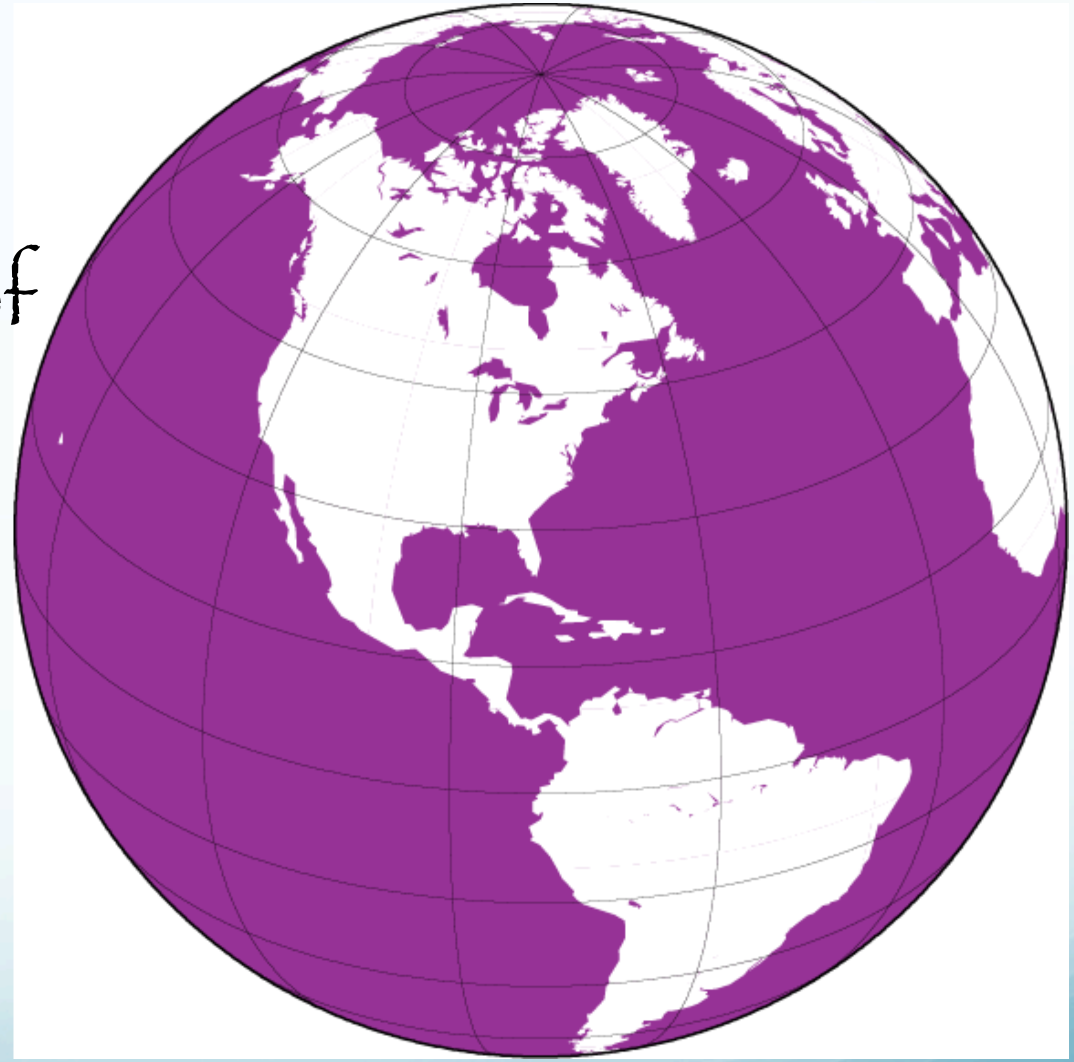
Controls resolution of coastline

`f` full

`h` high

`l` low

`c` crude

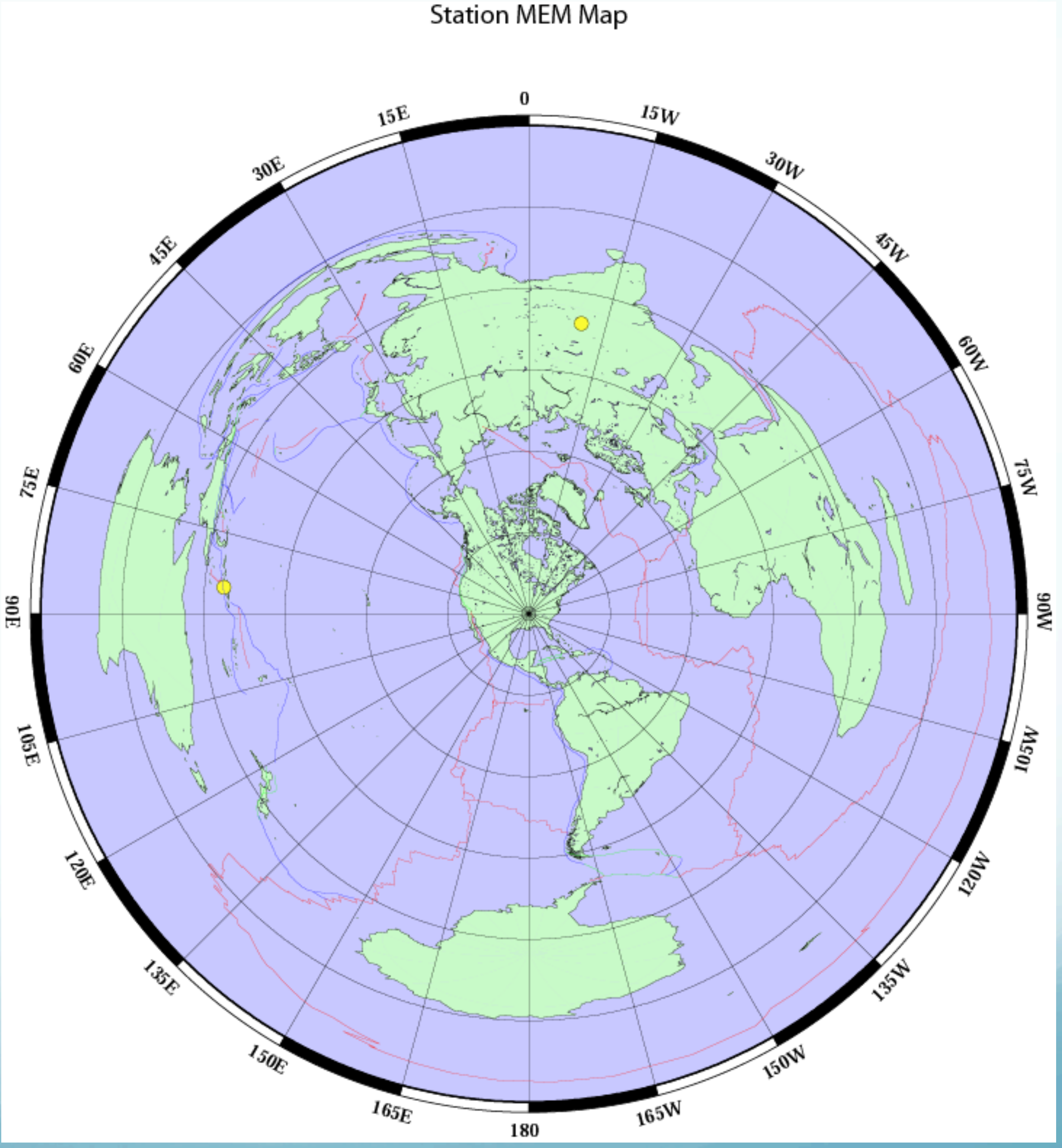


Helps manage file sizes.

Some useful maps.

The world centered on Memphis.

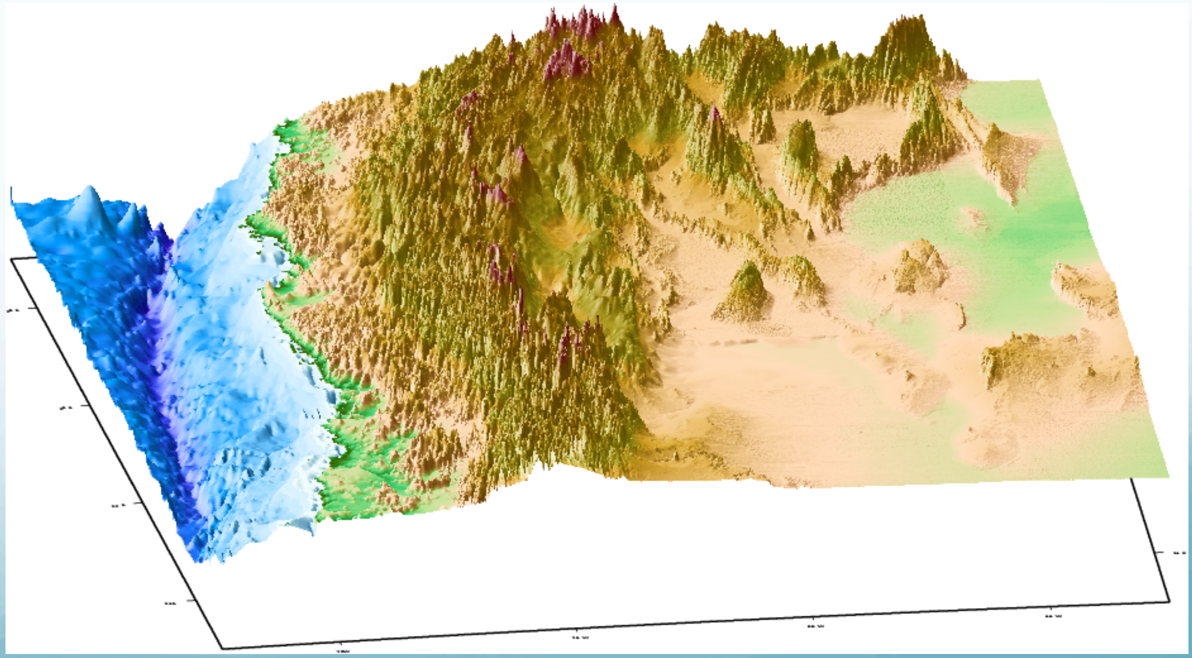
Use to get back azimuth and distance to earthquakes at a glance.



We want to plot

Earthquakes
Focal Mechanisms/Moment Tensors
Digitized geologic data
Topography/Bathymetry
Other Geophysical Data
Roads, Cities, etc.

What tools are there to handle these data sets?



GMT is one of them.

```
#!/bin/sh -f
#make a simple map with point data

LATMIN=10
LATMAX=30
LONMIN=-80
LONMAX=-55
SCALE=0.6
MEDYELLOW=255/255/192
LTBLUE=192/192/255
RED=255/0/0
DONTCLOSE=-K
DONTINIT=-O
CONTINUE="-K -O"
INVLATLON="-:"

pscoast -R$LONMIN/$LONMAX/$LATMIN/$LATMAX -Jm${SCALE} \
-B10 -G$MEDYELLOW -S$LTBLUE $DONTCLOSE -P > $0.ps
psxy -R -Jm${SCALE} -Sc0.2 -G$RED -W1/0 $DONTINIT \
$INVLATLON << END >> $0.ps
`preqs2gmt.sh`
END
```

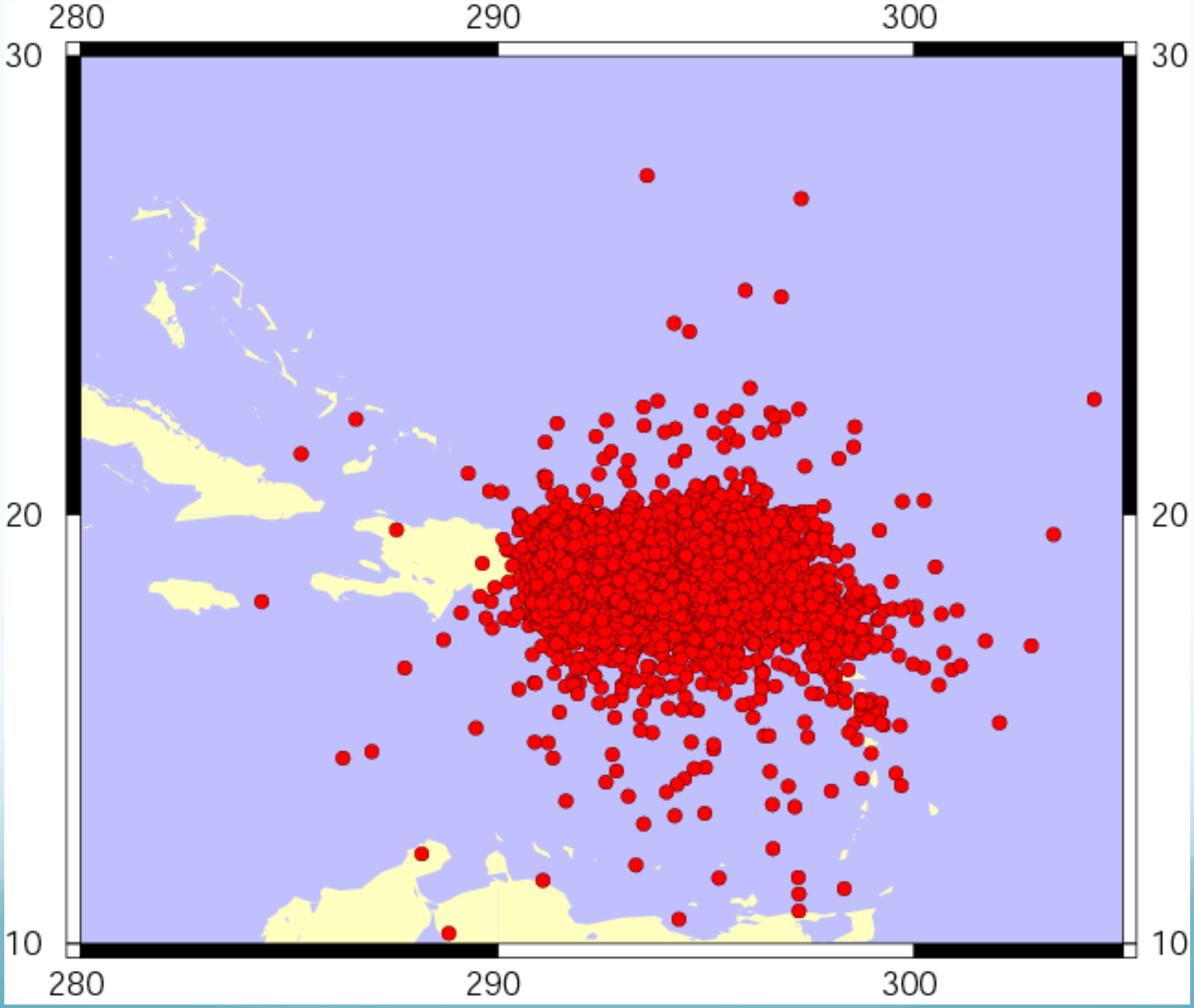
Set it up

pscoast to draw background map

psxy to draw the earthquakes (red circles with black outline)

preqs2gmt.sh (PuertoRican eqs to gmt) to prepare data on the fly, reads file puts out lat long

result



Example of GMT man page – expanded for understanding

psxy reads (x,*y*) pairs from *files* [or standard input] and generates *PostScript* code that will

Plot

lines, polygons, or symbols

at those locations on a map.

Plotting symbols

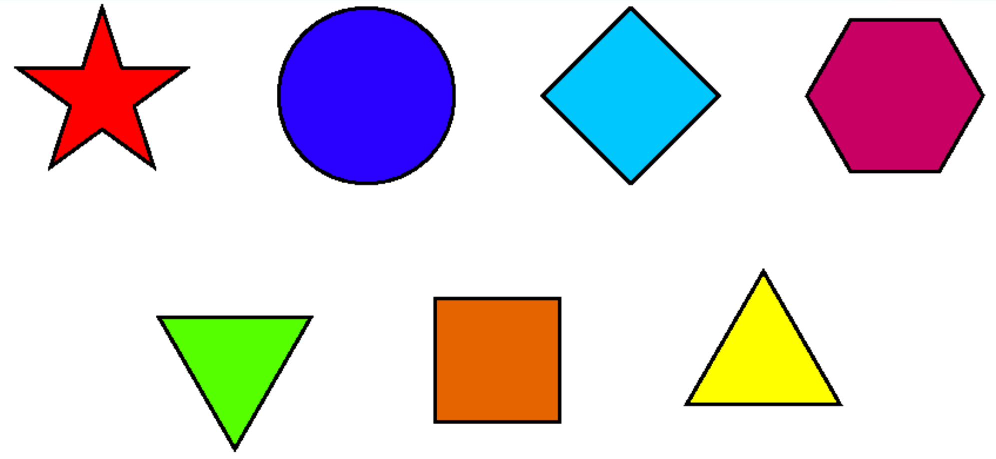
```
psxy -R -Jm${SCALE} -Sc0.2 -G$RED -W1/0 $DONTINIT \
$INVLATLON << END >> $0.ps
```

Flag is **-S**

Symbols you can plot with `psxy` – Point data

Star - `a`
Bar - `b`
Circle - `c`
Diamond – `d`
Ellipse – `e, E`
Front – `f`
Hexagon - `h`
Inverted triangle - `i`
Letter - `l`
Point - `p`

Square - `s`
Triangle - `t`
Vector – `v, V`
Wedge - `w`
Cross - `x`



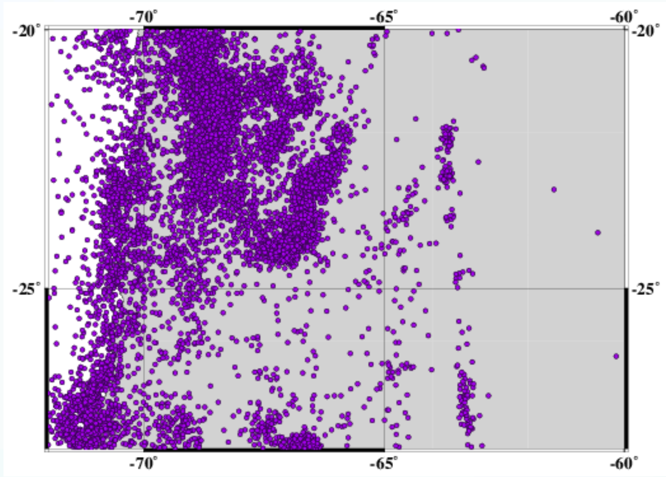
Plotting symbols

```
nawk '/^ PDE/ {print $6, $7}' $0.htm | psxy -R$REGION
$PROJ -Sc0.1 -Gpurple -L  -W.1/0 -: $CONTINUEPS $VERBOSE
>> $OUTFILE
```

-S specify symbol type and size.

Plotting symbols

```
nawk '/^ PDE/ {print $6, $7}' $0.htm | psxy -R$REGION
$PROJ -Sc0.1 -Gpurple -L  -W.1/0 -: $CONTINUEPS $VERBOSE
>> $OUTFILE
```



-S specify symbol type and size.

Use nawk to get data into psxy

```
PDE-W  2011  03 19 083501      -21.98  -68.87 117  5.2 MwRMT  4FM .......
PDE-W  2011  03 19 091240.87 -20.13  -69.08 102  4.7 MwRMT  3FM .......
PDE-W  2011  03 20 013306.18 -24.07  -66.79 189  4.5 mbGS   ... .......
PDE-W  2011  03 20 171225      -24.87  -70.20  49  5.0 MwRMT  4FM .......
PDE-W  2011  03 23 025737      -20.19  -70.82  26  4.0 mbGS   ... .......
PDE-W  2011  03 27 115427      -21.27  -70.29  58  4.3 mbGS   3F. .......
PDE-W  2011  03 29 114934      -20.10  -69.95  60  4.8 mbGS   4F. .......
PDE-W  2011  03 31 040737.45 -24.06  -66.65 181  4.5 mbGS   ... .......
PDE-W  2011  03 31 214111.87 -27.64  -67.32  61  4.1 mbGS   4F. .......
```

Plotting symbols

Setting size on the fly from the data

```
nawk '/^ PDE/ {print $6, $7}' $0.htm | psxy -R$REGION
$PROJ –Sc -Gpurple -L  -W.1/0 -: $CONTINUEPS $VERBOSE >>
$OUTFILE
```

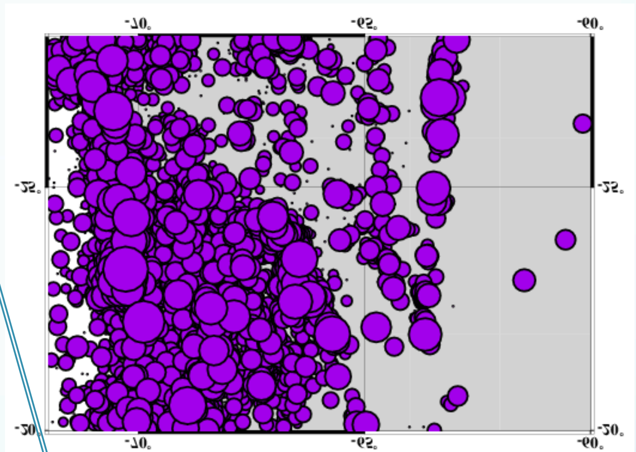
If a symbol is selected and no symbol size given, then psxy will interpret the third column of the input data as symbol size.

Symbols whose size is <= 0 are skipped.

Plotting symbols

Setting size on the fly from the data

```
nawk '/^ PDE/ {print $6, $7, ($9>0)?($9^2)/64:"0.01"}'
$0.htm | sort -k 3 -n | psxy -R$REGION $PROJ -Sc -Gpurple
-L  -W.1/0 -: $CONTINUEPS $VERBOSE >> $OUTFILE
```



-S set size on fly.

Use nawk to get data into psxy

PDE-W	2011	03	19	083501	-21.98	-68.87	117	5.2	MwRMT	4FM	....
PDE-W	2011	03	19	091240.87	-20.13	-69.08	102	4.7	MwRMT	3FM	....
PDE-W	2011	03	20	013306.18	-24.07	-66.79	189	4.5	mbGS	...	....
PDE-W	2011	03	20	171225	-24.87	-70.20	49	5.0	MwRMT	4FM	....
PDE-W	2011	03	23	025737	-20.19	-70.82	26	4.0	mbGS	...	....
PDE-W	2011	03	27	115427	-21.27	-70.29	58	4.3	mbGS	3F.	....
PDE-W	2011	03	29	114934	-20.10	-69.95	60	4.8	mbGS	4F.	....
PDE-W	2011	03	31	040737.45	-24.06	-66.65	181	4.5	mbGS	...	....
PDE-W	2011	03	31	214111.87	-27.64	-67.32	61	4.1	mbGS	4F.	....

Plotting Symbols

Setting color on the fly based on data.

```
nawk '/^ PDE/ {print $6, $7, $8, ($9>0)?($9^2)/64:"0.01"}'
$0.htm | sort -k 3 -n | psxy -R$REGION $PROJ -Sc -L
-Ceq.cpt -W5/0 -: $CONTINUEPS $VERBOSE >> $OUTFILE
```

`-C` Give a color palette (cpt) file (don't need –G, which fills symbol, anymore).

When used with `-S`, lets symbol color be determined by the `z`-value in the third column.

Additional fields are shifted over by one column (optional size would be 4th rather than 3rd field, etc.).

Plotting Symbols

Setting color on the fly based on data.

color palette (cpt) file.

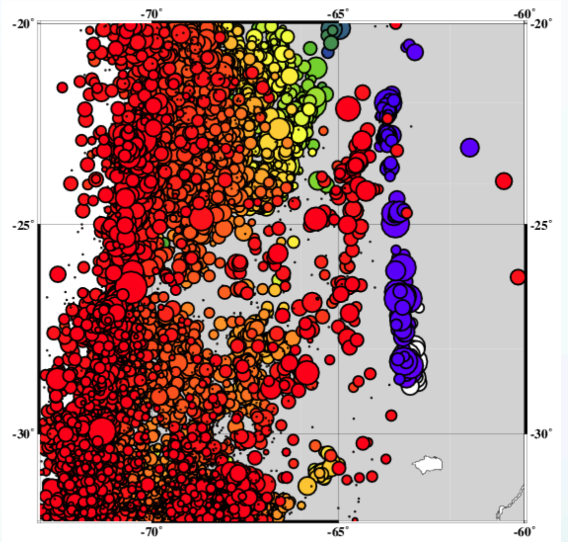
```
$ cat eq.cpt
000 255 000 000 100 255 100 000
100 255 100 000 200 255 255 000
200 255 255 000 300 100 200 000
300 100 200 000 400 000 000 255
400 000 000 255 600 100 000 255
```

Simple cpt file – define start and stop of non-overlapping data ranges and color for each range. Gives linear variation between limits.

$z_{1bottom}$ r g b z_{1top} r g b
$Z_{2bottom}$ r g b z_{2top} r g b
…

Plotting symbols
Setting color on the fly based on data.

```
nawk '/^ PDE/ {print $6, $7, $8, ($9>0)?($9^2)/64:"0.01"}'
$0.htm | sort -k 3 –n -r | psxy -R$REGION $PROJ -Sc -L
-Ceq.cpt -W5/0 -: $CONTINUEPS $VERBOSE >>
```



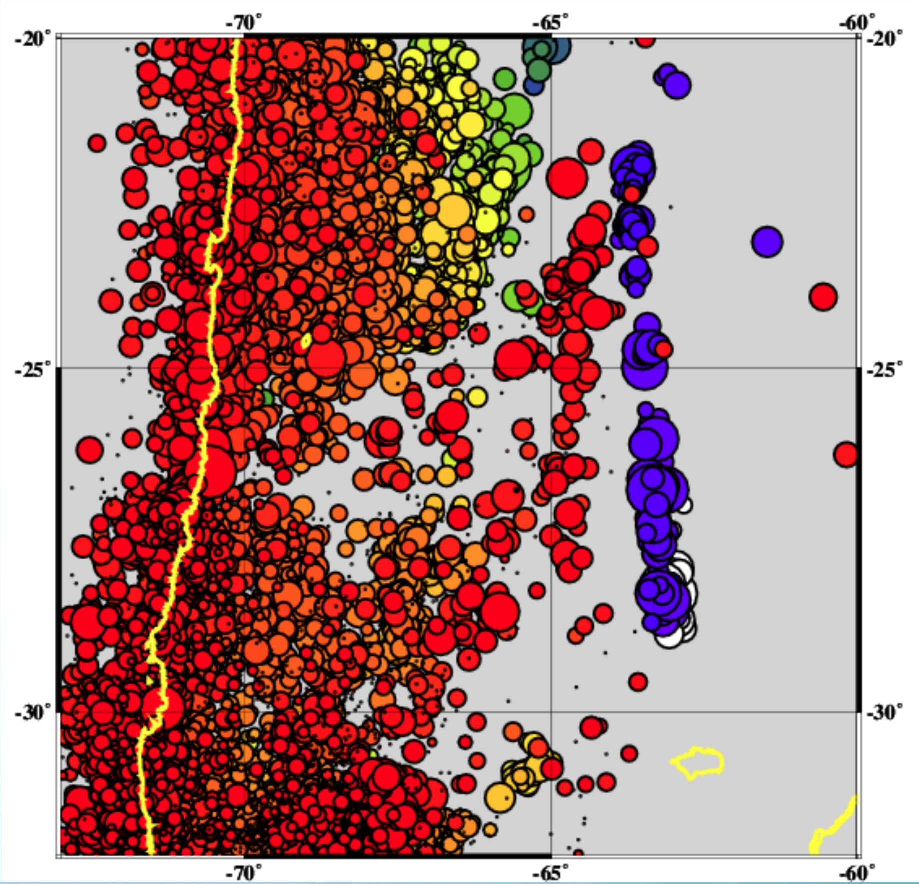
Setting color on the fly

Use nawk to get data into psxy

```
PDE-W  2011  03 19 083501     -21.98  -68.87 117  5.2 MwRMT   4FM .......
PDE-W  2011  03 19 091240.87  -20.13  -69.08 102  4.7 MwRMT   3FM .......
PDE-W  2011  03 20 013306.18  -24.07  -66.79 189  4.5 mbGS    ... .......
PDE-W  2011  03 20 171225     -24.87  -70.20  49  5.0 MwRMT   4FM .......
PDE-W  2011  03 23 025737     -20.19  -70.82  26  4.0 mbGS    ... .......
PDE-W  2011  03 27 115427     -21.27  -70.29  58  4.3 mbGS    3F. .......
PDE-W  2011  03 29 114934     -20.10  -69.95  60  4.8 mbGS    4F. .......
PDE-W  2011  03 31 040737.45  -24.06  -66.65 181  4.5 mbGS    ... .......
PDE-W  2011  03 31 214111.87  -27.64  -67.32  61  4.1 mbGS    4F. .......
```

Open with psbasemap.

Draw pscoast and grid last.



Plotting symbols

Setting symbol on the fly from the data

```
psxy -R -Jm${SCALE} -S -G$RED -W1/0 $DONTINIT \
$INVLATLON << END >> $0.ps
```

If no symbols are specified in the command line, then the symbol code must be present as last column in the input.

Definition of size for each symbol – look at man page

```
psxy -R -Jm${SCALE} –Sa0.5 -G$RED -W1/0 $DONTINIT \
$INVLATLON << END >> $0.ps
```

-Sa star. size is <u>diameter of circumscribing circle</u>.

Ellipse symbol

-Se ellipse. Direction (in degrees counter-clockwise from horizontal), major_axis, and minor_axis must be found in columns 3, 4, and 5.

-SE Same as -Se, except azimuth (in degrees east of north) should be given instead of direction.

The azimuth will be mapped into an angle based on the chosen map projection (-Se leaves the directions unchanged.) Furthermore, the axes lengths must be given in km instead of plot-distance units.

Vectors – center (?) on specified (x,y)

-Sv vector. Direction (in degrees counter-clockwise from horizontal) and length must be found in columns 3 and 4. *size*, if present, will be interpreted as arrowwidth/headlength/headwidth [Default is 0.075c/0.3c/0.25c (or 0.03i/0.12i/0.1i)]. By default arrow attributes remains invariant to the length of the arrow. To have the size of the vector scale down with decreasing size, append nnorm, where vectors shorter than norm will have their attributes scaled by length/norm. -SV Same as -Sv, except azimuth (in degrees east of north) should be given instead of direction. The azimuth will be mapped into an angle based on the chosen map projection (-Sv leaves the directions unchanged.)

Plot lines

leave out symbol flag, -s, will connect data points with great circle line segments.

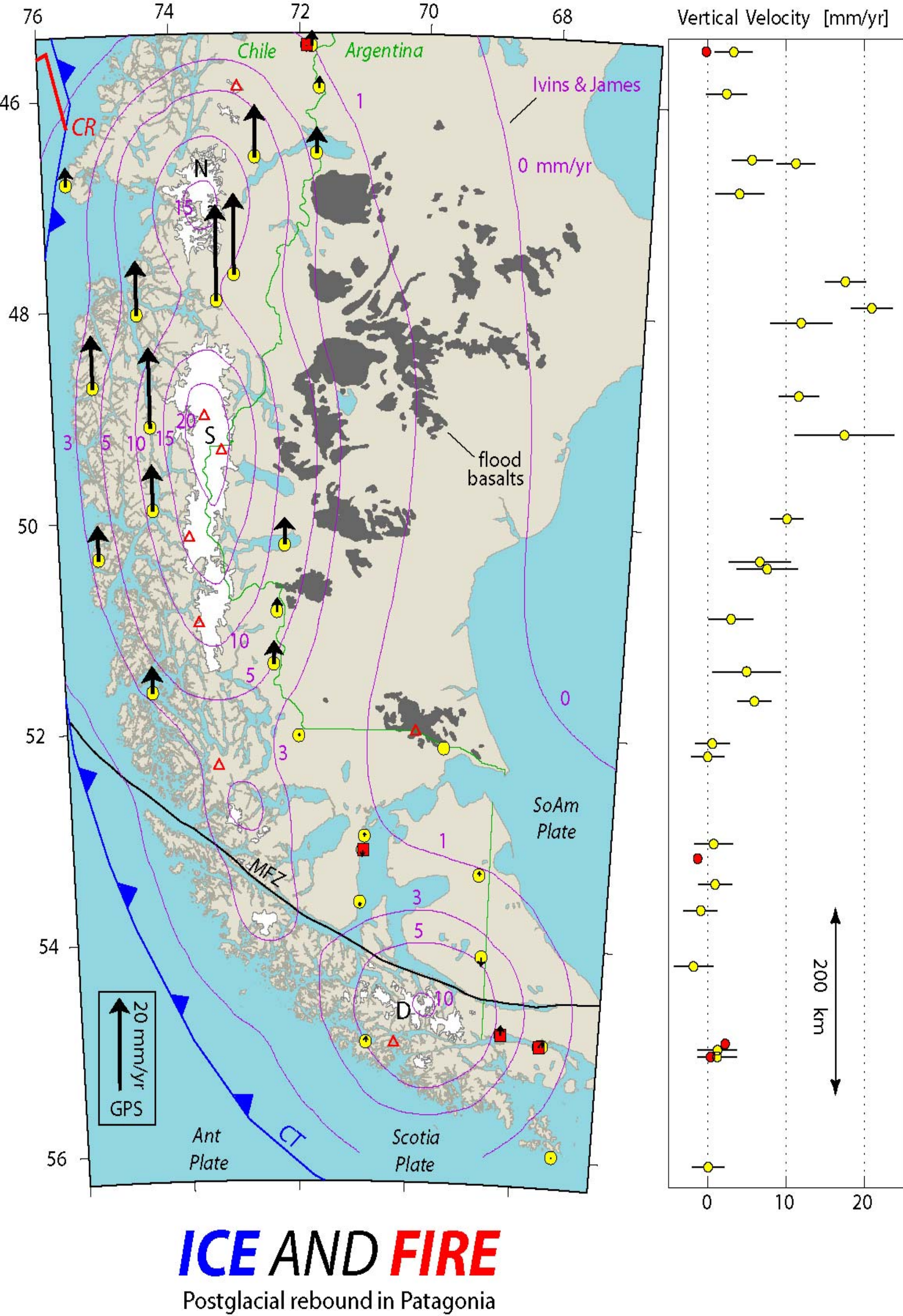
```
psxy -R$REGION $PROJ -W3/$RED $CONTINUEPS ridges >> $OUTFILE
```

(use **-A** to suppress great circle – I suppose it draws straight line between projected points, never used it).

```
psxy -R$REGION $PROJ –M -W3/
purple $CONTINUEPS
pgr_contours.dat >> $OUTFILE
```

Multiple segment files ("lift pen") may be plotted using the -`Mflag` option.

Segments are separated by a record whose first character is `flag`. [Default is '>'].



To explicitly close polygons when drawing lines (with great circle segment), use `-L`.

Need to do this is you want to fill the polygon.

Fill – for symbols and closed polygons defined by lines.

Shade interior with `-G`. If `-G` is set, `-W` (line width and color) will control whether the polygon outline is drawn or not.

If a symbol is selected and `-G` set, `-W` determines the fill color and outline/no out- line, respectively.

-G Select filling of polygons and symbols. Append the shade (0-255), color (r/g/b), or P|p*dpi/pattern* (polygons only) [Default is no fill]. Note when -M is chosen, *psxy* will search for -G and -W strings in all the subheaders and let any found values over-ride the command line settings.

Front symbol – goes on lines

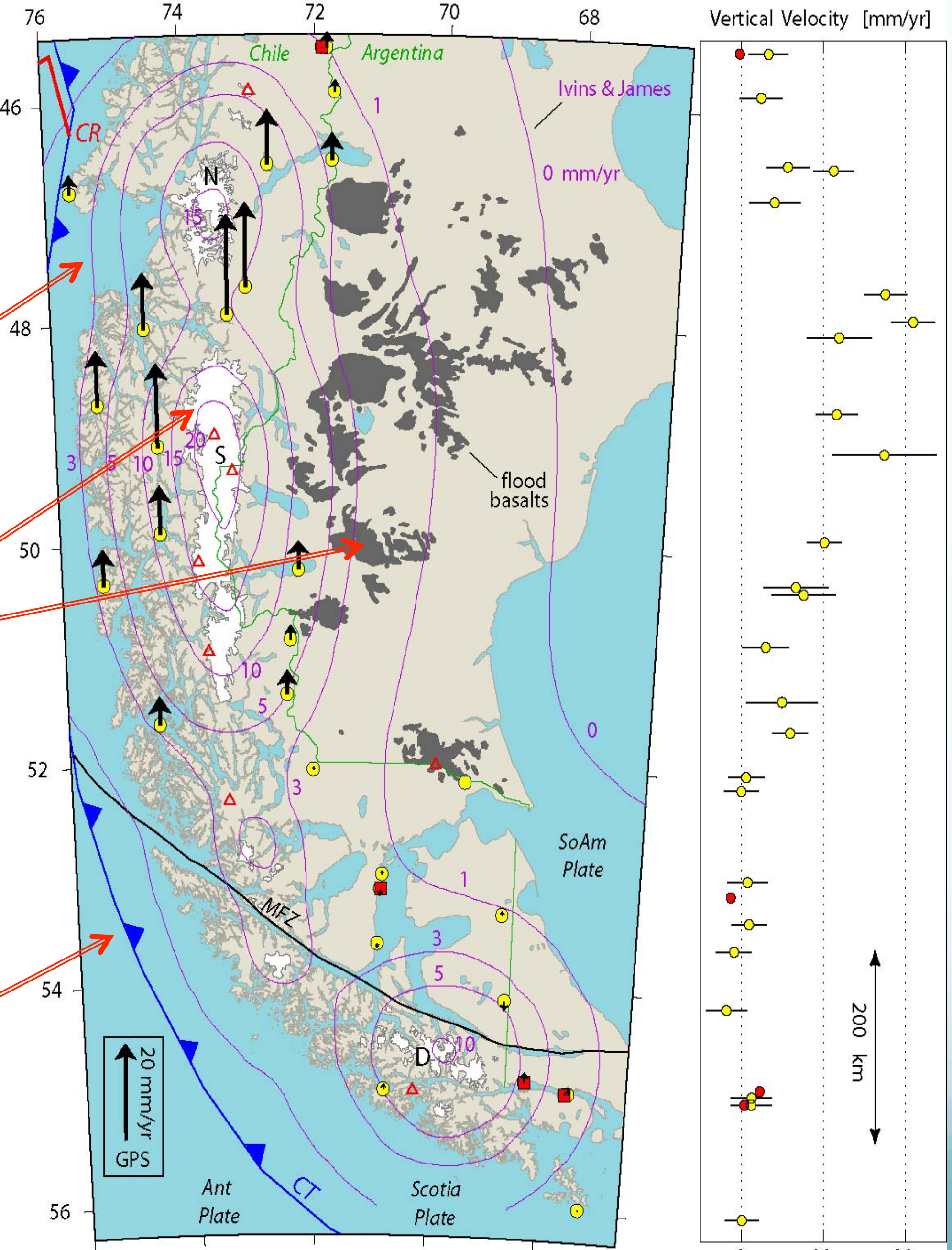
-Sf front. -Sfgap/size[dir][type][:offset]. Supply distance gap between symbols and symbol size. If gap is negative, it is interpreted to mean the number of symbols along the front instead. Append dir to plot symbols on the left or right side of the front [Default is centered]. Append type to specify which symbol to plot: box, circle, fault, slip, or triangle. [Default is fault]. Slip means left-lateral or right-lateral strike-slip arrows (centered is not an option). Append :offset to offset the first symbol from the beginning of the front by that amount [Default is 0].

Plot lines

Just plot line

Plot and fill (lines have form closed polygons.)

Geologic symbols (subduction zone)



ICE AND FIRE

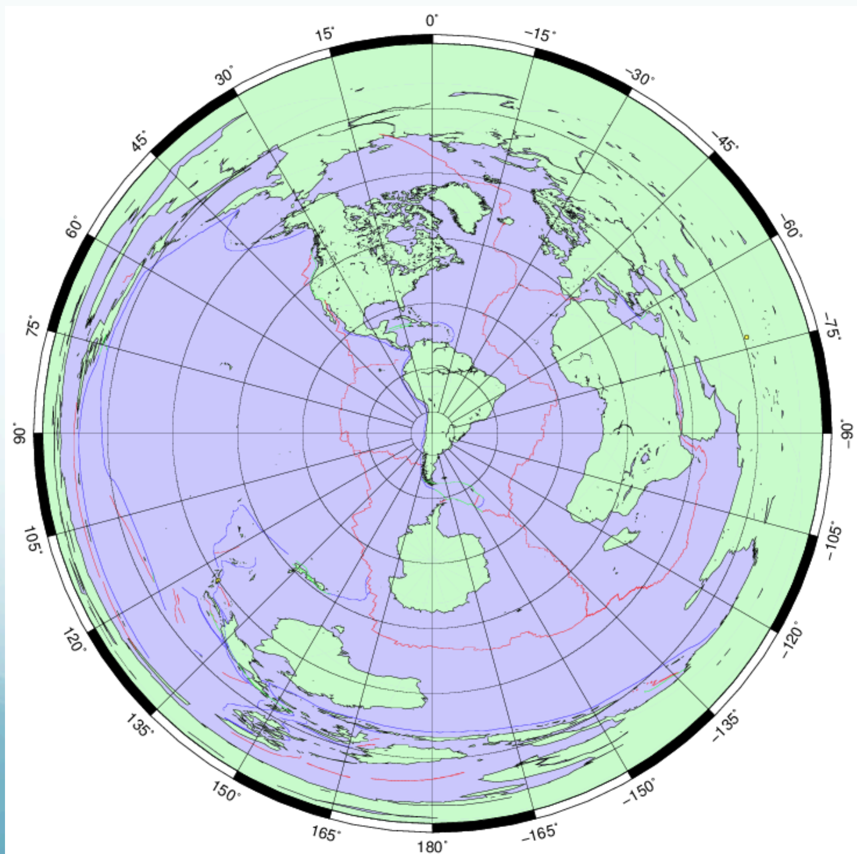
Postglacial rebound in Patagonia

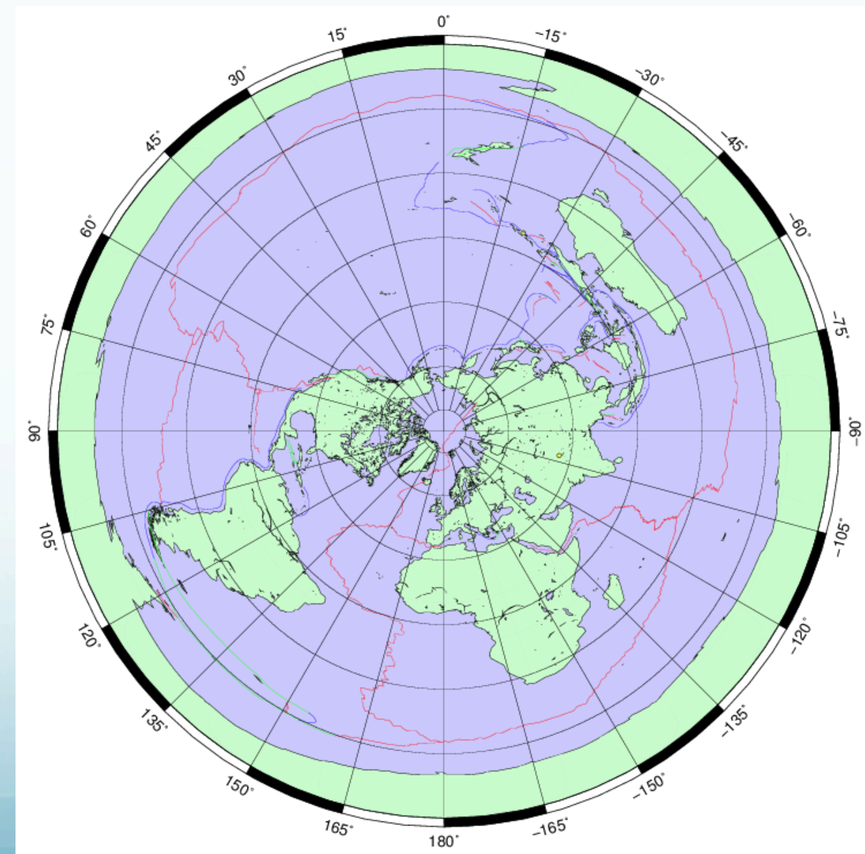
Fill – problems

If polygon not closed properly psxy draws ¿great circle or straight? line from first to last point and fills in closed polygons that this line creates (line from beginning to end may result in multiple polygons being formed (it closes an S shape with a straight line and you get two opposite facing half circles).

Fill – problems

The anti-pode fill problem seems to have gone away (except falls over when center is exactly North or South poles, even though no land at N pole).





Plotting lines

Both Symbol outline (a line) and Line properties are specified using –W switch.

Line thickness, color, pattern/texture.

```
-W Set pen attributes. [Defaults: width = 1, color = 0/0/0,
texture = solid]. Implicitly draws the outline of symbols with
selected pen.
```

Plotting lines

Setting color on the fly based on data.

```
psxy -R -Jm${SCALE} -S —CANDES.cpt -W1/0 $DONTINIT \
$INVLATLON << END >> $0.ps
`Feed in data`
END
```

If `-S` is not set (drawing lines), `psxy` expects the user to supply a multisegment polygon file (requires `-M`) and will look for `-Zval` strings in each multisegment header. The `val` will control the color via the cpt file.

Input geographic data order.

GMT was written by guys who made x-y plots.

x comes first, y comes second.

This means longitude comes first, latitude comes second (default would have been other way around if written by cartographer.)

(why is clockwise the direction it is?)

To switch data order use the `-:` switch

This is an important one – switches the order of ALL the grid referenced data on the input line.

pxsy pretty powerful but does not draw all the symbols needed for geophysics

Two important items not covered by psxy

psmeca - Focal Mechanisms/Moment Tensors

psvelo - Vectors with error ellipses

(replaced older psvelomeca program that broke UNIX philosophy by mixing two unrelated tasks).

Make focal mechanisms – use GMT filter (program/routine) **psmeca**

make/obtain input file – see **psmeca** documentation for large number of ways to define focal mechanism data

35.59	-90.48	12	220	65	150	4.5975	-0.25	-0.25
35.86	-89.95	16	220	75	150	4.0727	-0.25	0.25
36.37	-89.51	7.5	350	84	145	4.2020	-0.25	0.25
36.54	-89.68	9	85	60	-20	3.7118	0	0.5
36.56	-89.83	8	90	67.5	20	4.1068	-0.25	-0.25
36.64	-90.05	15	304	78	-28	4.6309	0	-0.5
37.16	-89.58	15	140	75	50	4.2547	0.25	0
37.22	-89.31	1.5	280	70	-20	3.5783	-0.25	0.25
37.36	-89.19	16	30	70	170	3.8250	0.25	0.25
37.44	-90.44	15	350	60	135	4.0126	0.25	0.25
37.48	-90.94	5	260	40	-70	4.5728	0.25	-0.25
37.91	-88.37	22	0	46	79	5.2612	-0.35	0.1
38.55	-88.07	15	310	70	0	4.3154	-0.25	-0.25
38.71	-87.95	10	135	70	15	4.9309	-0.25	0.25

Specify how data for focal mechanism is specified.

-Sa - Focal mechanisms in Aki and Richard convention

-Sc - Focal mechanisms in Harvard CMT convention

-Sm - Seismic moment tensor (Harvard CMT, with zero trace)

-Sp - Focal mechanisms given with partial data on both planes.

Scale follows selection letter, adjusts the scaling of the radius of the "beach ball", which will be proportional to the magnitude (x is one of a,c,m,p).

Make map with focal mechanisms (psmeca) and earthquake locations (psxy)

```
#!/bin/sh -f
REG=-92/-88/35/39
psmeca -R$REG << END -Jm4. -Bg1f1a1 -P -Sa2./0/0 -CP -: -K > $0.ps
`nawk '{print $1, $2, $3, $4, $5, $6, $7, $1+$8, $2+$9}'
practice_data.dat`
END
psxy -R$REG practice_data -Jm4. -Sc0.25 -: -G255/0/0 -W3/0 -O >> $0.ps
```

-S flag in psmeca for focal mechanism input format definition and size

-C for plotting beach ball offset from earthquake location and, PW, for connecting it to point at earthquake location with a line W thick.

Specify how size changes with respect to magnitude.

-Sx*scale* adjusts the scaling of the radius of the "beach ball", which will be proportional to the magnitude.

Scale is the size for magnitude = 5 (that is seismic scalar moment = 4*10e+23 dynes-cm) in inch (unless **c**, **i**, **m**, or **p** is appended). (**-T***0* option overlays best double couple transparently.)

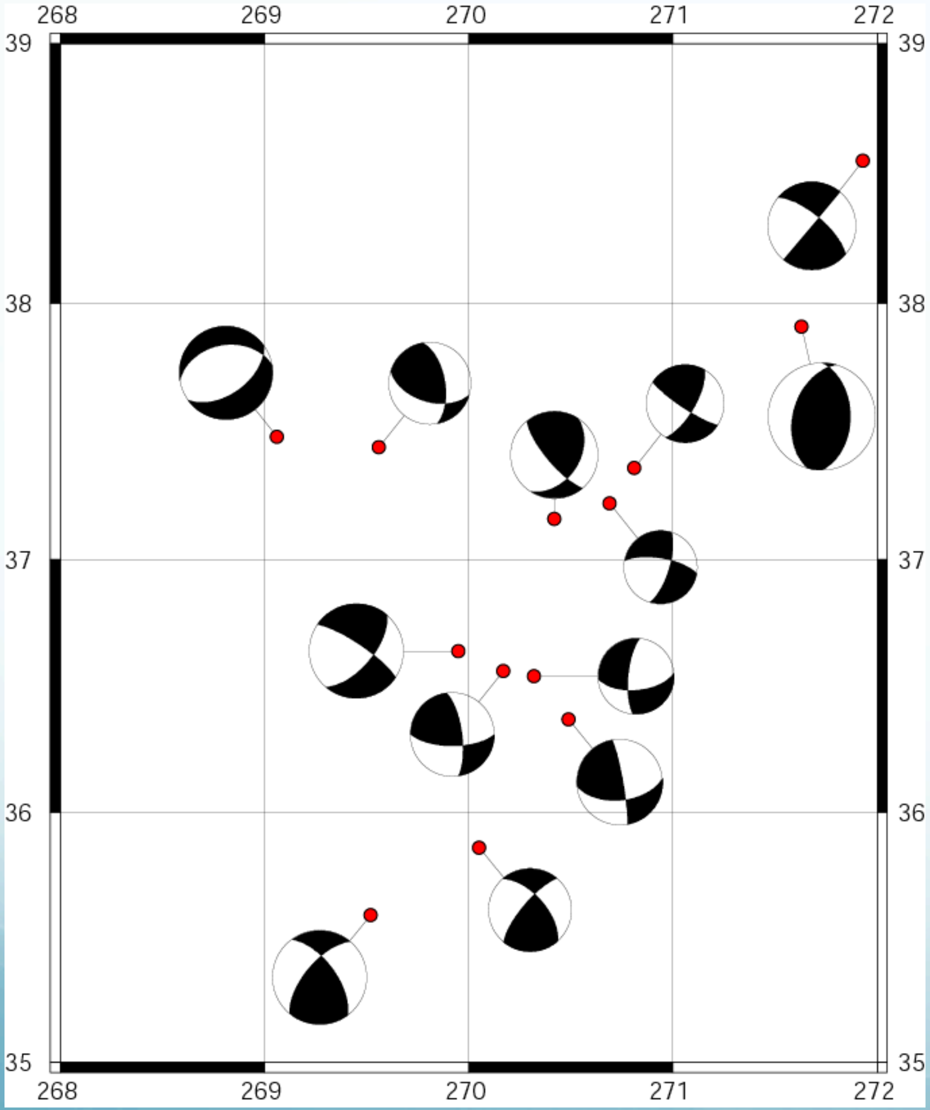
Put **-Sy***scale[c/i][/fontsize[/offset[u]]]* to plot the only double couple part of moment tensor. Put **-St***scale[c/i][/fontsize[/offset[u]]]* to plot zero trace moment tensor. The color or shade of the compressive quadrants can be specified with the **-G** option. The color or shade of the extensive quadrants can be specified with the **-E** option. Parameters are expected to be in the following columns

coloring.

The color or shade of the compressive quadrants can be specified with the **-G** option. The color or shade of the extensive quadrants can be specified with the **-E** option. Parameters are expected to be in the following columns

```
35.59  -90.48 12  220 65  150 4.5975 -0.25  -0.25
```

```
`nawk '{print $1, $2, $3, $4, $5, $6, $7, $1+$8, $2+$9}'
```

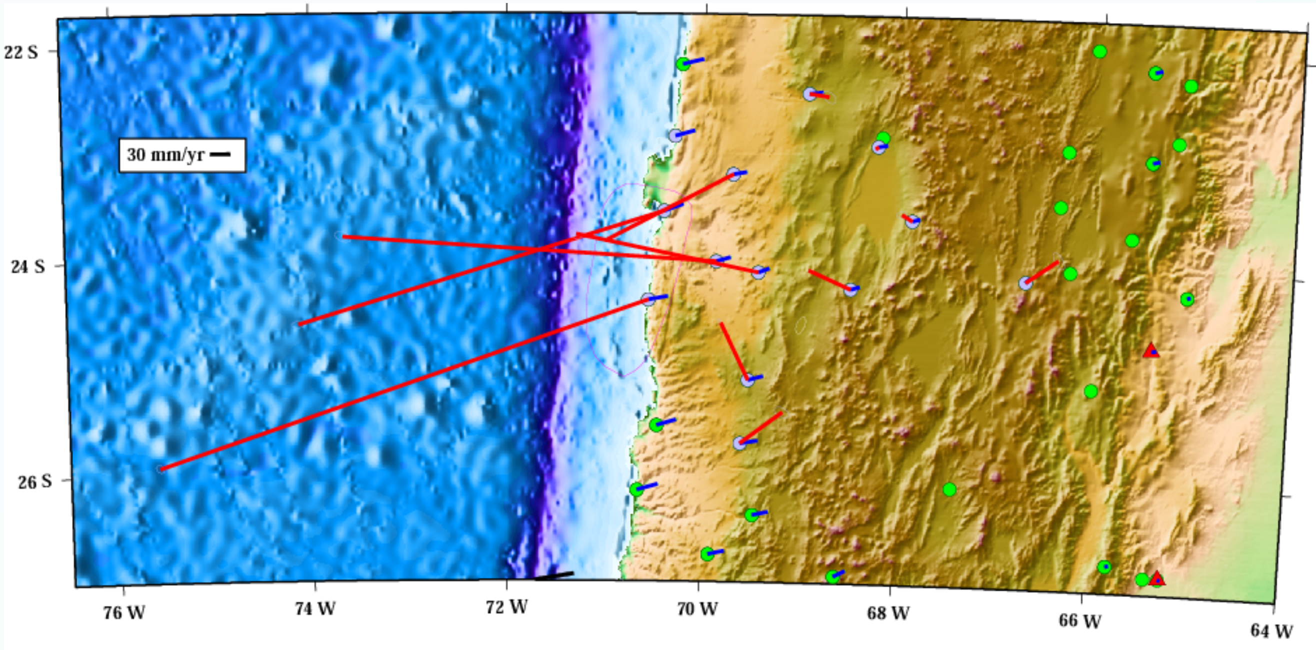


Uses "offsets" specified in columns 8 and 9 to reposition the focal mechanism.

You could put the lat, long you wanted in cols 8 and 9, but why calculate all of them by hand?

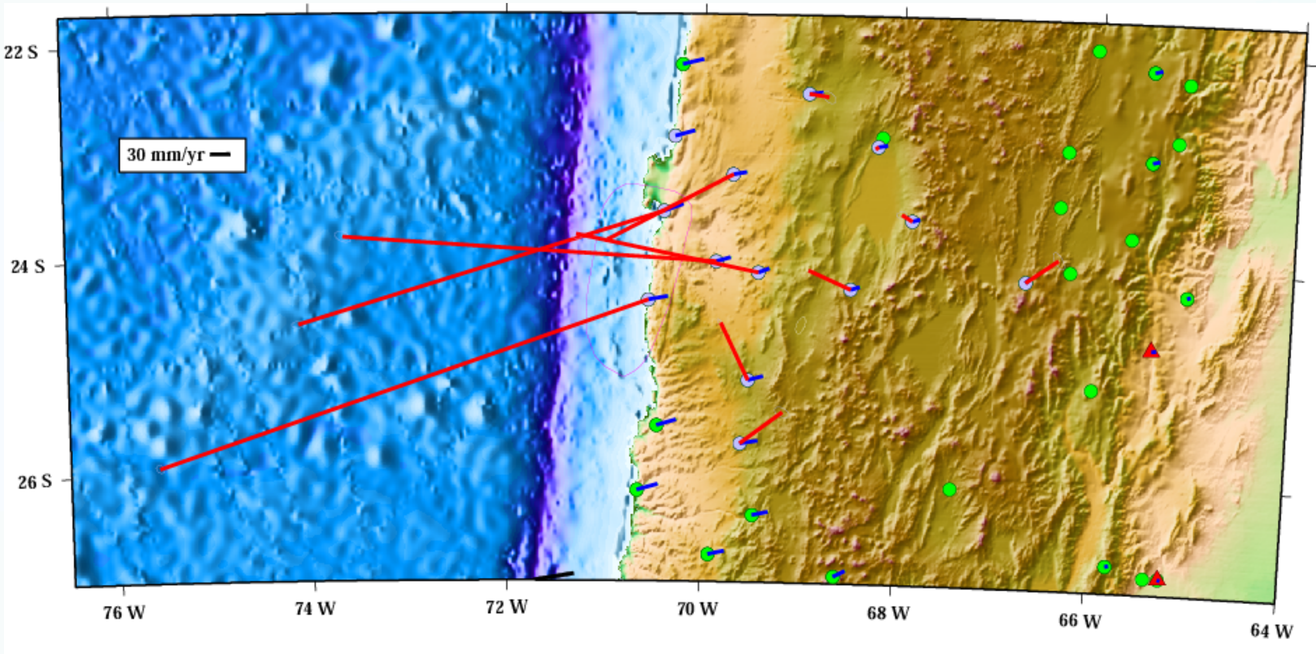
If you have to specify the offsets for each beachball depending on how things look (example to left), no easy way to do automatically, have to type in offsets or locations.

Are lower hemisphere plots.

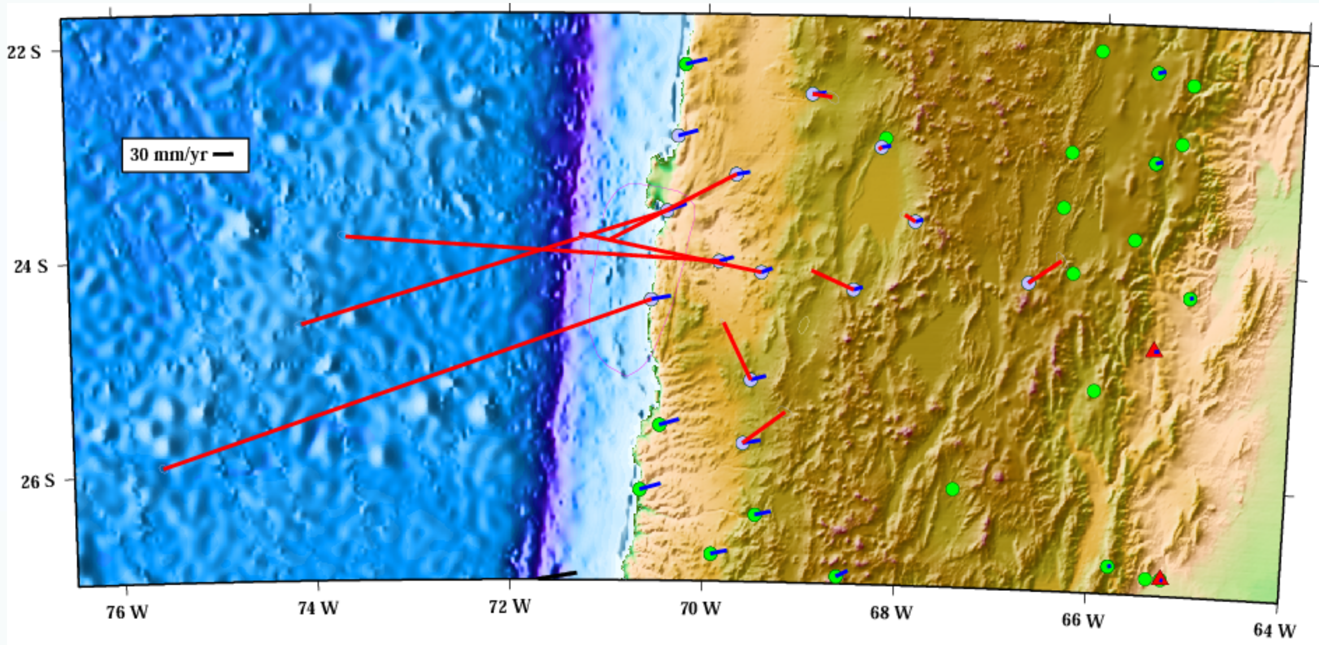


Plot

- Velocity vectors with error ellipses
- Anisitropy bars
- Rotational wedges
- Strain crosses




```
psvelo -R -$PROJ$SCALE -Sr$VELLEN/0.95/0 -W1/$PURPLE -G$PURPLE \
$VELARROW $CONTINUE $VBSE andaman_nicobar_coseis.dat \
>> $OUTPUTFILE
```



Various ways to define vector data

(ve, vw, or mag, az)

Vector length, error ellipse confidence for plot, label font size

Arrow shaft width, head length and width

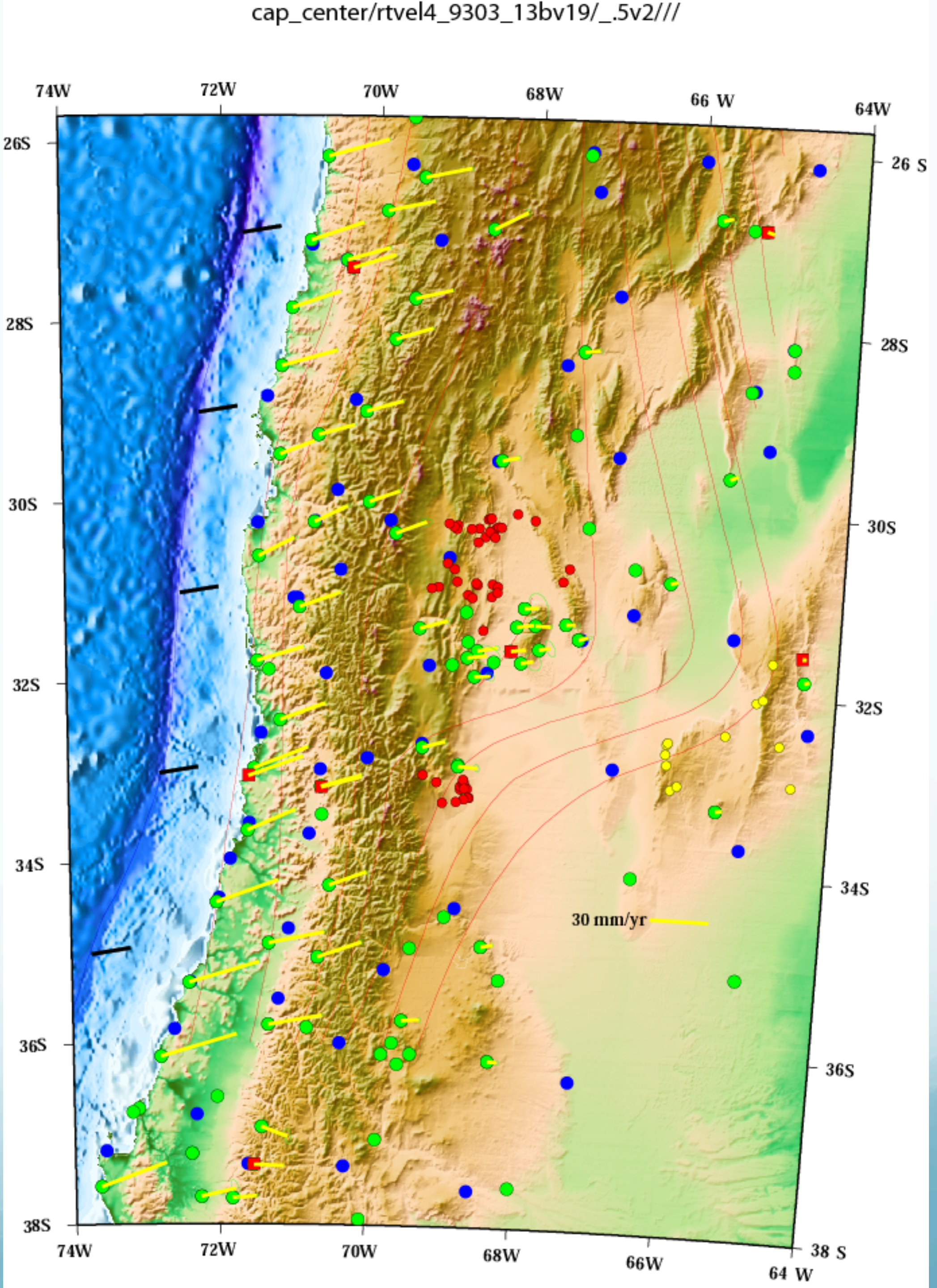
Data - lat lon vlat vlon siglat siglon corr

making pretty MAPS

How to do:

- color or b&w topo with shaded topo

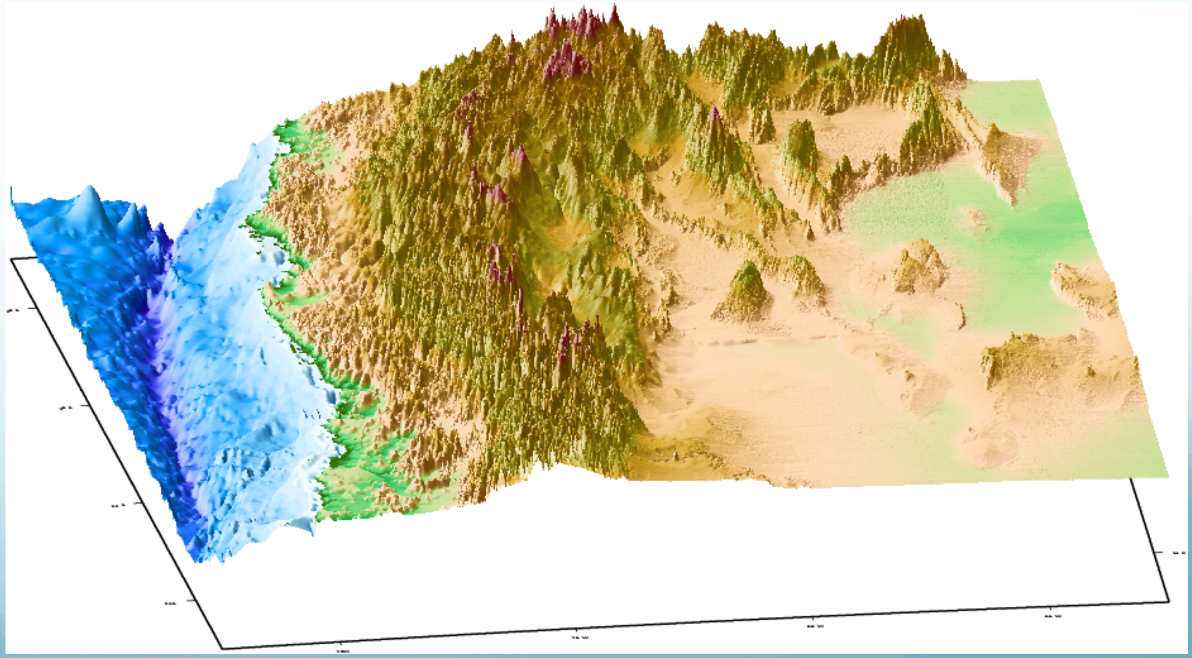
- how to combine topo and bathymetry



First – have to find data – what's available

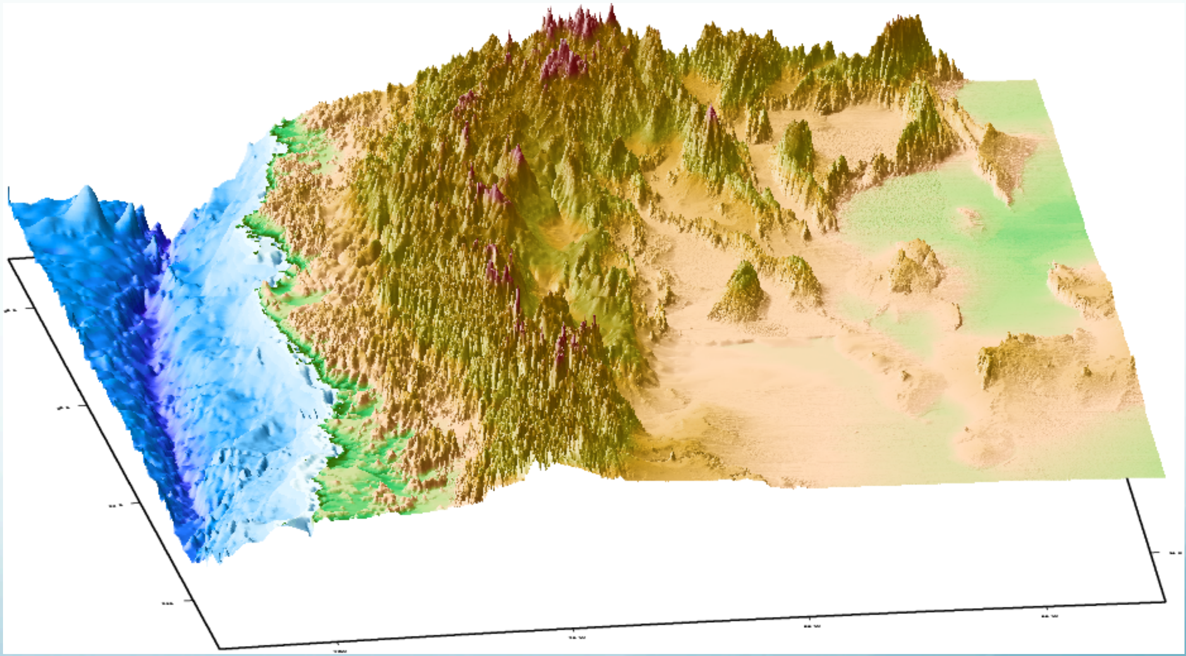
DEM's (Digital Elevation Models) of world – several resolutions, several kinds of data (GTOPO-30, ETOPO-5 , SRTM, seasat, obs/ predicted bath)

Really raster data (value on grid or in volume) – sucha as gravity, age sea floor, etc.



Where to get them?

(We have some online at CERI – makes it easy. Have not fully figured out SRTM yet.)



use `grdraster` to extract a subregion from the global bathymetry data set and make a new grid file for GMT.

`grdraster` is not part of "standard" GMT.

Is a "supplemental" GMT program.

There are a bunch (order 35-40) of such supplemental GMT programs like this around.

Many are written by others (not Smith and Wessell) and become “attached” to GMT and can be found on the GMT web page, but they are not officially part of GMT.

`psmeca` and `psvelo` (to draw focal mechanisms and vector fields) are in this class.

use grdraster to extract a subregion from the global bathymetry data set and make a new grid file for GMT.

$GRDRASTERREGION has same format at the REGION definition (min lon/max lon/min lat/max lat) and been previously set up to define the region

```
echo do seafloor
DATASET=10
DATAGRID=-I2m/2m
grdraster $DATASET -G${ROOTNAME}_2mtopo.grd $DATAGRID \
-R$GRDRASTERREGION -V
echo done with 2m topo grdraster
```

Let's look at the documentation first

Typing `grdraster` all by itself dumps the man page (GMT default behavior).

- reports

available data sets

Units

data coverage area

spacing and registration (pixel or grid – not important for now, except that when combining data sets they have to be the same).

```
alpaca/smalley 142:> grdraster
grdraster 3.4.3 - Extract a region from a raster and save in a grdfile
usage: grdraster <file number> -R<west/east/south/north>[r] \
[-G<grdfilename>] [-I<dx>[m][/<dy>[m]]][-bo[s][<n>]]
        <file number> (#) corresponds to one of these:

#  Data Description        Unit    Coverage     Spacing Registration
-------------------------------------------------------------------------
1 "ETOPO5 global topography"    "m"     -R0/359:55/-90/90      -I5m    G
2 "US Elevations from USGS"     "m"     -R234/294/24/50        -I0.5m  P
3 "Geo/Seasat grav from Haxby"  "mGal"  -R0/359:55/-90/90      -I5m    G
4 "Geo/Seasat geoid from Haxby" "m"     -R0/359:55/-90/90      -I5m    G
5 "Sea floor age from Cande"    "Ma"    -R0/359:55/-90/90      -I5m    P
6 "Sea floor age from Muller et al., 1998" "Ma" -R0/360/-72/90 -I6m    G
7 "Sea floor age errors Muller et al., 1997" "Ma" -R0/360/-72/72 -I6m  G
8 "1=land, 0=sea bitmask"       "T/F"   -R0/360/-90/90          -I5m   P
9 "USGS/SS ETOPO30s"    "m"             -R0/360/-90/90         -I0.5m  P
10 "2min Observed/Predicted Topo"    "m"   -R0/360/-72/72       -I2m    P
11 "et30wbath"    "m"    -R-78/-63/-25/-12        -I0.5m      P
-------------------------------------------------------------------------
```

First use `grdraster` to extract a subregion from the global data set

```
echo do seafloor
DATASET=10
DATAGRID=-I2m/2m
grdraster $DATASET -G${ROOTNAME}_2mtopo.grd $DATAGRID \
-R$GRDRASTERREGION -V
echo done with 2m topo grdraster
```

We have selected the `2m predicted sea floor topography` – data set `10`.

We have set the grid to the proper sample spacing (get from previous slide w/ data set properties).

First use `grdraster` to extract a subregion from the global data set

```
echo do seafloor
DATASET=10
DATAGRID=-I2m/2m
grdraster $DATASET -G${ROOTNAME}_2mtopo.grd $DATAGRID \
-R$GRDRASTERREGION -V
echo done with 2m topo grdraster
```

We are going to put the extracted data into a file called `${ROOTNAME}_2mtopo.grd`

Now we do the same for the land topographic data, using GTOPO-30, which only has data for land.

```
echo do topo
DATASET=9
DATAGRID=-I30c/30c
grdraster $DATASET -G${ROOTNAME}_topo.grd $DATAGRID \
-R$GRDRASTERREGION -V
echo done with gtopo grdraster
```

Now we select the ETOTO-30 topography – data set 9.

Notice that the grid has a different sample spacing than the bathymetry, otherwise this code snippet is the same.

Now we do the same for the land topographic data, using GTOPO-30, which only has data for land.

```
echo do topo
DATASET=9
DATAGRID=-I30c/30c
grdraster $DATASET -G${ROOTNAME}_topo.grd $DATAGRID \
-R$GRDRASTERREGION -V
echo done with gtopo grdraster
```

The data will go into a file called

`${ROOTNAME}_topo.grd`

We now have two complimentary data sets, one for topography and one for bathymetry and we have to combine them.

(for most maps, the newer, current dem files have land and sea and you don't have to do this – but some datasets still need it.)

Unfortunately, they have different sample spacing.

So we have to resample one of the data sets – lets do it to the sea floor (since it has the lower resolution – we will therefore be interpolating).

Use `grdsample` to resample the bathymetry as defined by `DATAGRID` and put in a new resampled file `${ROOTNAME}_30stopo.grd`

```
echo prep and merge bathy
DATAGRID=-I30c/30c
grdsample ${ROOTNAME}_2mtopo.grd -G${ROOTNAME}_30stopo.grd $DATAGRID \
-F -R$GRDRASTERREGION -V
```

Now we use `grdmath` to combine (AND) the two data sets (they have distinguishing values in the dataless points).

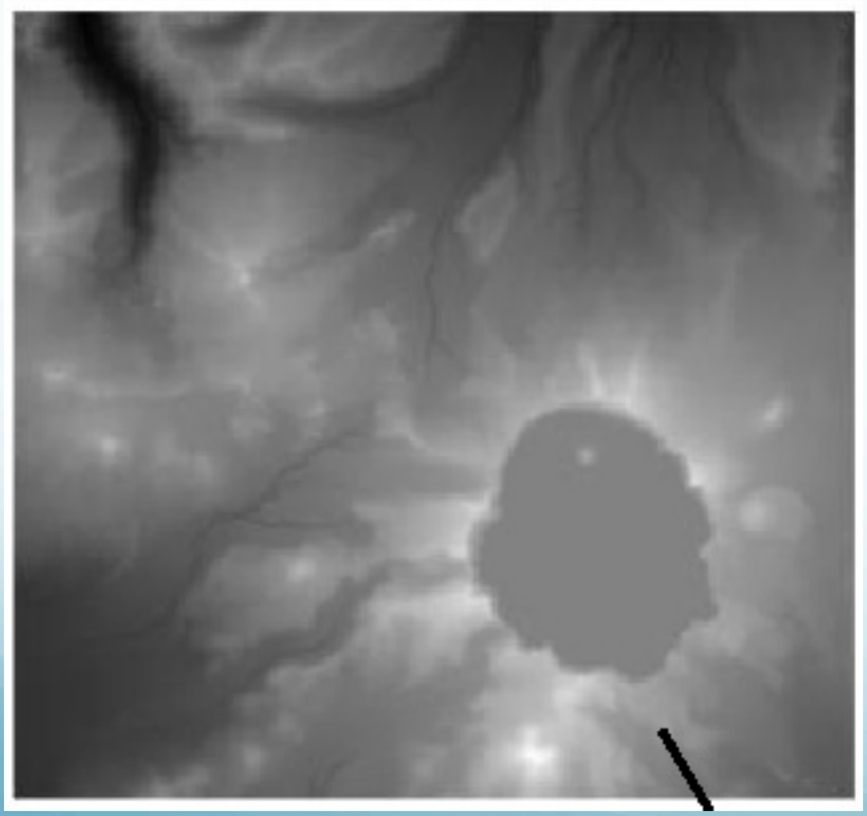
`grdmath` uses a stack and RPN – (Reverse or postfix Polish Notation, as opposed to prefix Polish Notation – what your invention gets called when your ethnic Polish name is unpronounceable in English)

```
grdmath -F -V ${ROOTNAME}_topo.grd ${ROOTNAME}_30stopo.grd AND = \
${ROOTNAME}_topobath.grd
echo done with merge bathy
```

And put the new topo file in `${ROOTNAME}_topobath.grd`

We are now done selecting the topographic and bathymetric data,

which can be used to generate coloring or grayscale.

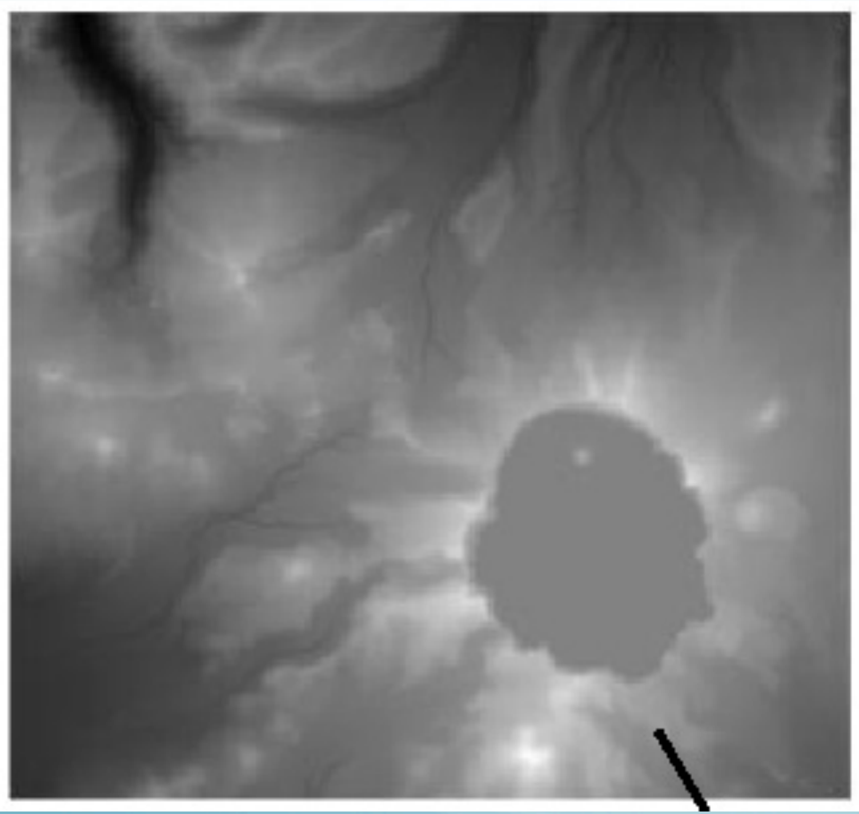


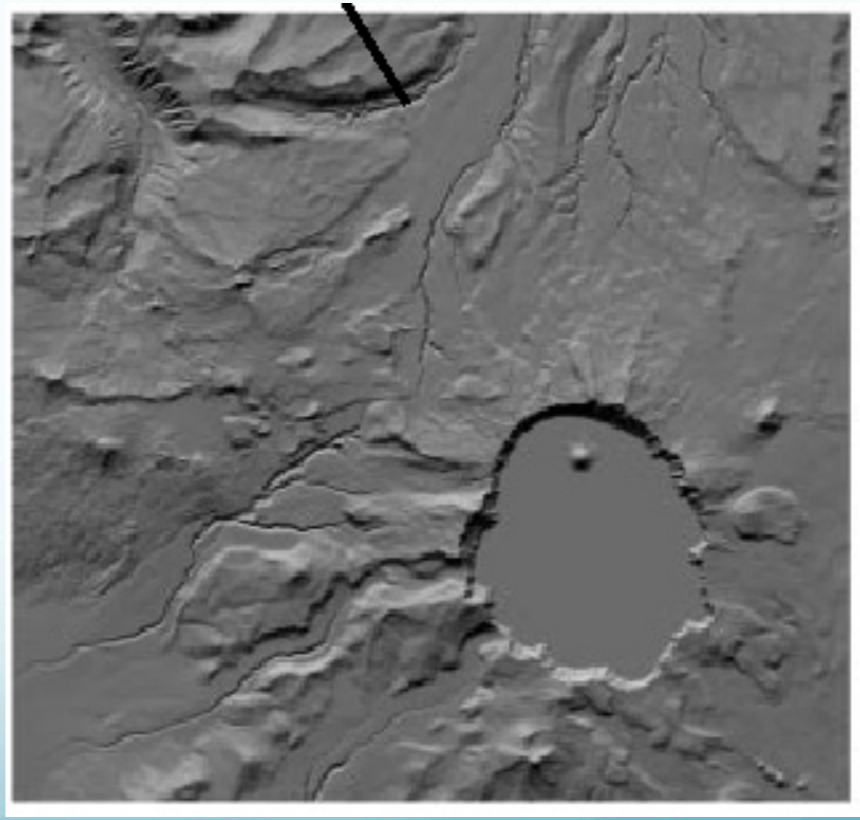
It is very hard, however, for the brain to interpret this view of the data.

What is the object in this figure?

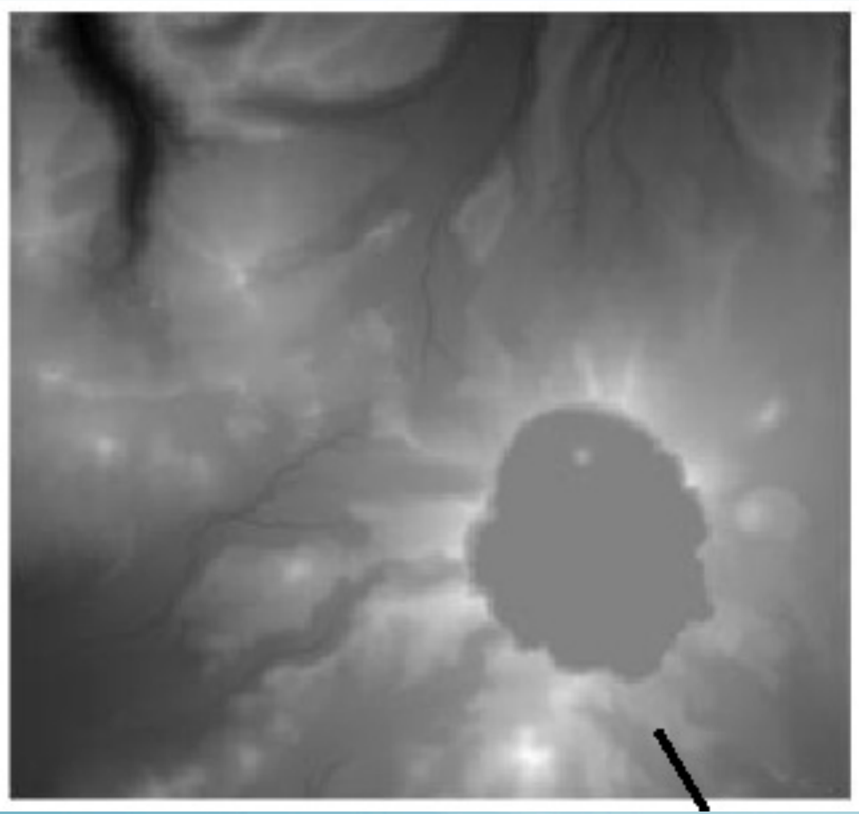
One needs to add shadows (shading) for the brain to "get the picture".

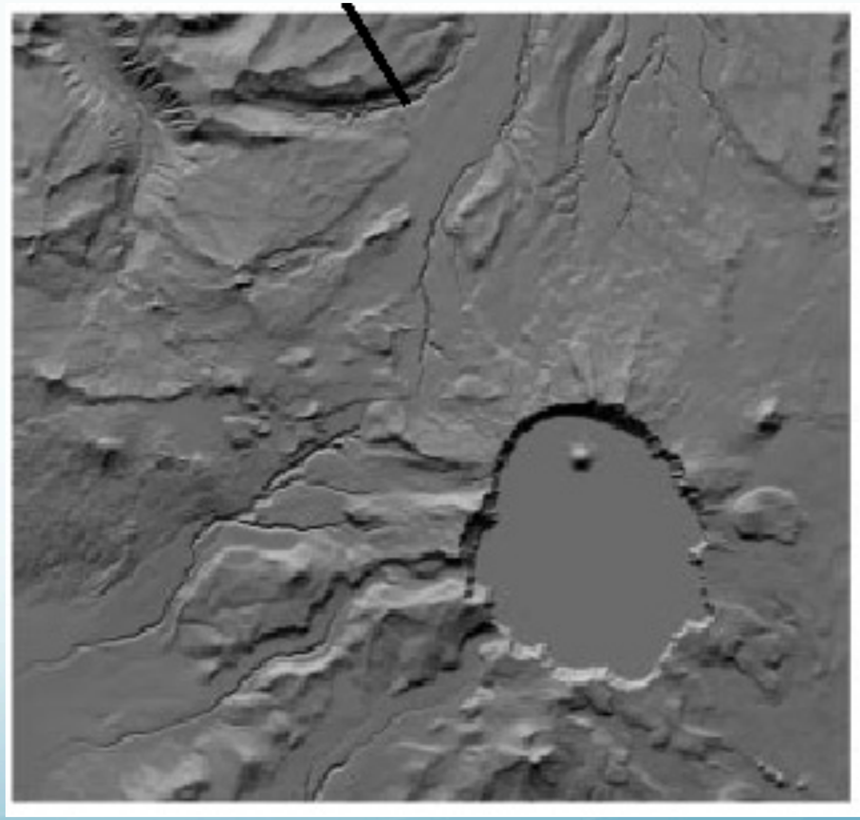
What does it look like now?





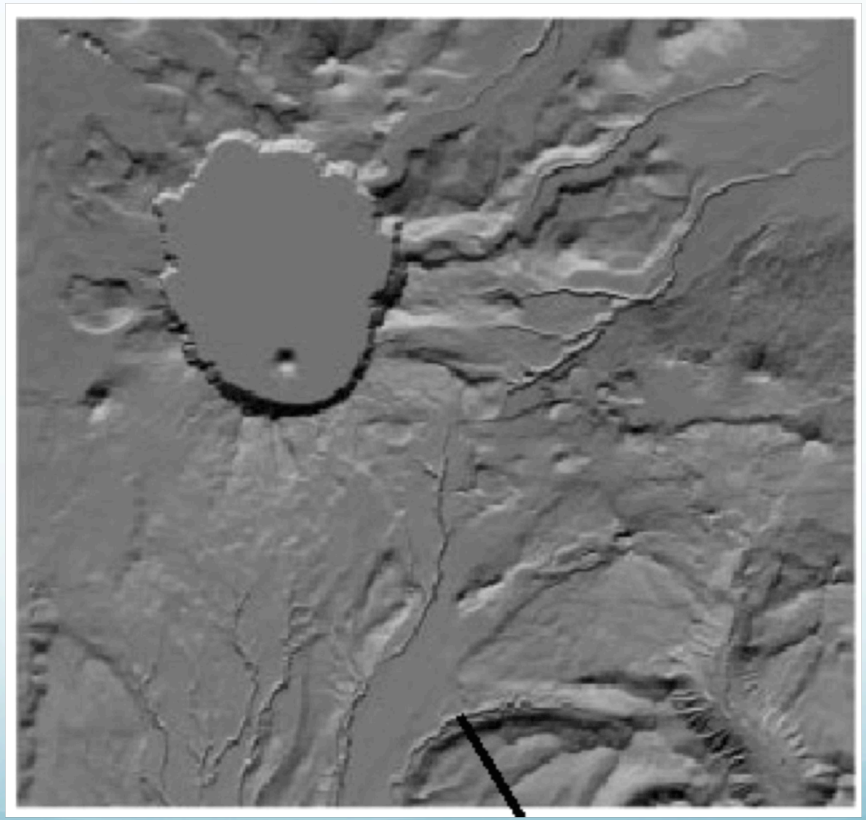
We "illuminate" the topography with a fake sun, specifying elevation and azimuth, and generate an intensity filter to be added to the color or grayscale image (actually for grayscale just use the intensity filter).



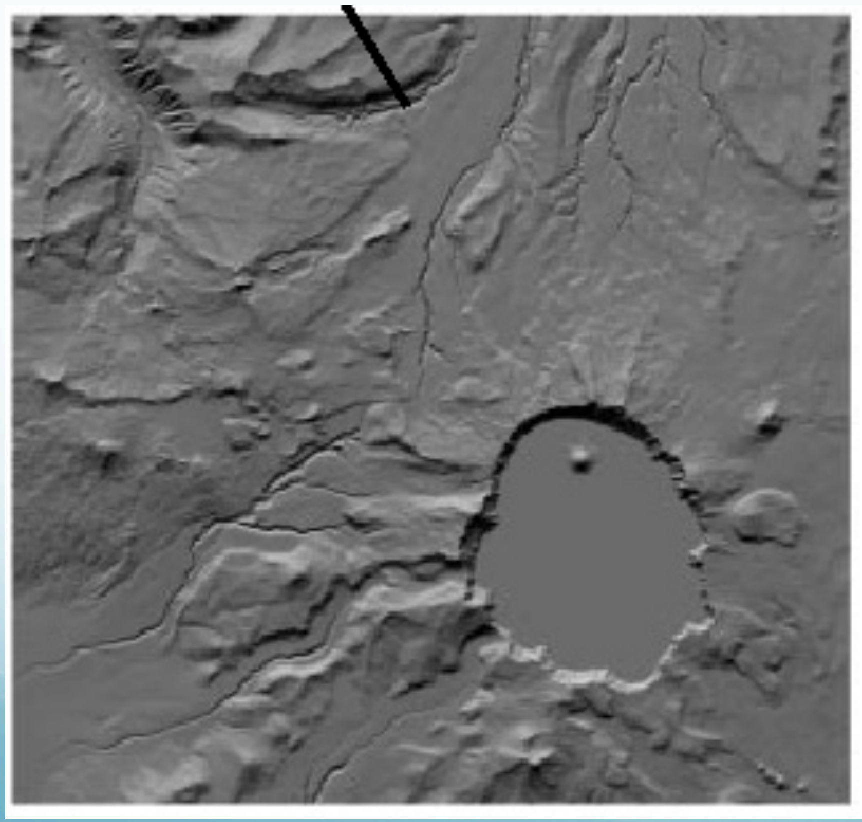


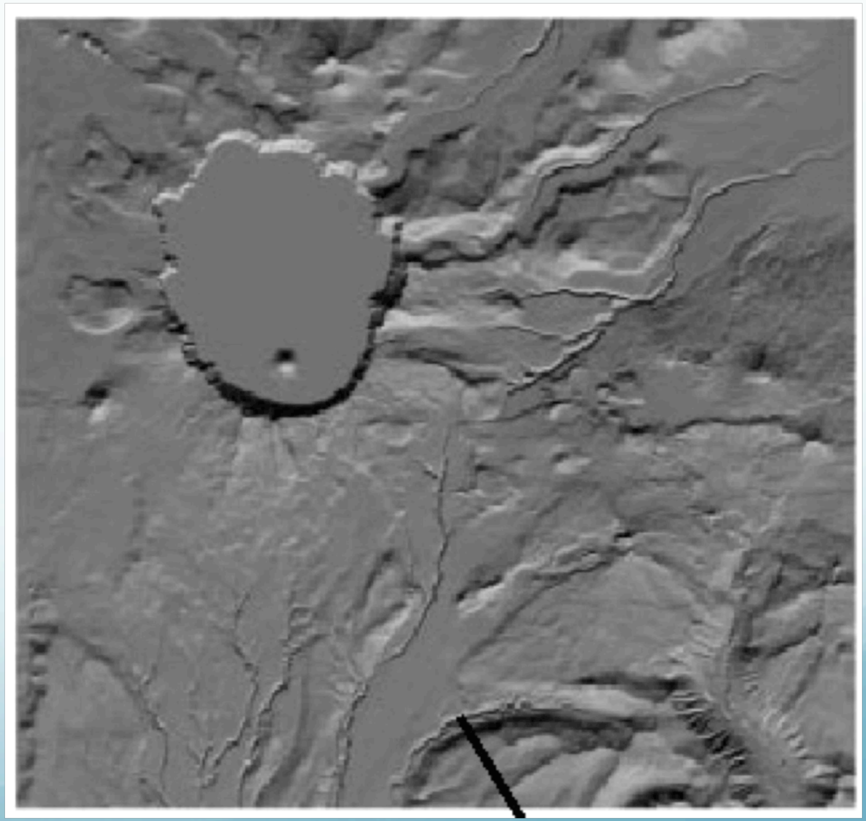
There is a slight problem however.

What is the object in this image?

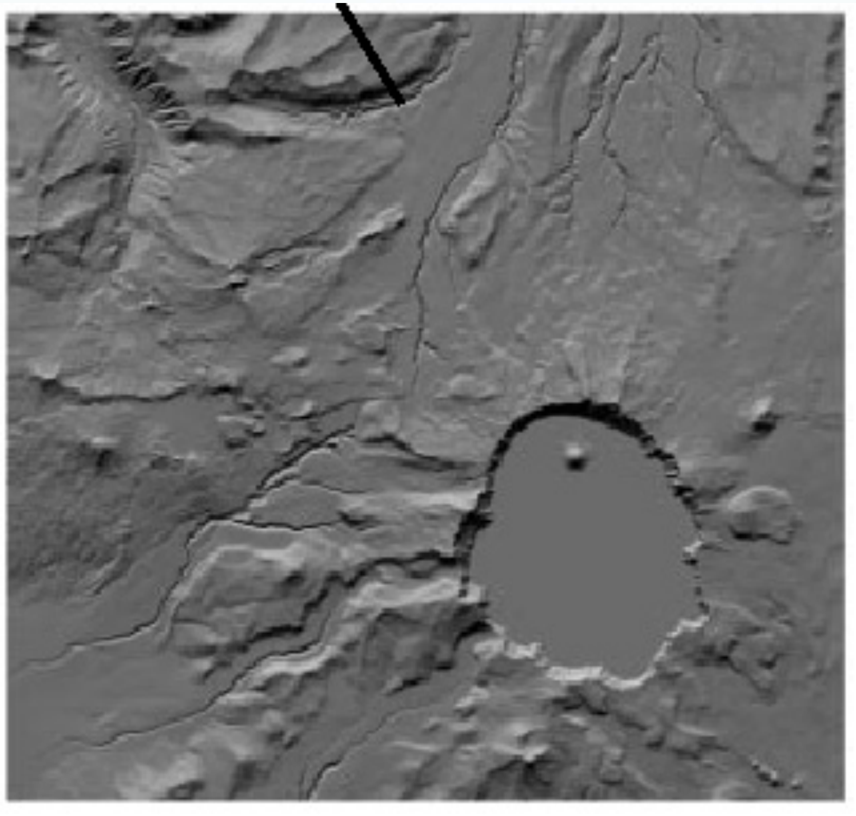


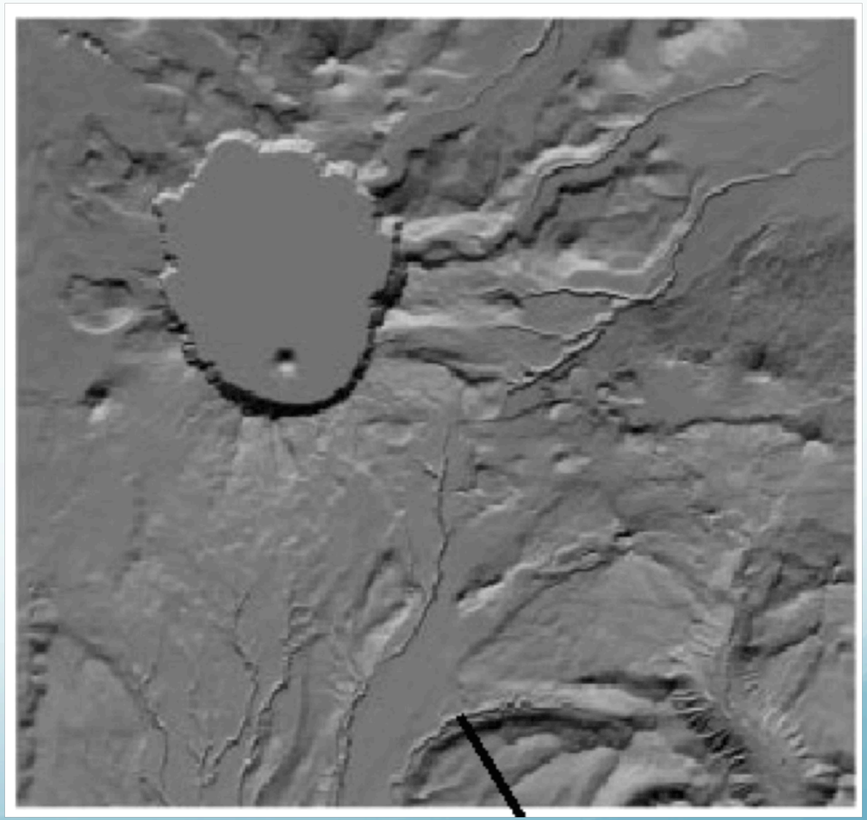
Both images are made from the same data (the one on the right is the one on the left rotated 180°, or vice versa), yet they look like very different objects (to most people – some people claim to see two of the same thing, and with the correct interpretation, i.e. the object on the earth)





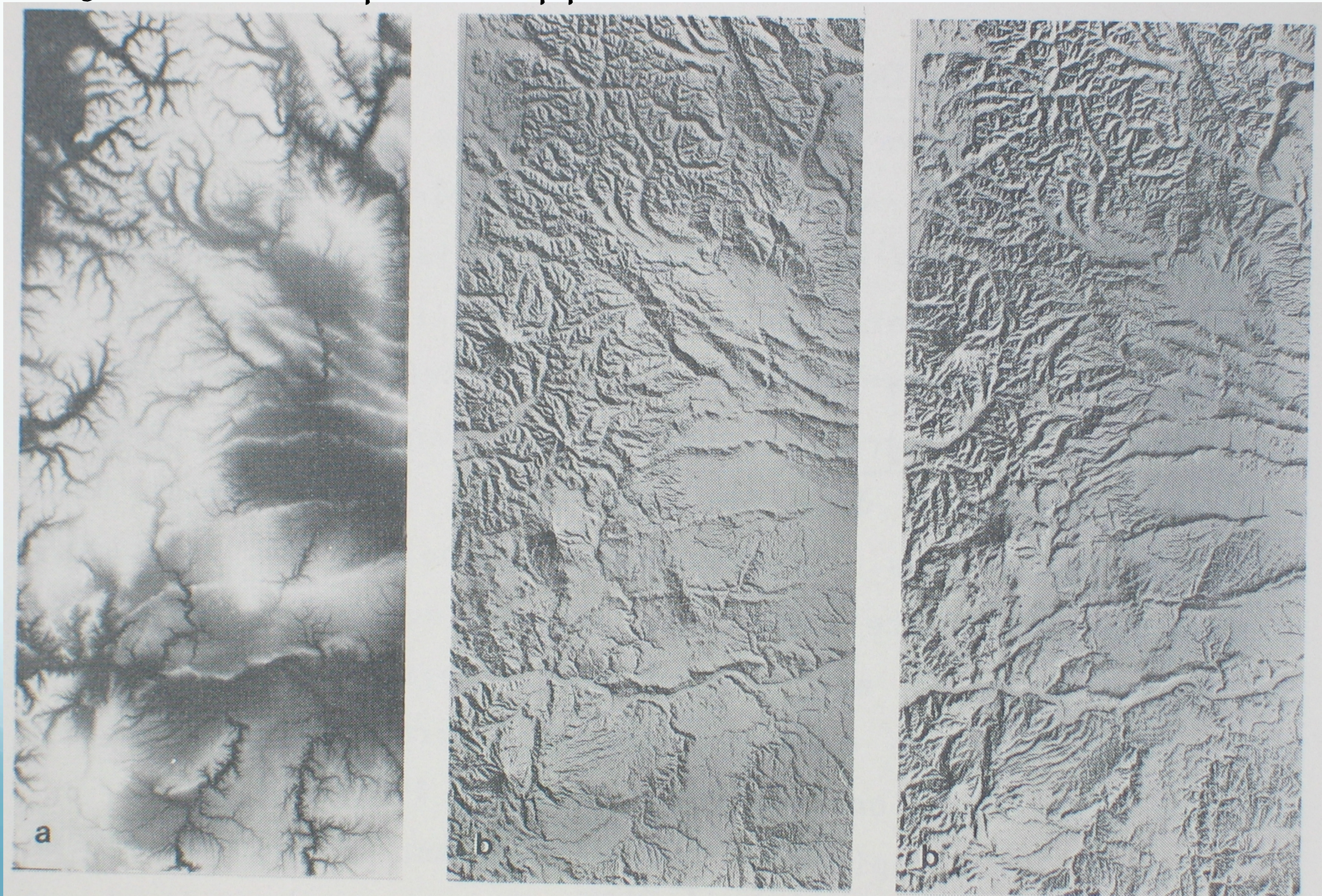
Is it a flat bottomed valley with a peak or a mesa with a small valley?



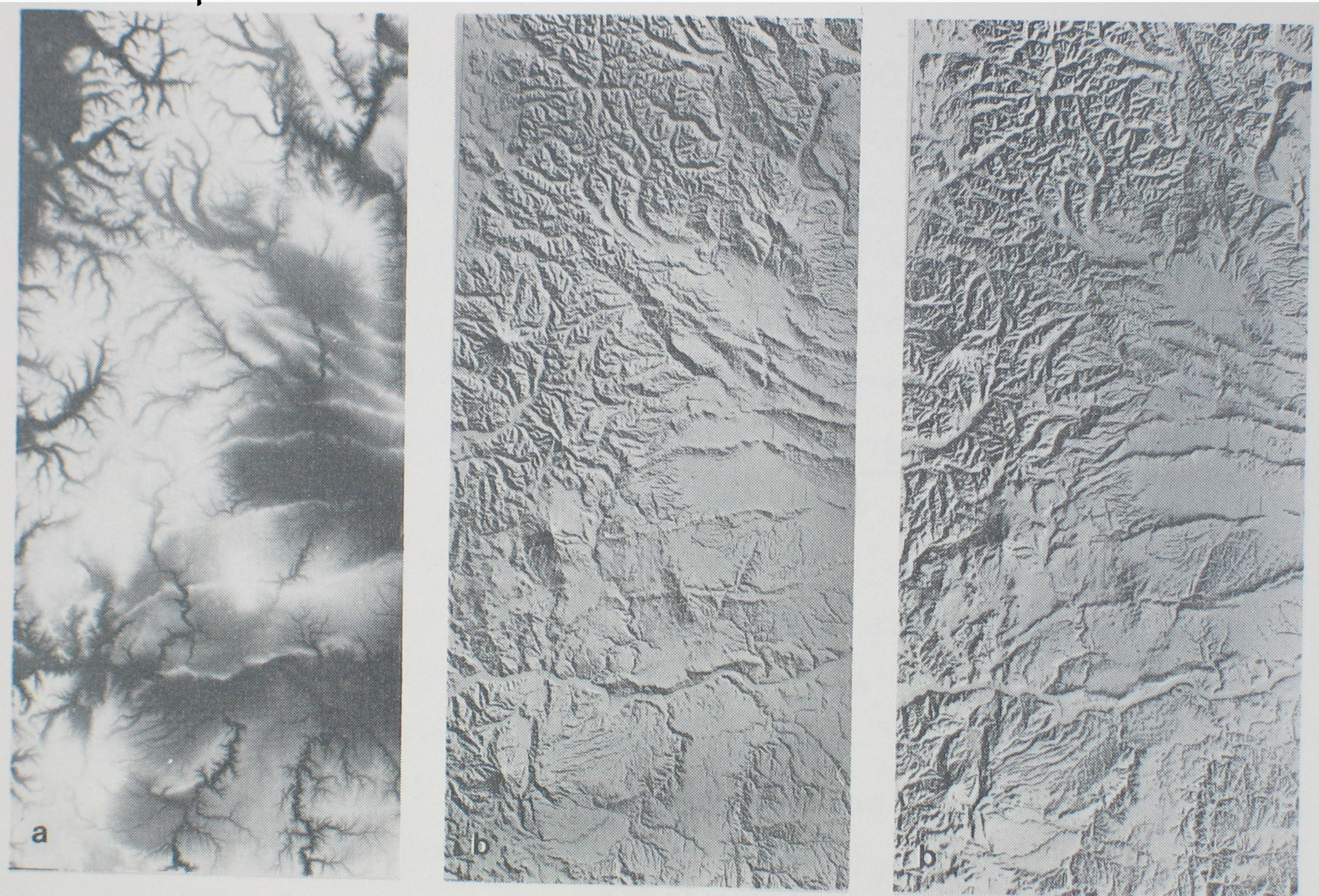


“Raw” data grey scale into topo

shaded topo “illuminated” from upper right lower right

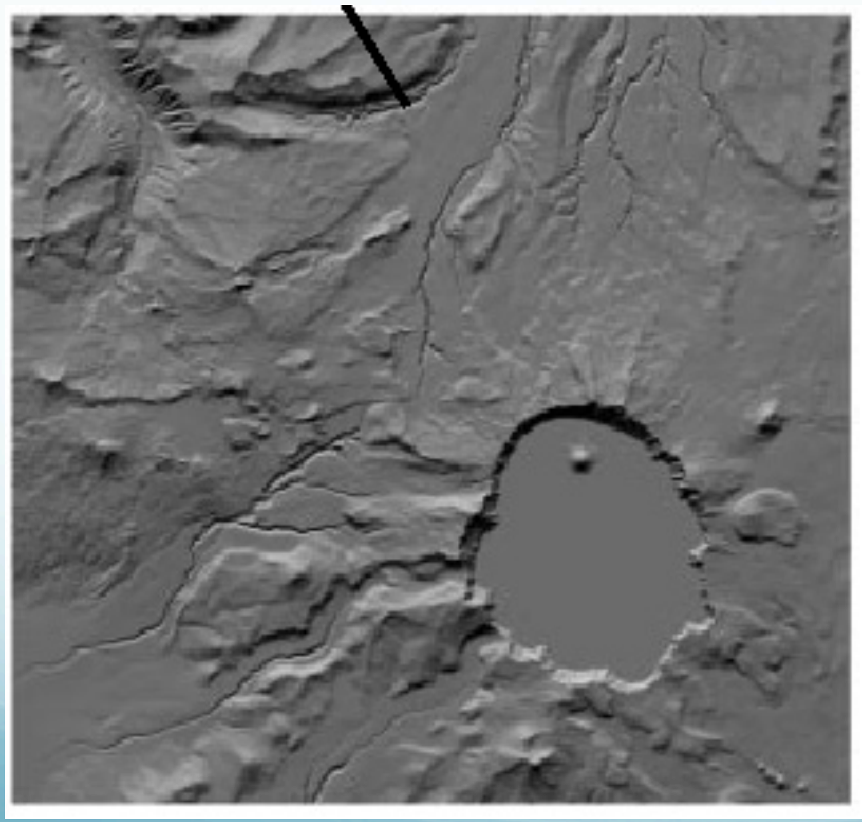


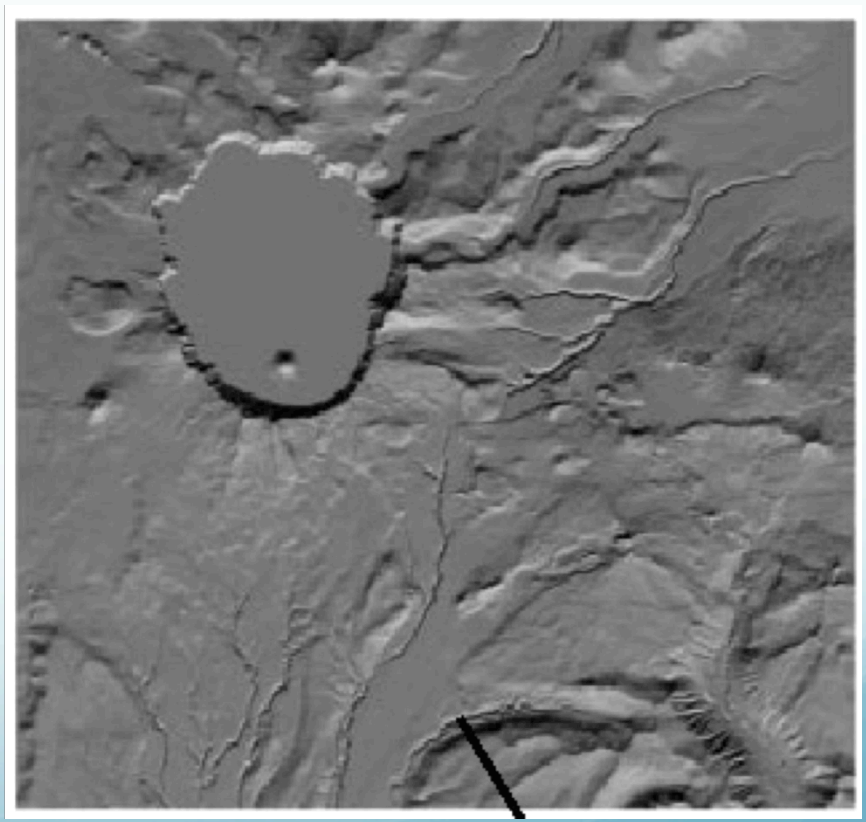
The left is an image of the data (altitude), two on the right are nice visual pictures but do not show the altitude.



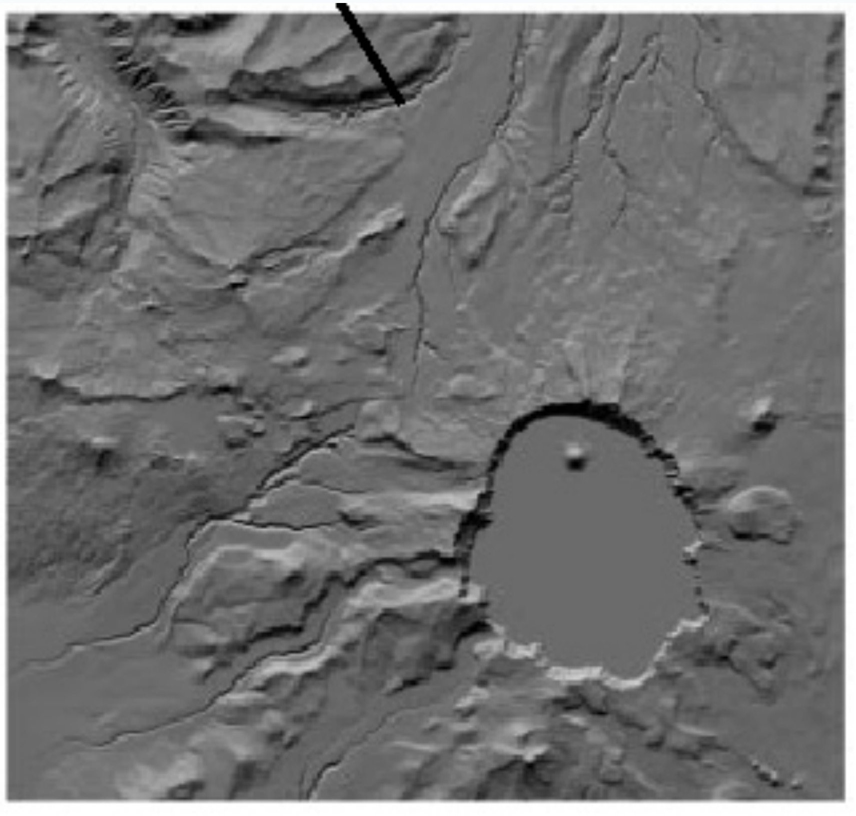
What you "see" depends on how the brain works – how the brain interprets up/down by shadows.

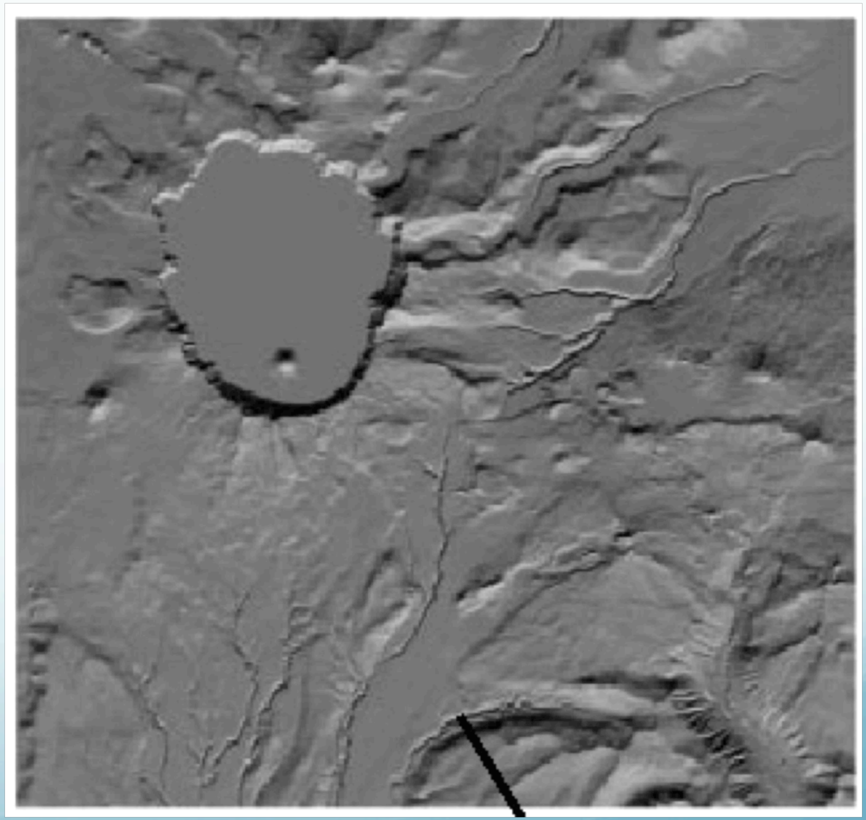
Light usually comes from "above" so the brain in



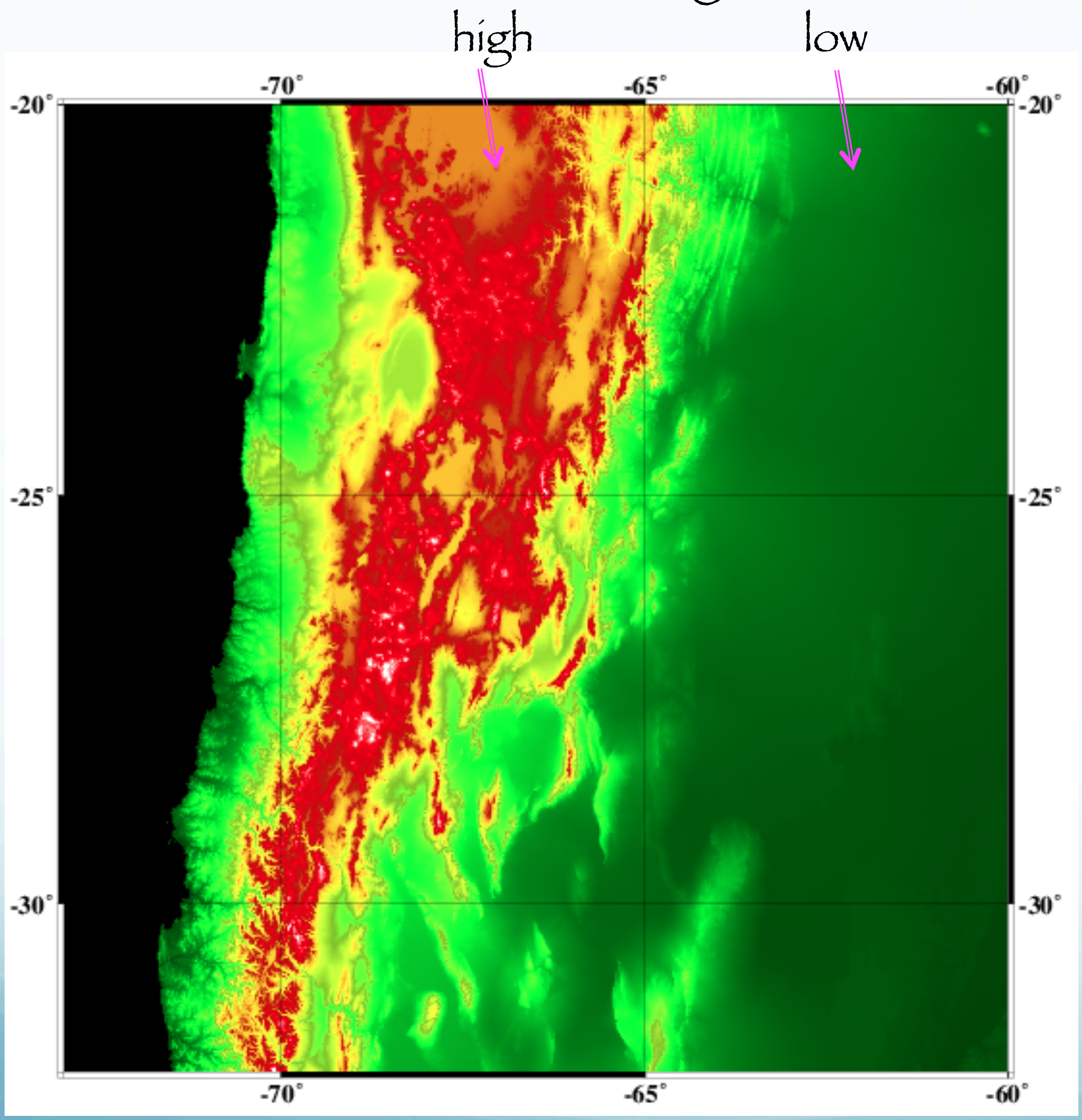


Once we started walking around, light usually comes from "above" so the brain interprets topography using this assumption to interpret what makes shadows.





Try with color



No better.

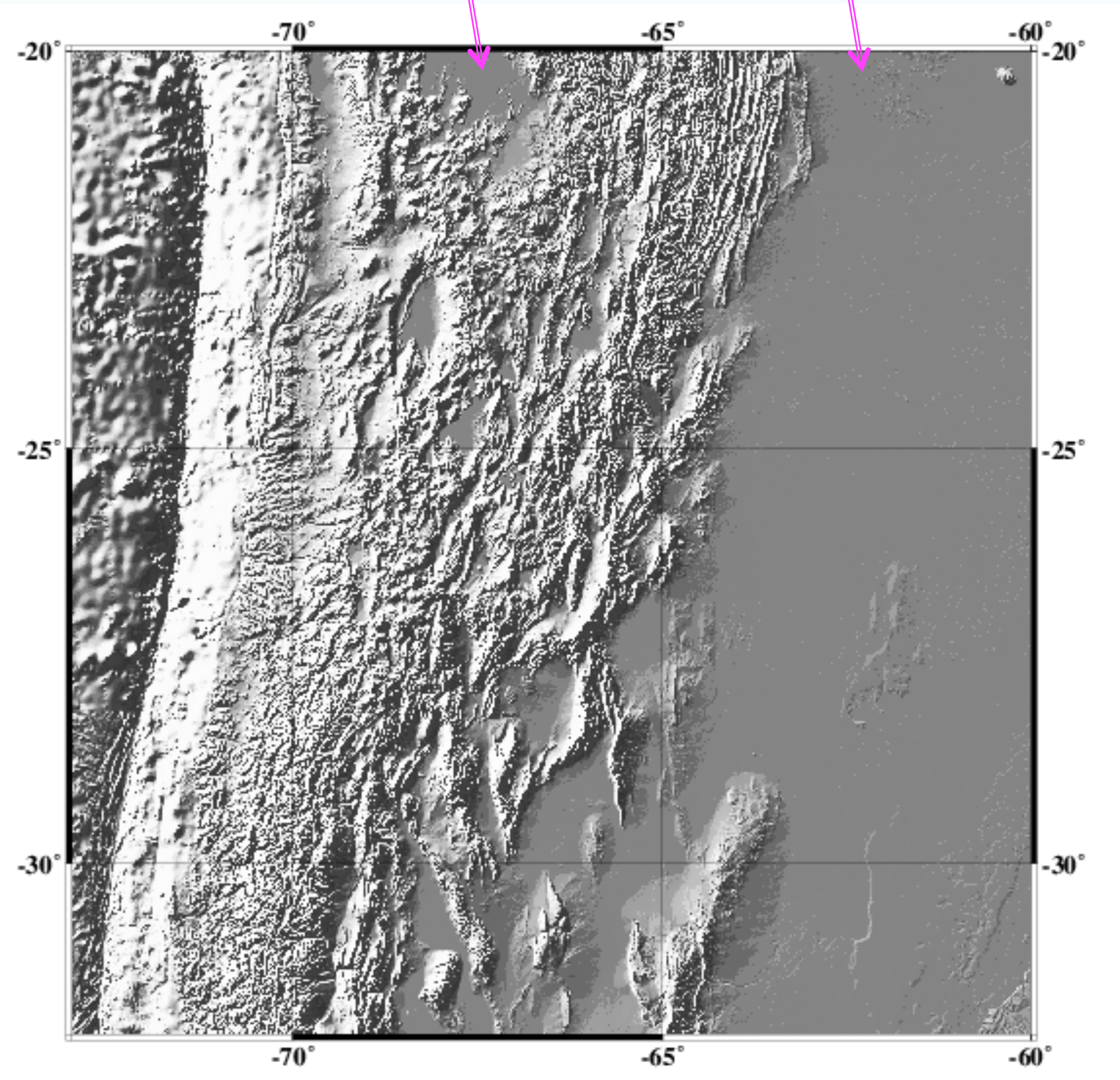
Red – high
Green - low

Again, it is very hard for the brain to interpret this view of the data.

Try with illumination

flat

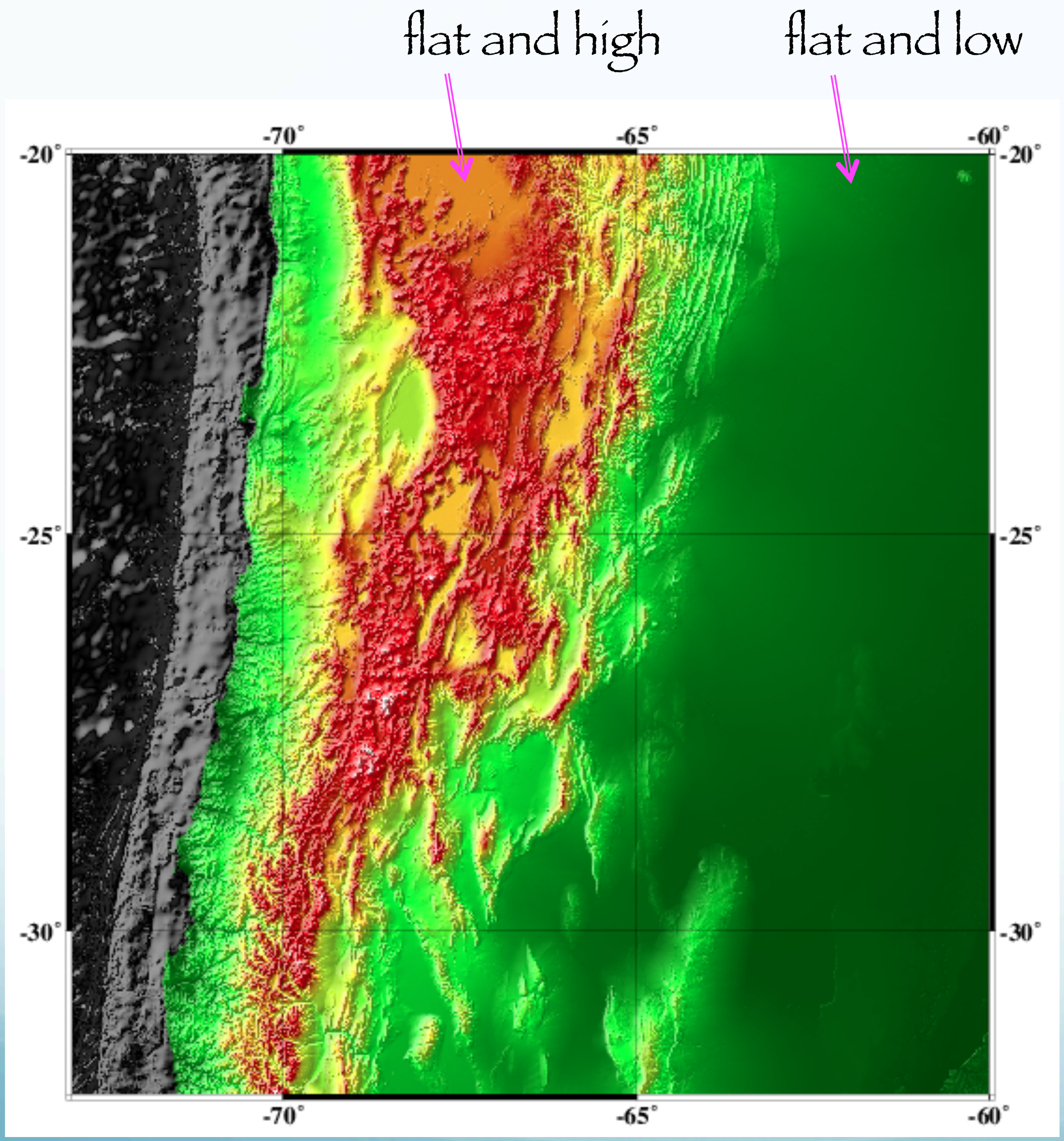
flat



A little better?

Can see slopes/ structure (mountains) but have lost altitude info.

Combine color and illumination



This is the best we can do.

The code that made the last 3 plots

```
 DATASET=9
GRDRASTERREGION=$REGION
DATAGRID=-I30c/30c
echo do 30s topo from GRDraster, dataset $DATASET
echo gmt topo data at $GMT_GRIDDIR
grdraster $DATASET -Gtopo.grd $DATAGRID -R$GRDRASTERREGION $PROJ $VBSE
grdinfo topo.grd
ls -l topo.grd
echo done with grdraster call
TOPOILLUM=315
grd2cpt topo.intns -Cgray $VBSE > bw.cpt
grdgradient topo.grd -A$TOPOILLUM -Gtopo.intns -Ne0.6 -V
#grey sca.e only
#  grdimage topo.intns -Itopo.intns -Cbw.cpt -R$REGION $PROJ -B5g5 $VBSE
$ORIENT > $OUTFILE
#color only
#  grdimage topo.grd -CapproxBryan.cpt -R$REGION $PROJ -B5g5 $ORIENT
$VBSE > $OUTFILE
#both together
  grdimage topo.grd -Itopo.intns -CapproxBryan.cpt -R$REGION $PROJ -B5g5
$VBSE $ORIENT > $OUTFILE
```

Back to making figures

GMT has a routine to do the shading: `grdgradient`.

I’ll also illuminate the ocean floor and the topography from slightly different angles – to bring out the “best” of both.

After generating the illumination, we have to combine the two files using `grdmath`.

I'll name the output files wilh `.intns` as extension.

```
NORM=-Nt
BATHILLUM=270
TOPOILLUM=315
grdgradient ${ROOTNAME}_topo.grd -A$TOPOILLUM \
-G${ROOTNAME}_topo.intns $NORM -V

grdgradient ${ROOTNAME}_30stopo.grd -A$BATHILLUM \
-G${ROOTNAME}_30stopo.intns $NORM -V

grdmath -F -V ${ROOTNAME}_topo.intns ${ROOTNAME}_30stopo.intns AND = \
${ROOTNAME}_topobath.intns

INTNSFILE=${ROOTNAME}_topobath
```

So now we have two grid files

- One with the topographic data
- One with the shading

Now we're ready to plot them together to make the map.

Finally we make our first contribution to the map (PostScript output file) using `grdimage`.

`grdimage` can combine the coloring of the data, based on the CPT (color something table) file, with the shading (which comes from the slopes of the data).

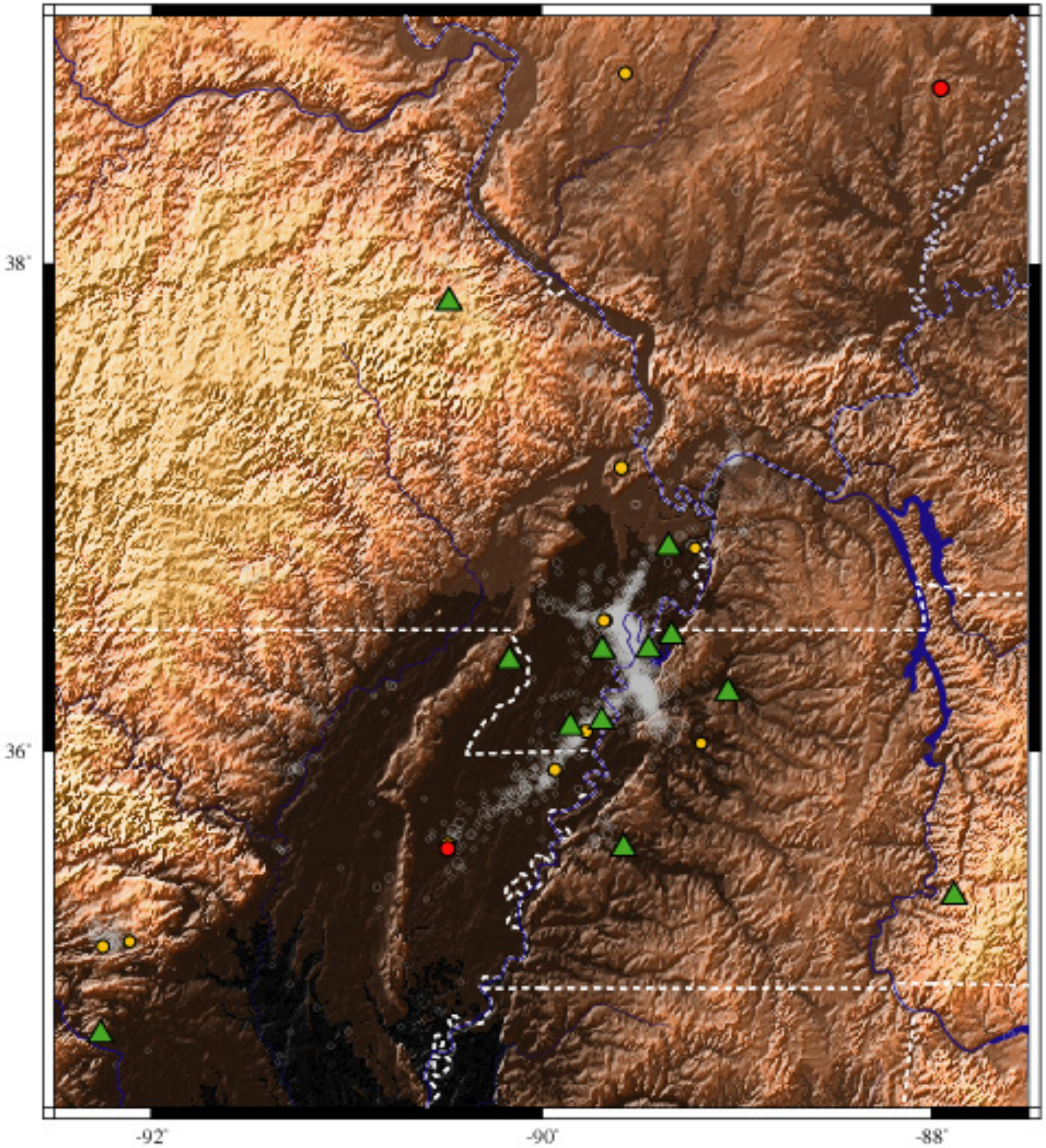
grdimage can combine the coloring of the data, based on the CPT file, with the shading (which comes from the slopes of the data).

```
echo color topo
CPTFILE=/gaia/opt/gmt/share/GMT_globe.cpt
grdimage $INTNSFILE.grd -I$INTNSFILE.intns -C$CPTFILE -R$REGION -$PROJ
$SCALE $GRID -K -X$XOFFSET -Y$YOFFSET -V $ORIENT > $OUTPUTFILE
echo done with color topo
```

The CPT file is the color table file. GMT has a bunch of them predefined (look in the directory referenced above).

GMT uses the R/G/B model for color

You can also make your own CPT files
(if you have lots of time) or rescale existing ones based on your data.



“copper” built-in cpt file

Now we can add other data – earthquakes, GPS vectors, focal mechanisms, etc.

```
psmeca -R -$PROJ$SCALE -Sd0.2/0/0 -G$RED $CONTINUE -L -W0.5/$BLACK \
india.cmt >> $OUTPUTFILE
```

Again being lazy, I don't like to have to keep track of the last GMT call (to keep track of whether or not I need the –O) so I use `$CONTINUE`.

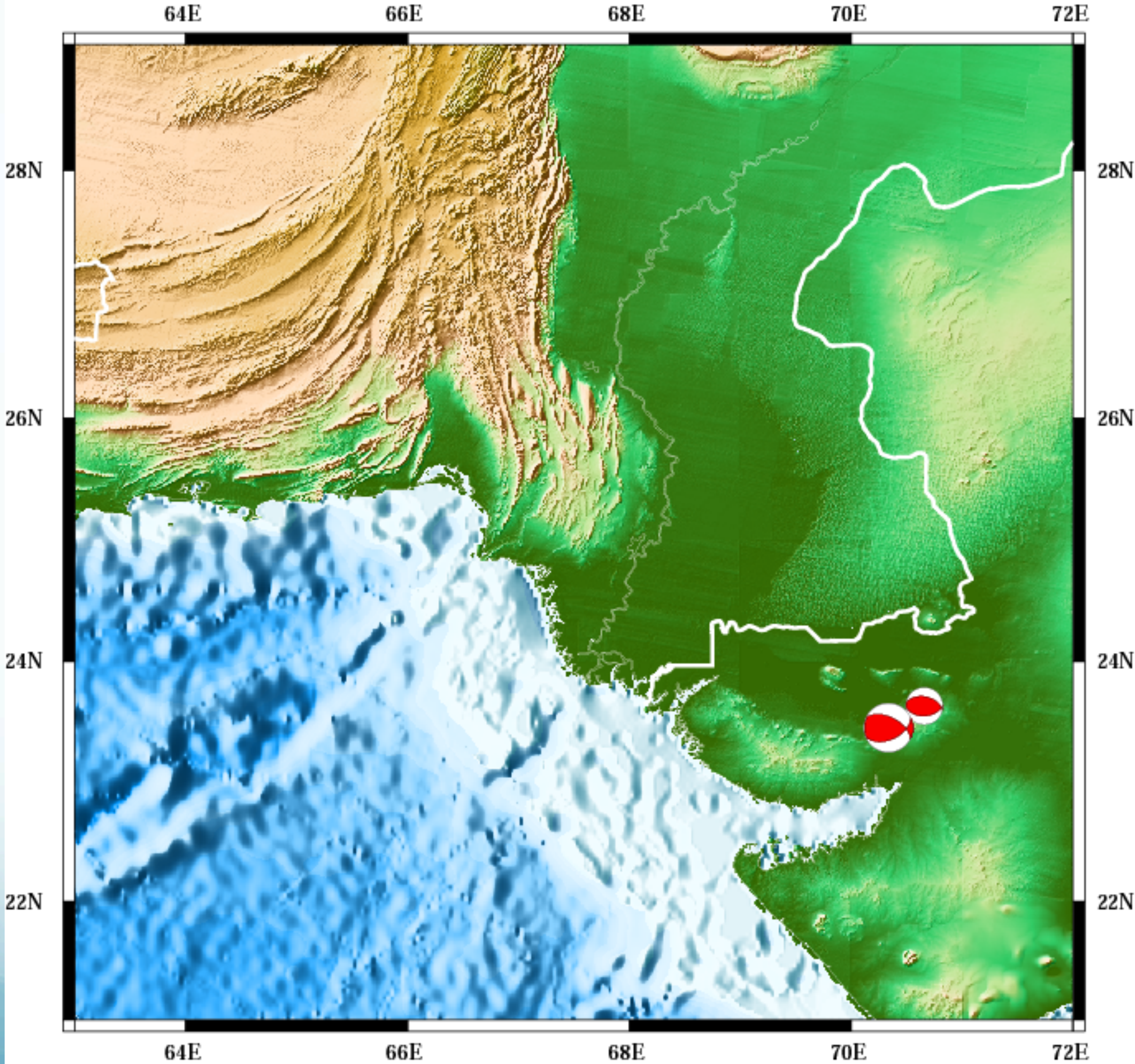
Then I check the output file for a **showpage** when I'm done – and write the PostScript myself when I need it.

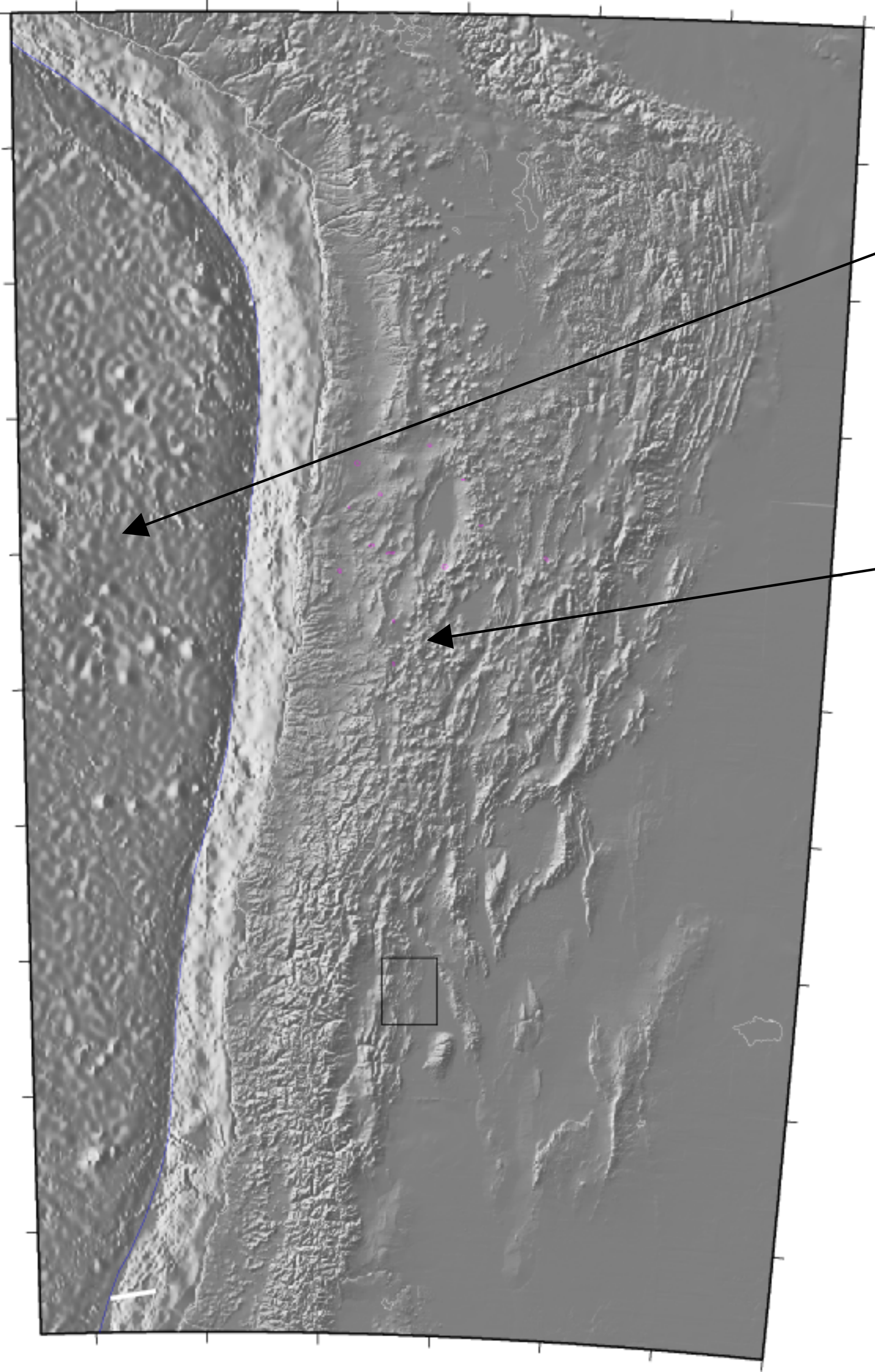
```
echo done with figure - clean up
SHOWPAGE=`tail -1 $OUTPUTFILE | nawk '{print $1}'`
echo check SHOWPAGE -${SHOWPAGE}-
if [ $SHOWPAGE != showpage ]
then
 echo add showpage
 echo showpage >> $OUTPUTFILE
fi
```

We then have to erase all the temporary files we made.

```
if [ $CLEAN = yes ]
then
 echo yes - clean up
 if [ $TOPO != notopo ]
 then

  \rm ${ROOTNAME}.cpt
  \rm ${ROOTNAME}.grd
  \rm ${ROOTNAME}.intns

  \rm ${ROOTNAME}_topo.grd
  \rm ${ROOTNAME}_topo.intns
  \rm ${ROOTNAME}_2mtopo.grd
  \rm ${ROOTNAME}_2mtopo.intns
  \rm ${ROOTNAME}_30stopo.grd
  \rm ${ROOTNAME}_30stopo.intns
  \rm ${ROOTNAME}_topobath.grd
  \rm ${ROOTNAME}_topobath.intns

 fi

 \rm ${ROOTNAME}.nawk
 \rm ${ROOTNAME}.tmp

fi
```

So here's our pretty MAP!





ETOPO -5

global

(5 min)

GTOPO-30

Land only

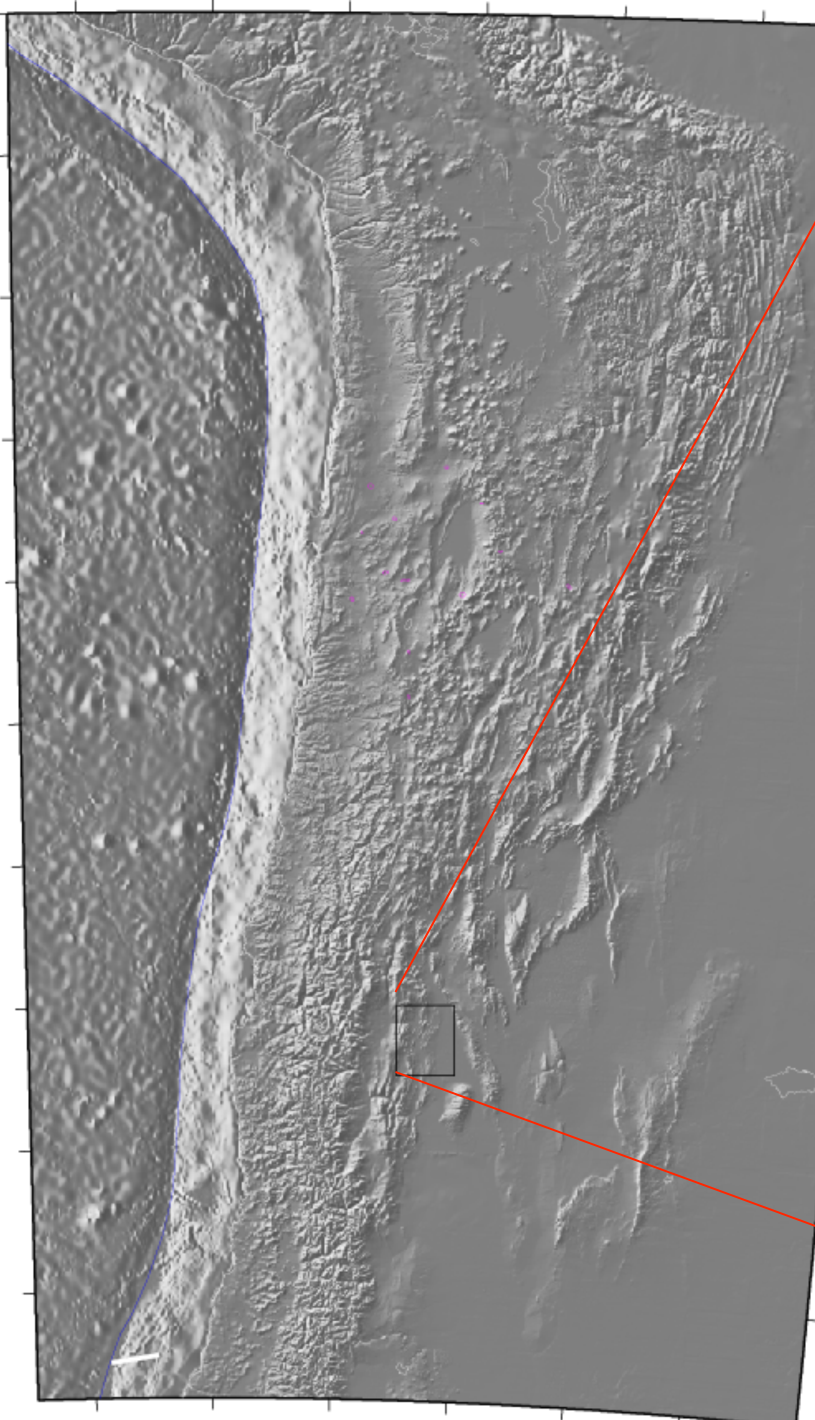
(30 sec)

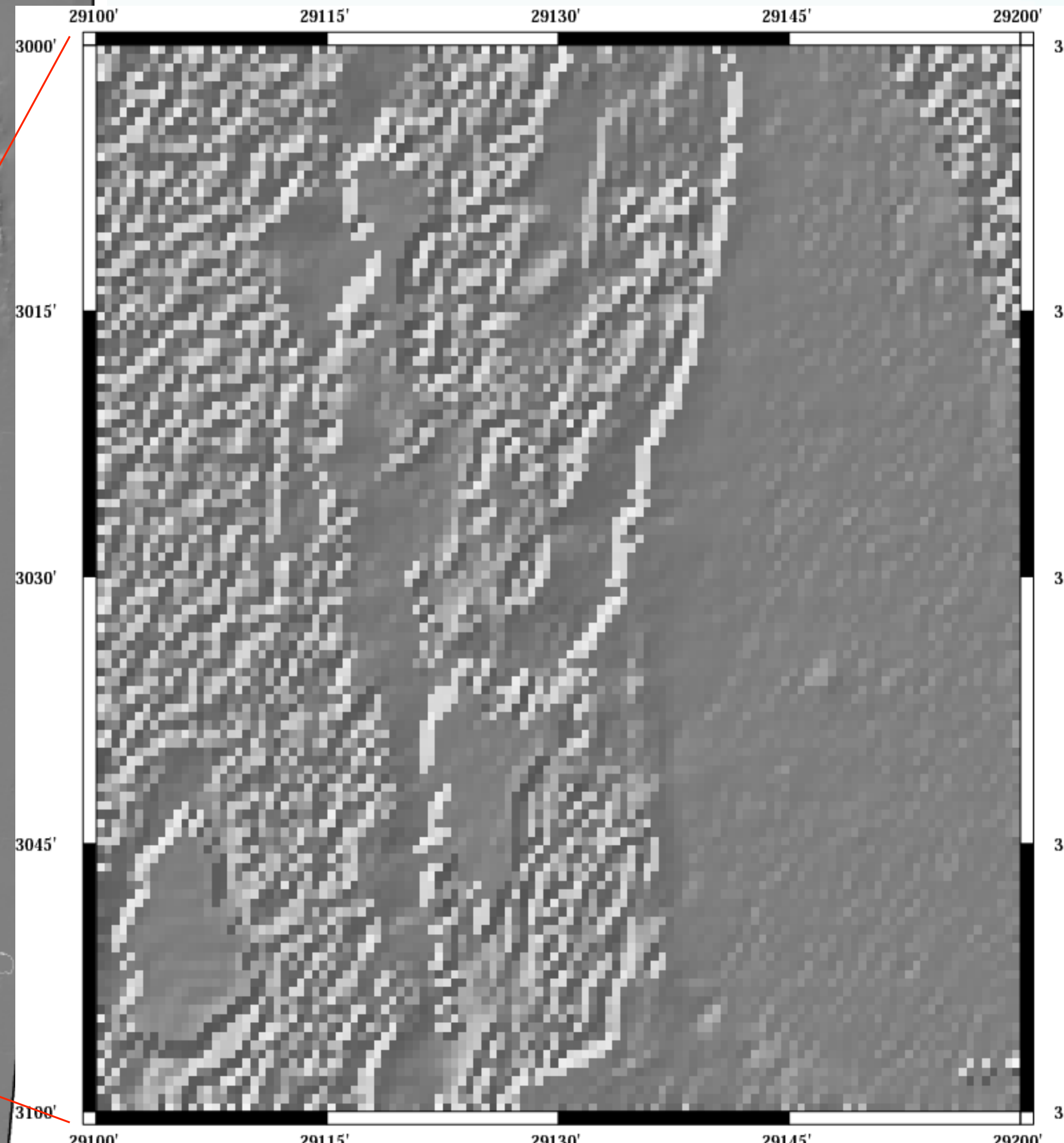
(These were combined using technique presented)

SRTM

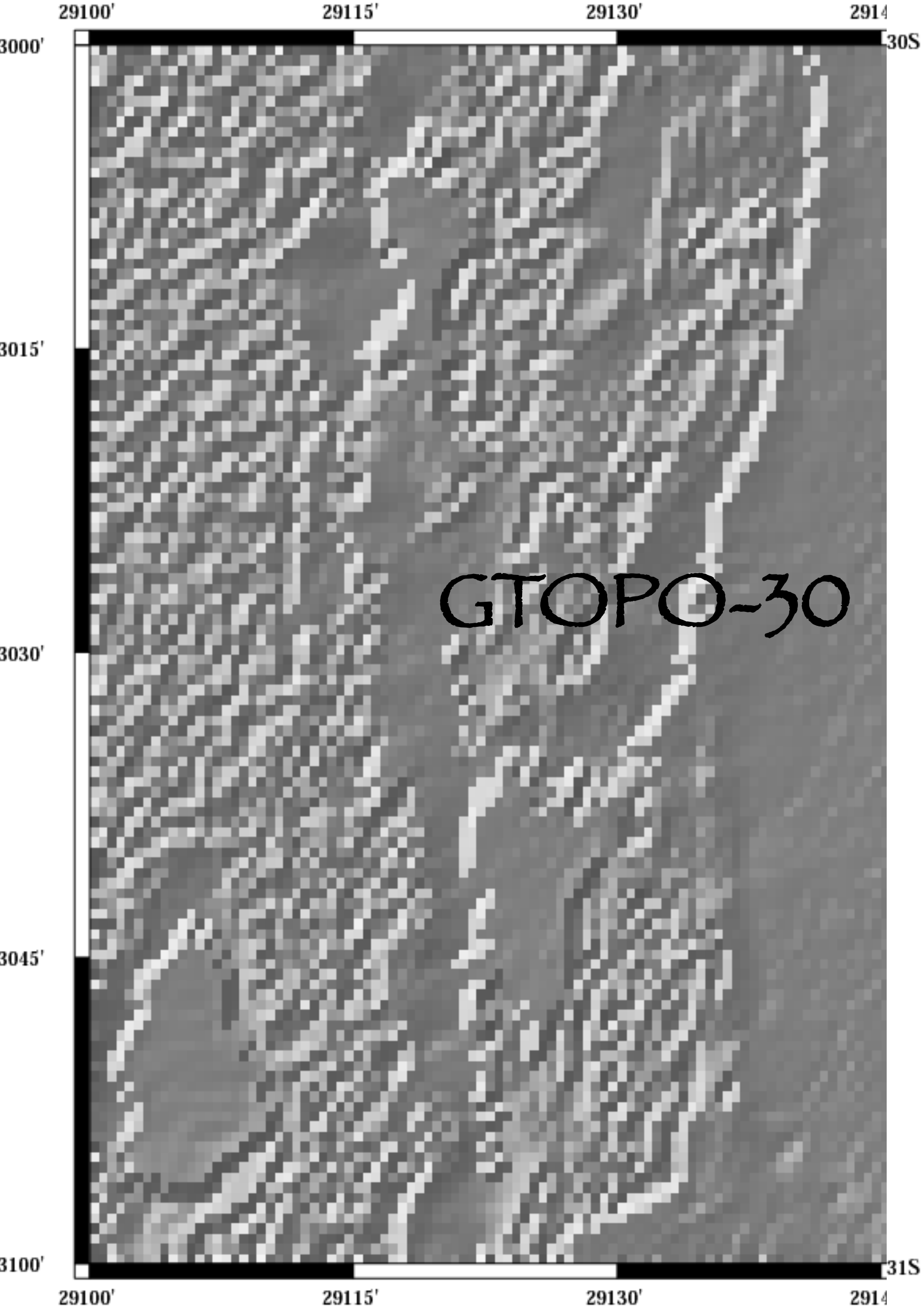
Land only

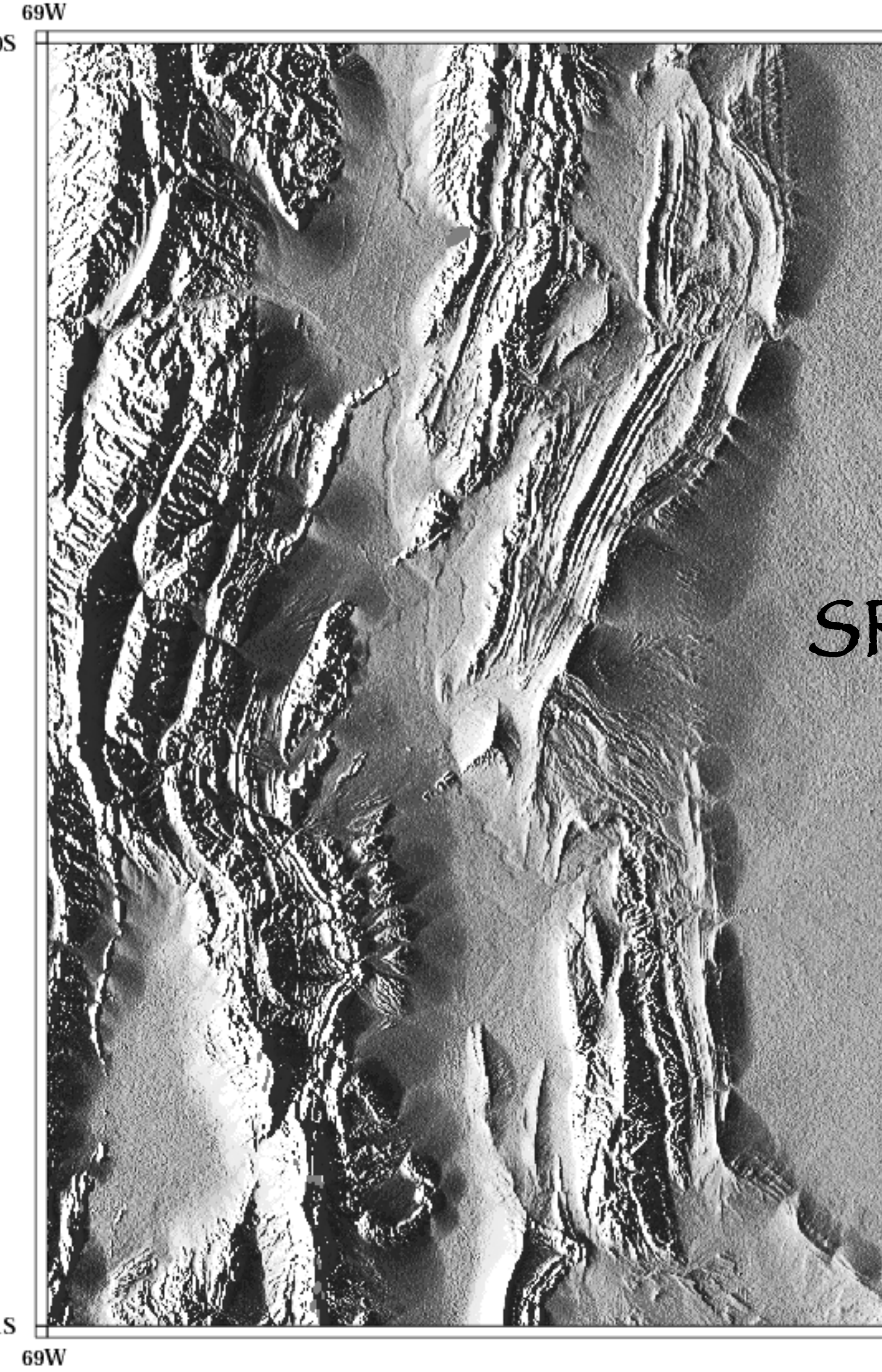
(3 sec)





GTOPO-30


29100'
29115'
29130'
2914
3000'
3015'
3030'
3045'
3100'
30S
31S
GTOPO-30


69W
69W

Plotting a single srtm file

```
#!/bin/sh
\rm tst.grd
grdgradient tile_31_69.grd -A270 -Gtst.intens -Ne0.6 -V \
grd2cpt tst.intens -Cgray > $0.cpt
grdimage tst.intens -Itst.intens -R-69/-68/-31/-30 -Jm7 \
-B1g1a -P -C$0.cpt > $0.ps
```

Plotting multiple 1x1 degree tiles possible, but more slightly complicated (see me).

I can't get SRTM data into `grdraster` format input file (any volunteers?)

General GMT shell script will look something like this

Call to set up base map – this may or may not plot any data

Series of GMT calls to add various kinds of data

Last GMT call “closes” file

Majority of work is in manipulating the data files using all the standard UNIX tools.

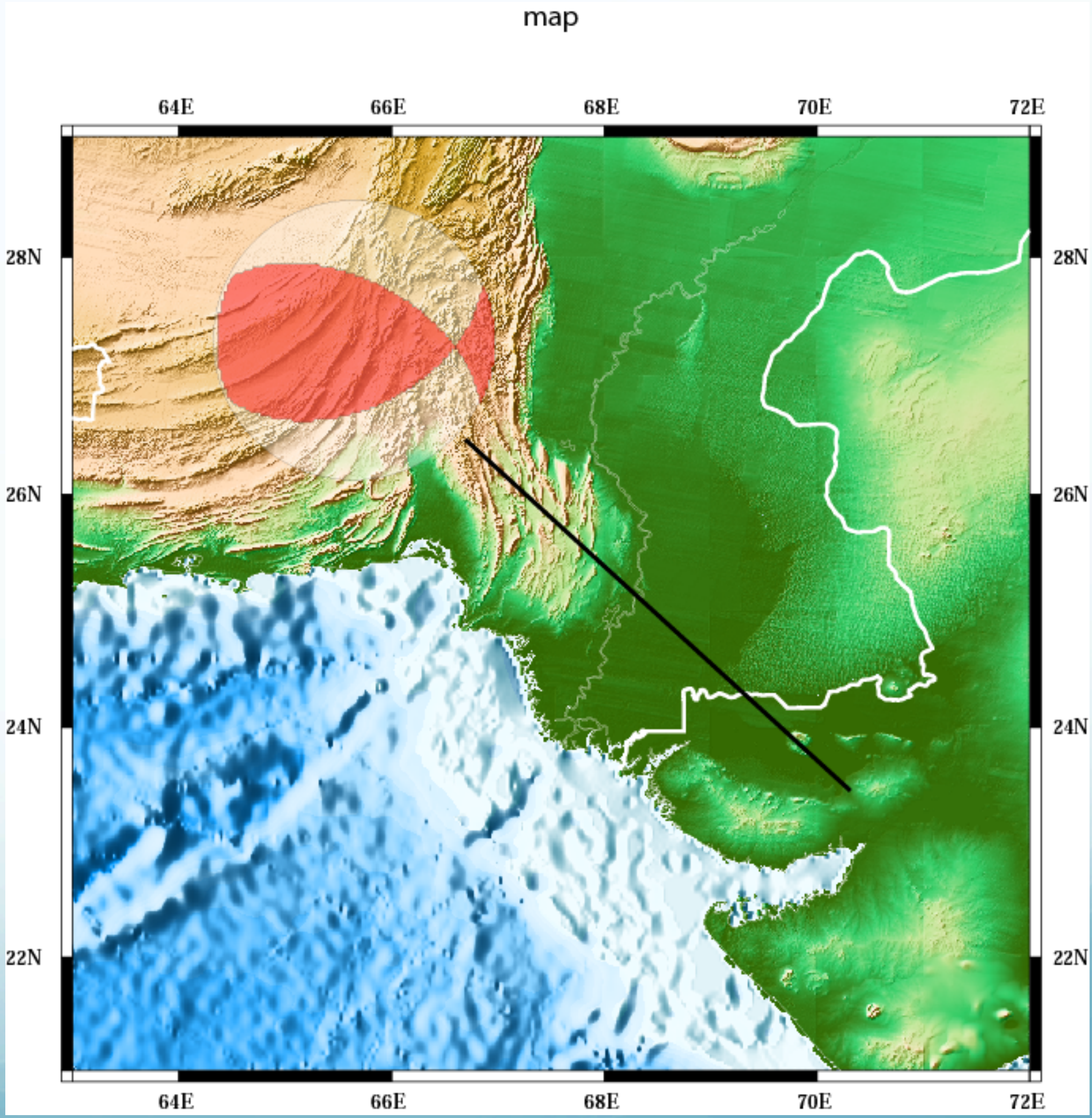
Finally, you can put the finishing touches on your figure with Adobe Illustrator (which works with PostScript files)

Change line thicknesses, types (dash, etc.), fill colors; annotate; etc.

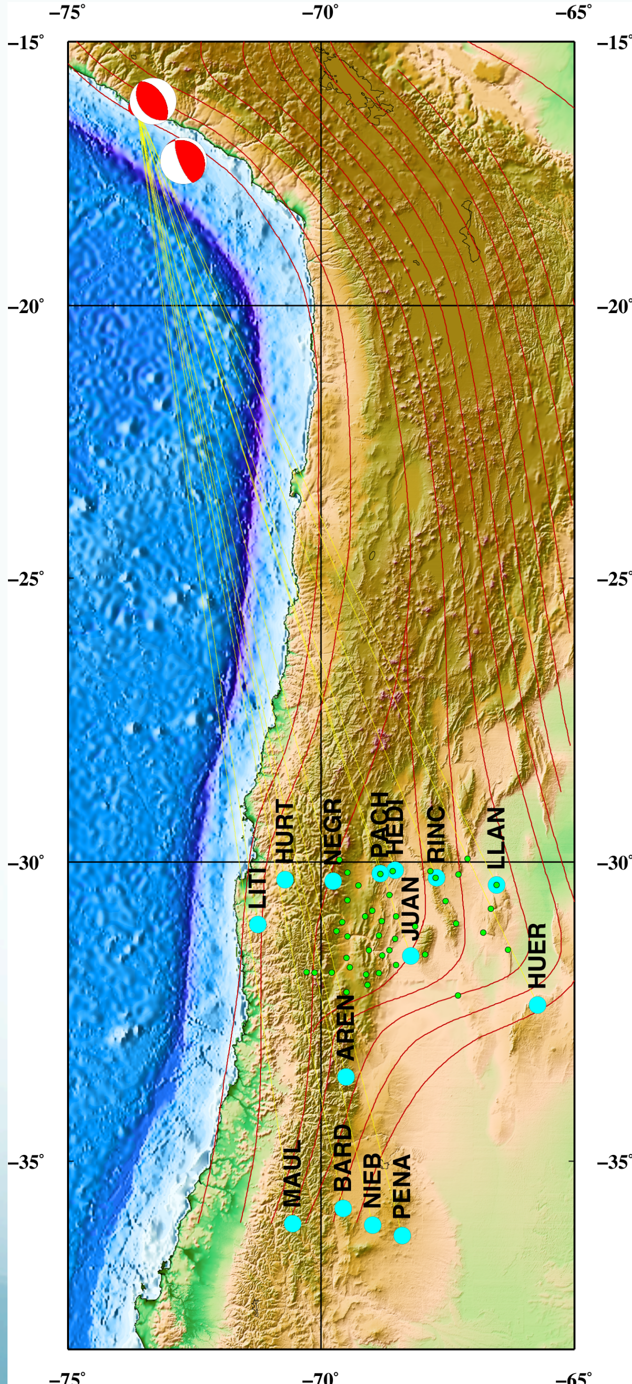
Make focal mechanism transparent.

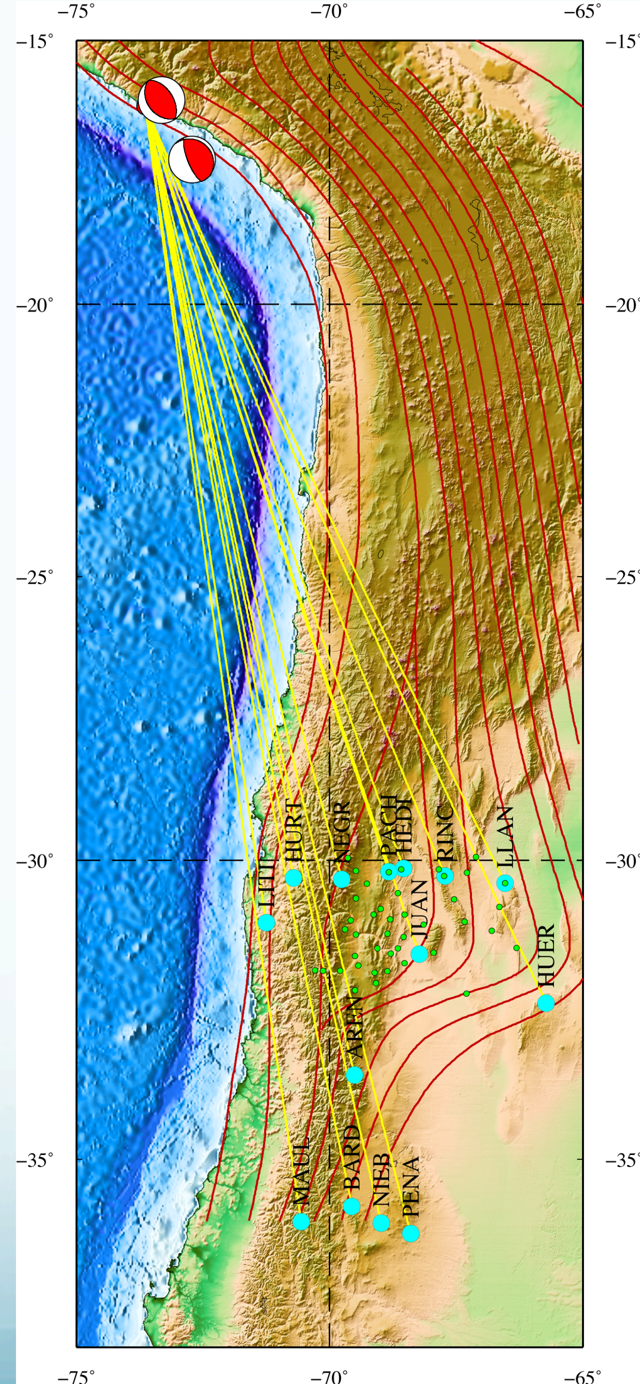
Paste in other stuff.

Lots well documented problems going over to Adobe – principally with annotation/text.



Before edit with Illustrator





After edit with Illustrator

Creating a map

gmtset: change individual GMT default parameters

(grdimage: plot topography)

pscoast: Plot coastlines, filled continents, rivers, political borders (, map border).

psxy: Plot symbols, polygons, and lines in 2-D

pstext: Plot text strings

psmeca: Plot focal mechanisms

psvelo: plot gps velocity vectors

Setup

```
#!/bin/bash

ROOT=$HOME/unixside
GEODATA=$ROOT/geolfigs
SAMDATA=$ROOT/geolfigs
VBSE=-V

REGION=-75/-65/-38/-15
PROJ=-Jm0.9
```

WHITE=255
DKGRAY=64
LTGRAY=192
VLTGRAY=225
EXTGRAY=250
GRAY=128
BLACK=0
BLACKP1=1
BLACKP2=2
BLACKP3=3
BLACKP4=4
WHITEM1=254
WHITEM2=253

RED=255/0/0
RED1=254/0/0
DKRED=196/0/0
BLUE=0/0/255
GREEN=0/255/0
LTGREEN=192/255/192
DKGREEN=0/196/0
YELLOW=255/255/0
ORANGE=255/192/0
MAGENTA=255/0/255
DKMAGENTA=181/0/223
CYAN=0/255/255
LTCYAN=196/255/255
LTBLUE=192/192/255
VLTBLUE=225/255/255
VLTBLUE=240/250/255
LTRED=255/225/225
PINK=255/225/255
BROWN=160/64/32
LTBROWN=224/128/96
REDBROWN=255/96/64
VLTBROWN=229/225/209
MUDBLUE=193/213/232

```
MOREPS=-K
CONTINUEPS="-K -O"
ENDPS=-O
PORTRAIT=-P
OUTFILE=$0.ps
```

Get Bathymetry

```
GRDRASTERREGION=$REGION
DATASET=10
DATAGRID=-I2m/2m
grdraster $DATASET -G${ROOTNAME}_2mtopo.grd $DATAGRID -R
$GRDRASTERREGION $VBSE
```

Get Topography

```
DATASET=9
DATAGRID=-I30c/30c
grdraster $DATASET -G${ROOTNAME}_topo.grd $DATAGRID -R
$GRDRASTERREGION $VBSE
```

Illuminate topography

```
BATHILLUM=270
TOPOILLUM=315
NORM=-Nt
grdgradient ${ROOTNAME}_topo.grd -A$TOPOILLUM -G${ROOTNAME}
_topo.intns $NORM $VBSE
INTNSFILE=${ROOTNAME}_topobath
```

Resample (up/interpolate) bathymetry

```
grdsample ${ROOTNAME}_2mtopo.grd -G${ROOTNAME}_30stopo.grd
$DATAGRID -F -R$GRDRASTERREGION $VBSE
```

Illuminate resampled bathymetry

```
grdgradient ${ROOTNAME}_30stopo.grd -A$BATHILLUM -G${ROOTNAME}
_30stopo.intns $NORM $VBSE
```

Combine bathymetry and topo data sets. Have to do for both color topo and shading.

```
grdmath $VBSE ${ROOTNAME}_topo.grd ${ROOTNAME}_30stopo.grd AND
= ${ROOTNAME}_topobath.grd
grdmath $VBSE ${ROOTNAME}_topo.intns ${ROOTNAME}_30stopo.intns
AND = ${ROOTNAME}_topobath.intns
```

(see grdmath man page

```
Name     #args   Returns
-----------------------
. . .
AND 2  1 NaN if A and B == NaN, B if A == NaN, else A.
. . .
```

Select color table, some more setup, render shaded color topo. This call has all the setup info (projection, offset, orientation, etc.)

```
CPTFILE=$ROOT/dem/GMT_globe.cpt
XOFFSET=4.8
YOFFSET=3.6
grdimage $INTNSFILE.grd -I$INTNSFILE.intns -C$CPTFILE -R
$REGION $PROJ $MOREPS -X$XOFFSET -Y$YOFFSET $PORTRAIT $VBSE >
$OUTFILE
```

Draw coastline

```
pscoast -R$REGION $PROJ -B5g10 -W1 $CONTINUEPS -Dh $VBSE >>
$OUTFILE
```

Draw Wadati-Benioff zone contour lines

```
LINE=-W2./$DKRED
WBZFILE=${ROOTNAME}.WBZ
\rm $WBZFILE
touch $WBZFILE
cat $SAMDATA/0836_25km_bend_notrench.gmt >> $WBZFILE
cat $SAMDATA/575.gmt >> $WBZFILE
nawk 'BEGIN {print "$"} !/\$/ { print $2, $1}' $SAMDATA/
sj-100-km-well-defined.gmt >> $WBZFILE
nawk '{ print $1, $2}' $SAMDATA/0836_100km_extn.gmt >>
$WBZFILE
psxy -R$REGION $PROJ -M$ $CONTINUEPS $LINE $WBZFILE $VBSE >>
$OUTFILE
```

Draw lines from earthquake to stations

```
sac <$MACRO | nawk -f sachdr.nawk > $0.tmp

EQLAT=-16.26
EQLON=-73.64
psxy -R$REGION $PROJ -M$ -L -W1/$YELLOW $CONTINUEPS $VBSE
<<END>> $OUTFILE
`nawk '{print '$EQLON','$EQLAT'}{print $1,$2}{print "$"}'
$0.tmp`
END
```

Plot stations

```
psxy -R$REGION $PROJ -Sc0.3 -G$CYAN -L -W.1/0 $CONTINUEPS
$0.tmp $VBSE >> $OUTFILE
```

Could also have done with

```
#sac <$MACRO | nawk -f sachdr.nawk | psxy -R$REGION $PROJ -
Sc0.1 -G$CYAN -L -W.1/0 $CONTUNUEPS $VBSE >> $OUTFILE
#psxy -R$REGION $PROJ -Sc0.3 -G$CYAN -L -W.1/0 $CONTUNUEPS
$VBSE <<END>> $OUTFILE
#`sac <$MACRO | nawk -f sachdr.nawk`
#END
```

Plot earthquake

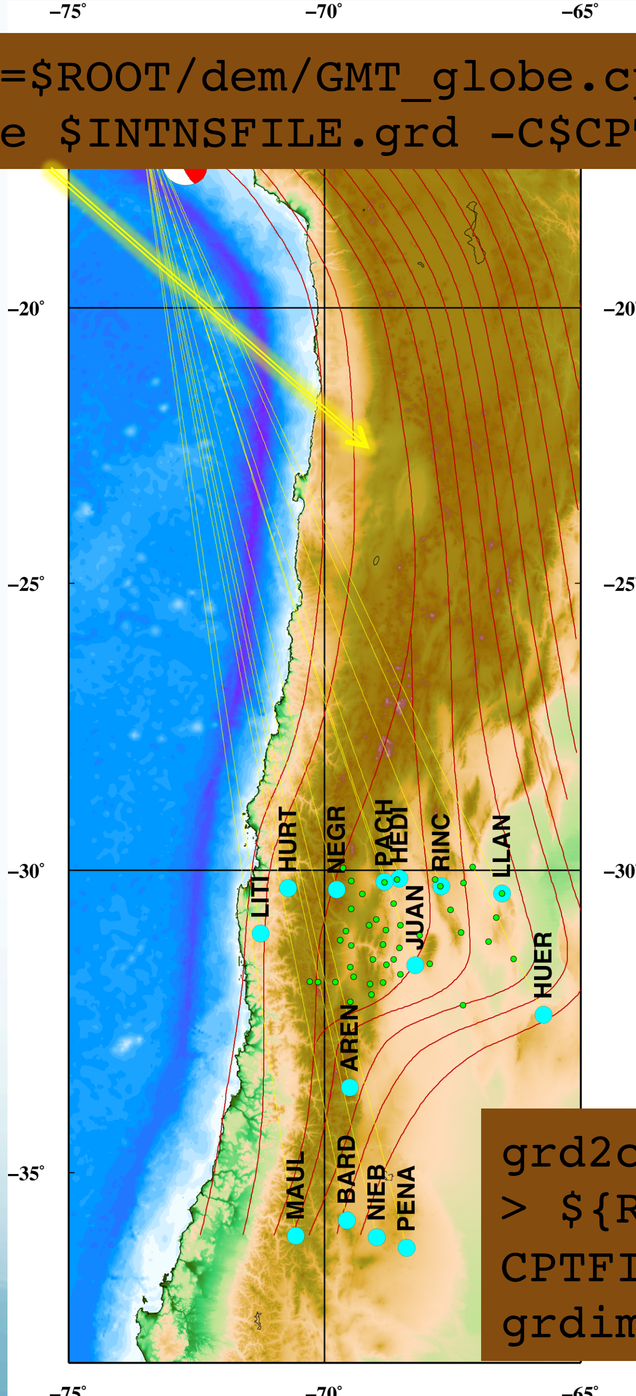
```
echo $EQLON $EQLAT | psxy -R$REGION $PROJ -Sc0.3 -G$RED -L -W.
1/0 $CONTINUEPS $VBSE >> $OUTFILE
```

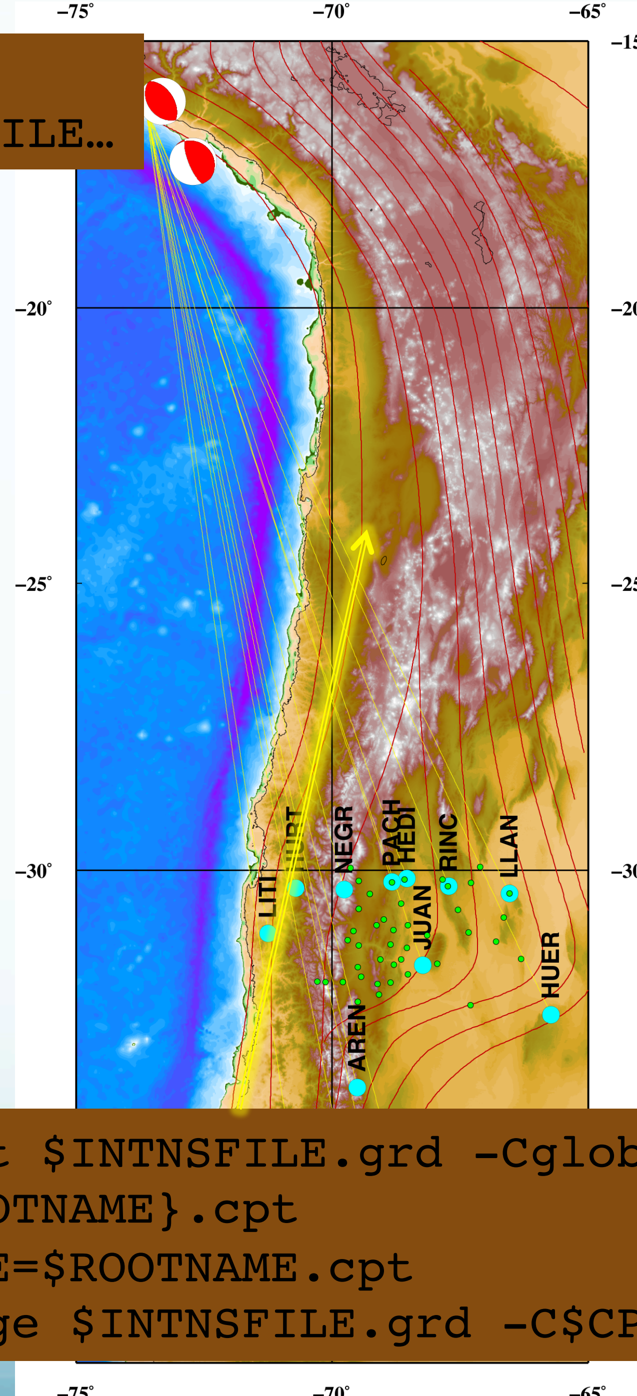
Plot focal mechanism

```
MECASIZE=.5
psmeca -R$REGION $PROJ -Sd$MECASIZE/0/0 -G$RED $ENDPS -L -
W0.5/$BLACK $VBSE << END >> $OUTFILE
`nawk '{print $1, $2, $3, $4, $5, $6, $7, $8, $9, $10}'
eq.cmt`
`nawk '{print $1, $2, $3, $4, $5, $6, $7, $8, $9, $10}'
eq.usgsmt`
```

```
CPTFILE=$ROOT/dem/GMT_globe.cpt
grdimage $INTNSFILE.grd -C$CPTFILE…
```

Topo to color, no shading



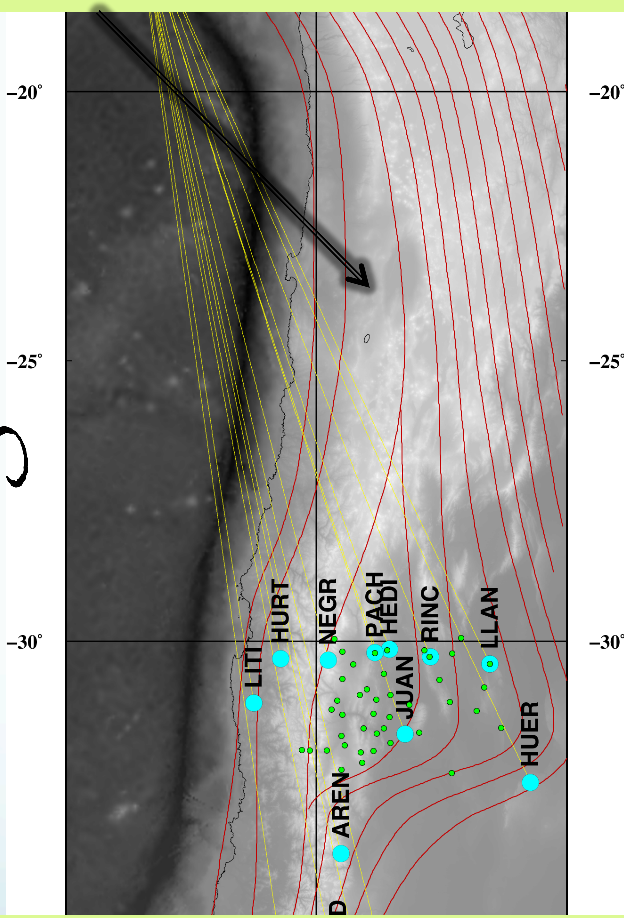


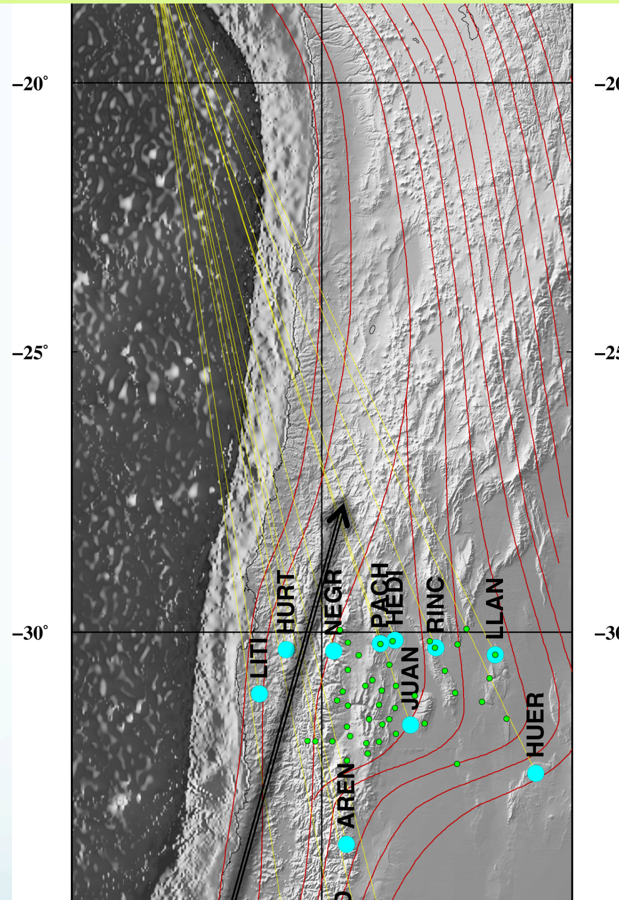
Topo to color (color rescaled for wider range), no shading

```
grd2cpt $INTNSFILE.grd -Cglobe -E128
> ${ROOTNAME}.cpt
CPTFILE=$ROOTNAME.cpt
grdimage $INTNSFILE.grd -C$CPTFILE…
```

```
rd2cpt $INTNSFILE.grd -Cgray -E128 > ${ROOTNAME}.cpt
CPTFILE=$ROOTNAME.cpt
#topo to graysacel plus shading
grdimage $INTNSFILE.grd -C$CPTFILE…
```

Topo to grayscale, no shading

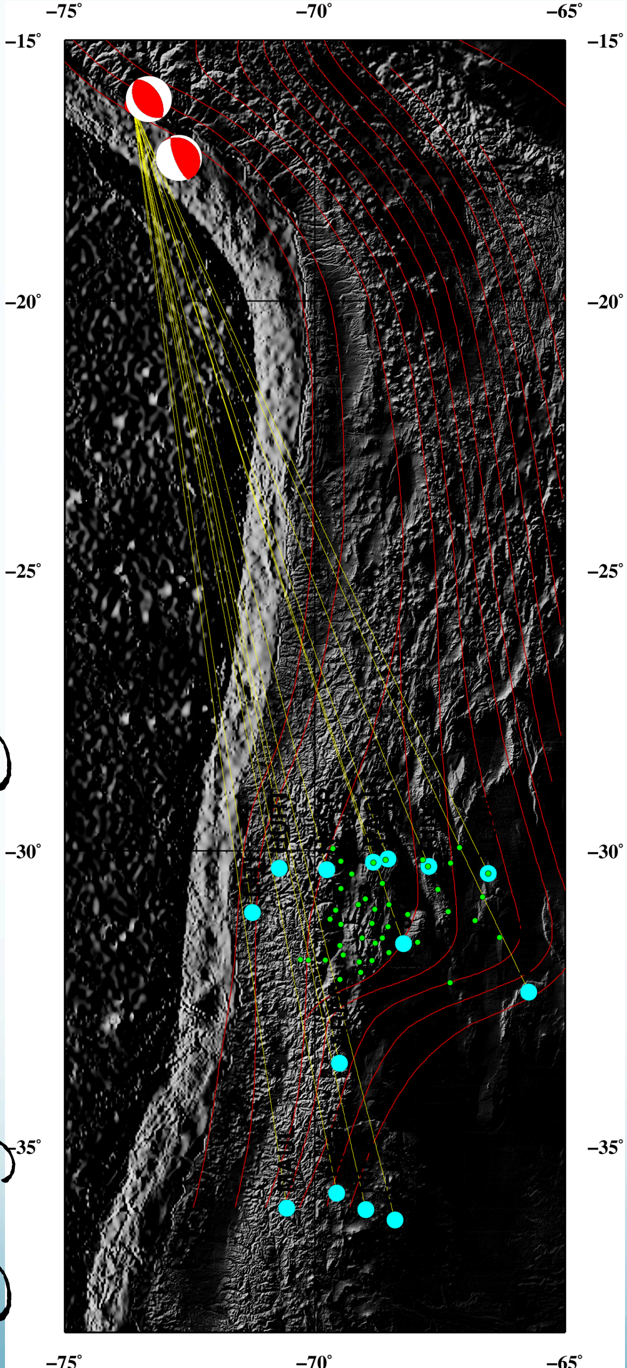


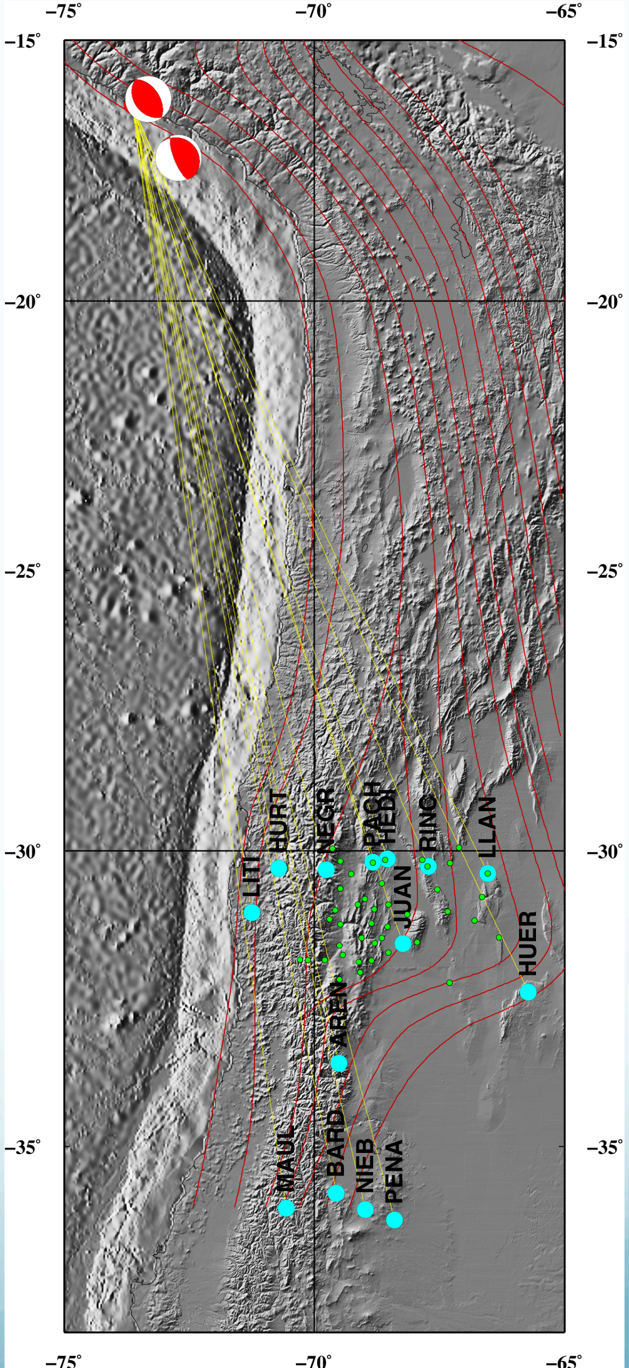


Topo to grayscale, with shading

```
rd2cpt $INTNSFILE.grd -Cgray -E128 > ${ROOTNAME}.cpt
CPTFILE=$ROOTNAME.cpt
#topo to graysacel plus shading
grdimage $INTNSFILE.grd -I$INTNSFILE.intns -C$CPTFILE…
```

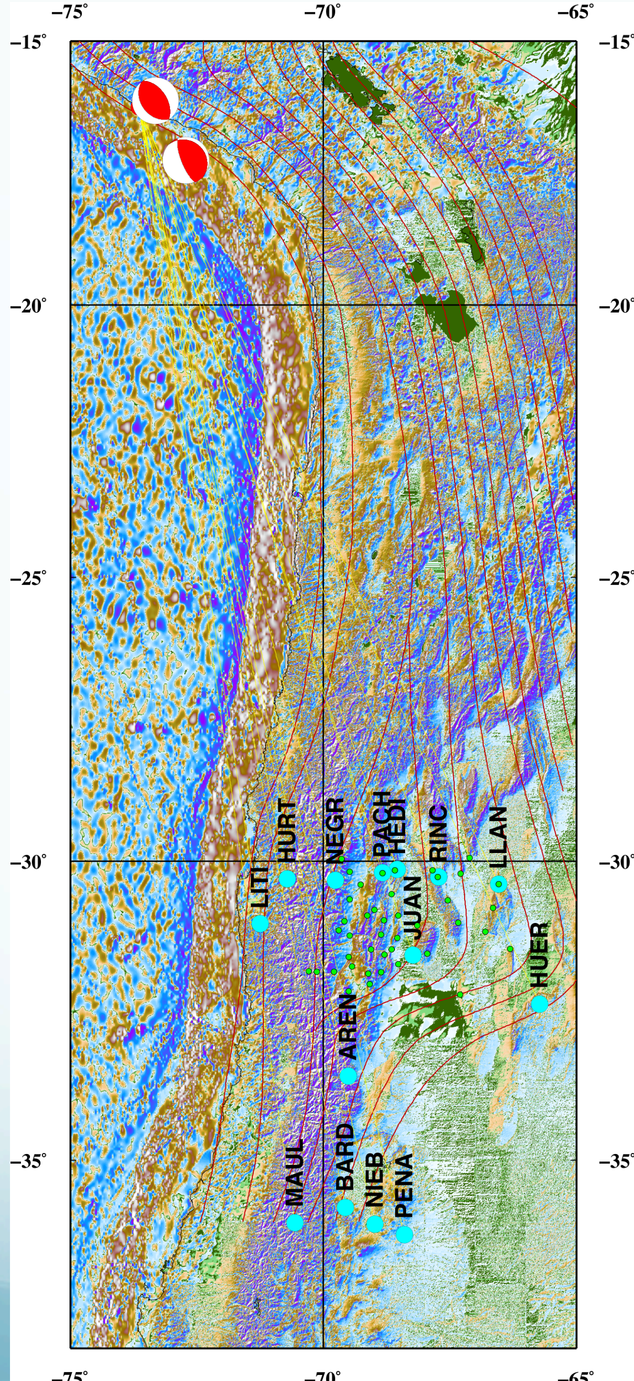
Shading (intensity) to grayscale, high contrast





Shading (intensity) to grayscale, low contrast

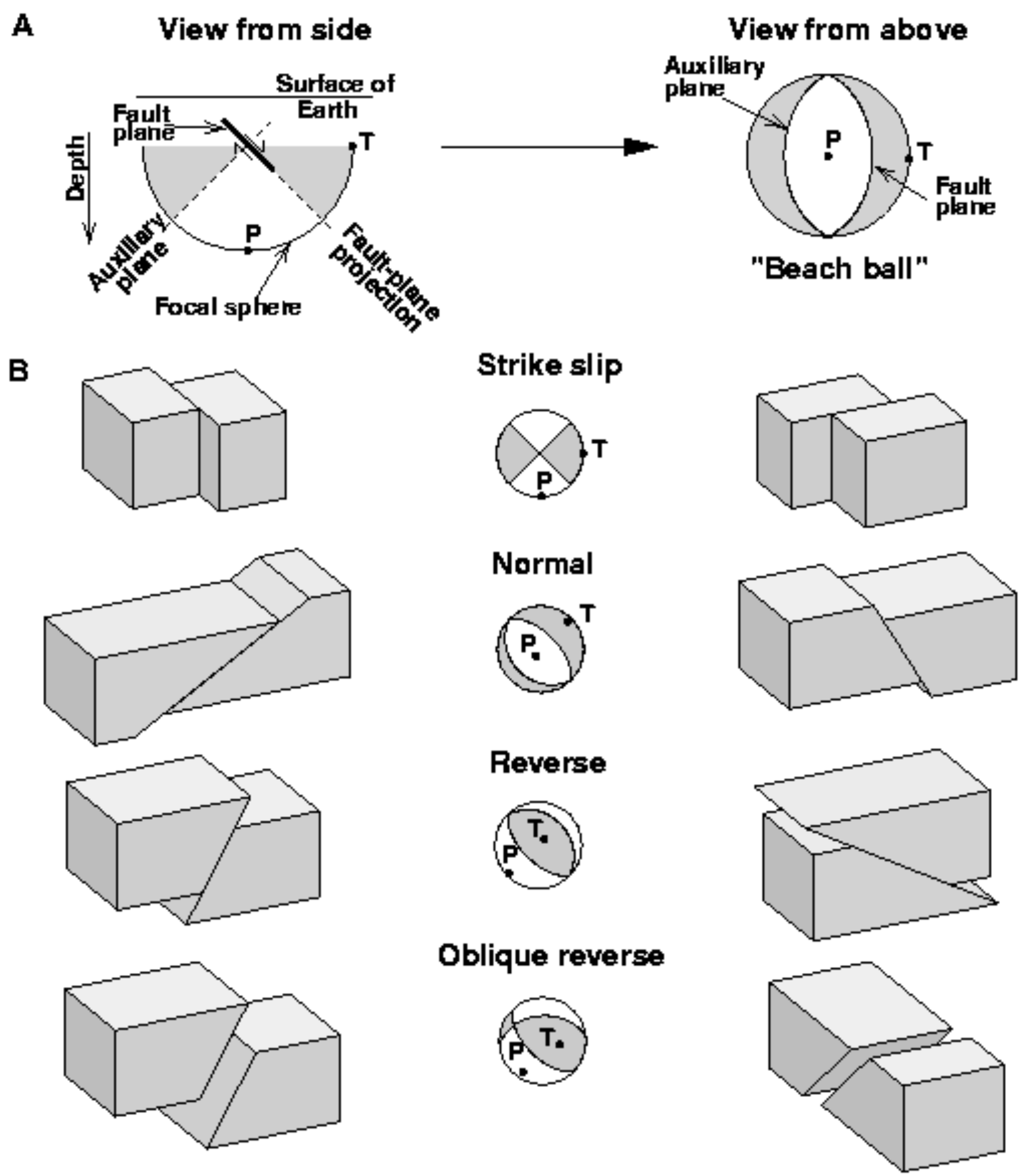
Shading (intensity) to color



```
grdgradient
grdgradient 4.3.1 - Compute directional gradients from grid files

usage: grdgradient <infile> -G<outfile> [-A<azim>[/<azim2>]] [-D[a][o][n]]
[-E[s|p]<azim>/<elev[ambient/diffuse/specular/shine]>]
[-L<flag>] [-M] [-N[t_or_e][<amp>[/<sigma>[/<offset>]]]] [-S<slopefile>] [-V]

    <infile> is name of input grid file

    OPTIONS:
    -A sets azimuth (0-360 CW from North (+y)) for directional derivatives
      -A<azim>/<azim2> will compute two directions and save the one larger in
magnitude.
    -D finds the direction of grad z.
       Append c to get cartesian angle (0-360 CCW from East (+x)) [Default:
azimuth]
       Append o to get bidirectional orientations [0-180] rather than directions
[0-360]
       Append n to add 90 degrees to the values from c or o
    -E Compute Lambertian radiance appropriate to use with grdimage/grdview.
       -E<azim/elev> sets azimuth and elevation of light vector.
       -E<azim/elev/ambient/diffuse/specular/shine> sets azim, elev and
        other parameters that control the reflectance properties of the surface.
        Default values are: 0.55/0.6/0.4/10
        Specify '=' to get the default value (e.g. -E60/30/=/0.5)
       Append s to use a simpler Lambertian algorithm (note that with this form
       you only have to provide the azimuth and elevation parameters)
       Append p to use the Peucker picewise linear aproximation (simpler but
faster algorithm)
       Note that in this case the azimuth and elevation are hardwired to 315 and
45 degrees
```

```
         This means that even if you provide other values they will be ignored.
      -G output file for results from -A or -D
      -L sets boundary conditions.  <flag> can be either
         g for geographic boundary conditions
         or one or both of
         x for periodic boundary conditions on x
         y for periodic boundary conditions on y
         [Default:  Natural conditions]
      -M to use map units.  In this case, dx,dy of grid
         will be converted from degrees lon,lat into meters (Flat-earth
approximation).
         Default computes gradient in units of data/grid_distance.
      -N will normalize gradients so that max |grad| = <amp> [1.0]
        -Nt will make atan transform, then scale to <amp> [1.0]
        -Ne will make exp  transform, then scale to <amp> [1.0]
        -Nt<amp>/<sigma>[/<offset>] or -Ne<amp>/<sigma>[/<offset>] sets sigma
           (and offset) for transform. [sigma, offset estimated from data]
      -S output file for |grad z|; requires -D
      -V Run in verbose mode [OFF].
```

Schematic diagram of a focal mechanism



USGS, 1996

Why is GMT so popular?

The price is right!
(But there's also no such thing as a free lunch!)

Offers unlimited flexibility since it can be called from the command line, inside scripts, and from user programs.

Has attracted many users because of its high quality PostScript output.

"Easily" installs on almost any (including windows) computer.

GMT Defaults

There are about 100 parameters which can be adjusted individually to modify the appearance of plots or affect the manipulation of data. Each as a default value.

GMT defaults are kept in a file called ~/.gmtdefaults4. There are tons of them and you can find out what they are and what the mean reading the man page for gmtdefaults.

When a program is run, it initializes all parameters to the GMT defaults, then tries to open the file .gmtdefaults4 in the current directory.

If not found, it looks in a sub-directory ~/.gmt, and finally in your home directory itself.

If successful, the program will read the contents and set the default values to those provided in the file.

If a script works for the author who gave it to you and not for you (in terms of size, position on page, etc.), your defaults are probably different.

To view your current gmtdefault setting

```
%gmtdefaults –L
```

To view the list of options for each default parameter

```
%man gmtdefaults
```

Plotting Defaults

example of start of .gmtdefaults4

#

GMT-SYSTEM 4.2.1 Defaults file

#

#-------- Plot Media Parameters --

PAGE_COLOR = 255/255/255

PAGE_ORIENTATION = landscape

PAPER_MEDIA = letter

#--- Basemap Annotation Parameters --

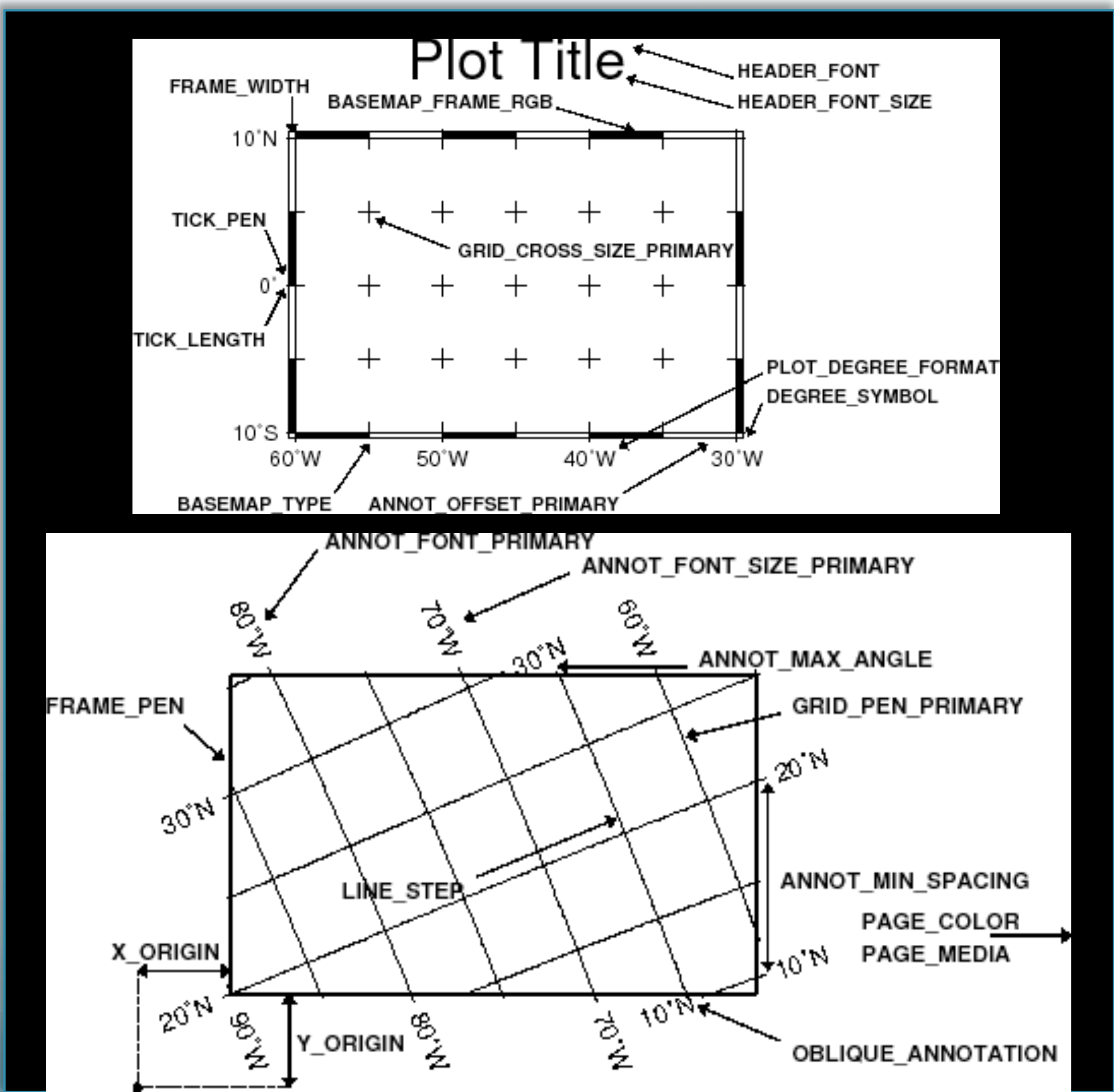
ANNOT_MIN_ANGLE = 20

ANNOT_MIN_SPACING = 0

ANNOT_FONT_PRIMARY = Helvetica

ANNOT_FONT_SIZE = 14p

ANNOT_OFFSET_PRIMARY = 0.075i



Changing the defaults

You can edit your local copy of .gmtdefaults4 using nedit or vim

You can explicitly reset a default within a script using the command gmtset

```
#!/bin/sh
gmtset PAPER_MEDIA  letter
```

NOTE:

GMT uses the NETCDF data base package for DEMs (and some other stuff).

Another "free" UNIX package.

This has to be installed and maintained separately (and is done so by Mitch).

One has to put the SRTM files one downloads from NASA, the USGS or other source into NETCDF files (this is the hard part).

Have covered lots of stuff,

but even more stuff has not been covered

– there are 60 GMT and 35+ Supplemental programs!

Plus power of UNIX to manipulate them.

Automating getting data from webpages.

Use i-Macro in Firefox to save keystrokes on some web page, including saving the webpage.

Have i-Macro save the keystrokes in a file.

Then use the command

```
/usr/bin/open /Applications/Firefox.app http://run.imacros.net/?m=getPDE.iim
```

To "rerun" the keystrokes, stored in the file `getPDE.iim`, and get fresh data.

You will see Firefox open up and the web pages will flash by. At the end you will have a new data file!

The file `getPDE.iim` is found at `/Users/robertsmalley/imacros/macros` (or has a soft link to there)

And has the following contents (what you typed into the various boxes on the web page). You can edit this file to change region, start and stop days, depth, file names, etc.

```
VERSION BUILD=7400919 RECORDER=FX
TAB T=1
TAB CLOSEALLOTHERS
URL GOTO=http://earthquake.usgs.gov/earthquakes/eqarchives/epic/epic_rect.php
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SLAT2 CONTENT=-24
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SLAT1 CONTENT=-42
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SLON1 CONTENT=-77
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SLON2 CONTENT=-63
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SYEAR CONTENT=2010
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SMONTH CONTENT=2
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:SDAY CONTENT=27
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:EYEAR CONTENT=2015
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:EMONTH CONTENT=12
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:EDAY CONTENT=31
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:NDEP1 CONTENT=0
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:http://neic.usgs.gov/cgi-bin/epic/epic.cgi ATTR=ID:NDEP2 CONTENT=50
TAG POS=1 TYPE=INPUT:SUBMIT FORM=ID:epic-form ATTR=NAME:SUBMIT&&VALUE:Submit<SP>Search
SAVEAS TYPE=CPL FOLDER=/users/robertsmalley/unixside/geolfigs FILE=chilePDE.htm
```