Introduction to Generic Mapping Tools
GMT

Monday, Aug 15, 2011

Short course on USArray data processing for the next generation of seismologists III

Suzan van der Lee

Go to a work directory/folder where you have assembled USArray SAC files for a particular earthquake. Also make sure you know the path to the "codes" directory/folder that you used for the preceding SAC, Response, and Shell scripting exercise.

During those exercises we made an event "list" for the one event you worked on yesterday in the SAC and Response sessions. Having an event list (with a single earthquake) is great, but if we had a station list we could visualize the geography of our data. Surprise: You can use a program that calls routines from the SAC library (sacio.a) to make the station list for the sac files you provide as arguments! To get a station list (of coordinates and station name) for all sac files in your directory, type

```
>../codes/sac2sl *SAC
```

This sends the requested coordinates to your screen. Also, some stations appear multiple times with the same coordinates because they are associated with multiple data streams and channels, each represented by a separate file. Instead we can type

```
>../codes/sac2sl *BHZ*SAC  > station.list
```

so we read only one file for most stations and we redirected the output to a file called "station.list". The two lists (event.list and station.list) can be used in a shell script (see yesterday's session on Scripting) along with generic mapping tools to create a map in PostScript format. The script evmap.gmt will use sac2sl to create its own station list from whichever sac files (ending with ".SAC") are found in the directory. It expects to find sac2sl in ../codes, so please make sure it lives there. Then type

```
> ../codes/evmap.gmt
```

which produces a file called "map.ps" and looks like this:

```
#!/bin/tcsh                          (the TC shell is like C shell, but with auto-
                                     completion functionality really only important in
date                                 interactive shells such as your login shell)
set seismograms = `ls *SAC`          (place all local SAC files in variable "seismograms")
                                     (extract only coordinates and 1 ID from event.list)
set evcoor = `more event.list | awk '{print $1, $2, $4}'`
../codes/sac2sl $seismograms | sort -k 3 -u > station.list
                        (get the station list and sort alphabetically while removing all duplicates.
                        Next, regroup coordinates in file "gc.list" to be used for plotting great
                        circle segments between event and stations)
awk '{print ela, elo; print $1, $2; print ">"}' ela=$evcoor[1] elo=$evcoor[2] station.list > gc.list
set proj = Oa-83/47/170/14           (map projection variable for GMT)
set range = 175/43/-32/3r            (map area variable for GMT, this one for North America)
psbasemap -R$range -J$proj -Ba0 -X3. -Y3. -K -V -P > map.ps          (map frame)
psbasemap -R -J$proj -Ba15g15f5wsEN -K -O -V >> map.ps               (repeat frame, with
ticks)
pscoast -R -J$proj -W4 -C128 -N1 -A2000 -O -K -V >> map.ps           (coast lines)
psxy -: gc.list -R -J$proj -K -O -M -W1/0 -V >> map.ps               (great circle segments)
psxy -: event.list -R -J$proj -K -O -Sc0.3 -W1/0 -G255/255/0 -V >> map.ps     (event)
```

psxy -: station.list -R -J$proj -O -St0.2 -W1/0 -G255/0/0 -V >> map.ps (*stations)*

The map can be shown with “gs”, or “open”, or “Preview”. Alternatively you can convert the map.ps to a map.pdf using the command “ps2pdf map.ps”

> gs map.ps

This script is centered on North America. If your event is elsewhere it will fall off the map and you may have to adjust the gmt script, particularly the map projection and map area variables. Also, if your “sac2sl” is not in the “../codes” directory, you will have to adjust this part of the gmt script before you run it to produce the map.

The commands that start with “ps” are GMT commands/tools. The tools generate PostScript (ps) language (designed for printing), which we’ll redirect to a file called map.ps. This, along with plotting it results in “> map.ps ; gs map.ps” pasted after every command. Let’s see how the GMT tools uses in this script break down, by typing

> psbasemap > map.ps; gs map.ps

Aha, it didn’t get enough input arguments, so it’s telling us how to use it properly. Use the map projection and map area from the script:

> psbasemap -Ba0 -JOa-83/47/-10/14 -R-32/3/175/43r > map.ps; gs map.ps
> psbasemap -Ba0 -JOa-83/47/-10/14 -R-32/3/175/43r -X12 -Y3 > map.ps; gs map.ps
> psbasemap -Ba15g15f5wsEN -JOa-83/47/-10/14 -R-32/3/175/43r -X12 -Y3 > map.ps; gs map.ps

So this drew a map frame for us, including a grid. Now let’s get the coastlines:

> pscoast –J –R –W > map.ps ; gs map.ps

two things happened: GMT remembered the settings for –J and –R form the previous GMT command (psbasemap), but it did not include its postscript output (the map frame & grid) in the file map.ps. To fix this, tell psbasemap that more postscript is following (-K) and tell pscoast that a Postscript file with header already exists (-O) and to append (>>) rather than overwrite (>) map.ps. At the same time, add the -V flag to both commands to receive more communication (verbose output) from the gmt tools.

> psbasemap -B -J -R -X12 -Y3 –K > map.ps
> pscoast –J –R –W –O –V >> map.ps ; gs map.ps

So many lakes in North America! Let’s unclutter the map by only mapping features that are larger than 2000 km:

> psbasemap -B -J -R -X12 -Y3 –K > map.ps
> pscoast –J –R –W –A2000 –O –V >> map.ps ; gs map.ps

One can add political borders, rivers, use colors, fill patterns, etc. there are tons of possibilities beyond this simple example. Please the references at the end for more information. Let’s continue here with our surface wave path (great-circle segment) lists:

> psbasemap -B -J -R -X12 -Y3 –K > map.ps
> pscoast –J –R –W –A2000 –O –K –V >> map.ps
> psxy -: gc.list –J -R -O -M -V >> map.ps; gs map.ps

I added a –M to indicate that file gc.list is not one single line but a collection of separate lines and I needed to use -: because GMT expects to read longitude first and latitude second, while my files have latitude first and longitude second. Let's finish up:

```
> psbasemap -B -J -R -X12 -Y3 –K > map.ps
> pscoast –J –R –W –A2000 –O –K –V >> map.ps
> psxy -: gc.list –J -R -O -K -M -V >> map.ps
> psxy -: event.list –J -R -W1/0 -Sc0.3 -G255/255/0 –O –K -V >> map.ps
> psxy -: station.list –J -R -W1/0 -St0.2 -G255/0/0 -O -V >> map.ps
```

And voila, our basic, simple map is ready. I used -W1/0 (draw a black line around the symbol) -Sc0.3 (make a circle) -G255/255/0 (make it yellow)" for the event and -St0.2 (make a triangle) -G255/0/0 (make it red) for the stations.

Example scripts for GMT are in /Applications/GMT/share/doc/gmt/examples/ex??

Exercises

1. Use a subset or all of the GMT commands used here in a shell script and run it.
2. Take evmap.gmt or a DUDE script (c??.*) and modify it to your liking, then run it.

Important references:

Type the command gmt

```
> gmt
```

or type the gmt command (from the output of "gmt") to get information on how to use it.

GMT Home page:
http://gmt.soest.hawaii.edu/

DOCS on the left menu goes to tutorials, manuals, cookbook, and reference info

EXAMPLES on the left menu show output of gmt scripts and provides the script

GMT mailing list:
http://gmt.soest.hawaii.edu/gmt/gmt_maillist.html

GMT development wiki:
http://www.soest.hawaii.edu/wiki/index.php/Generic_Mapping_Tools