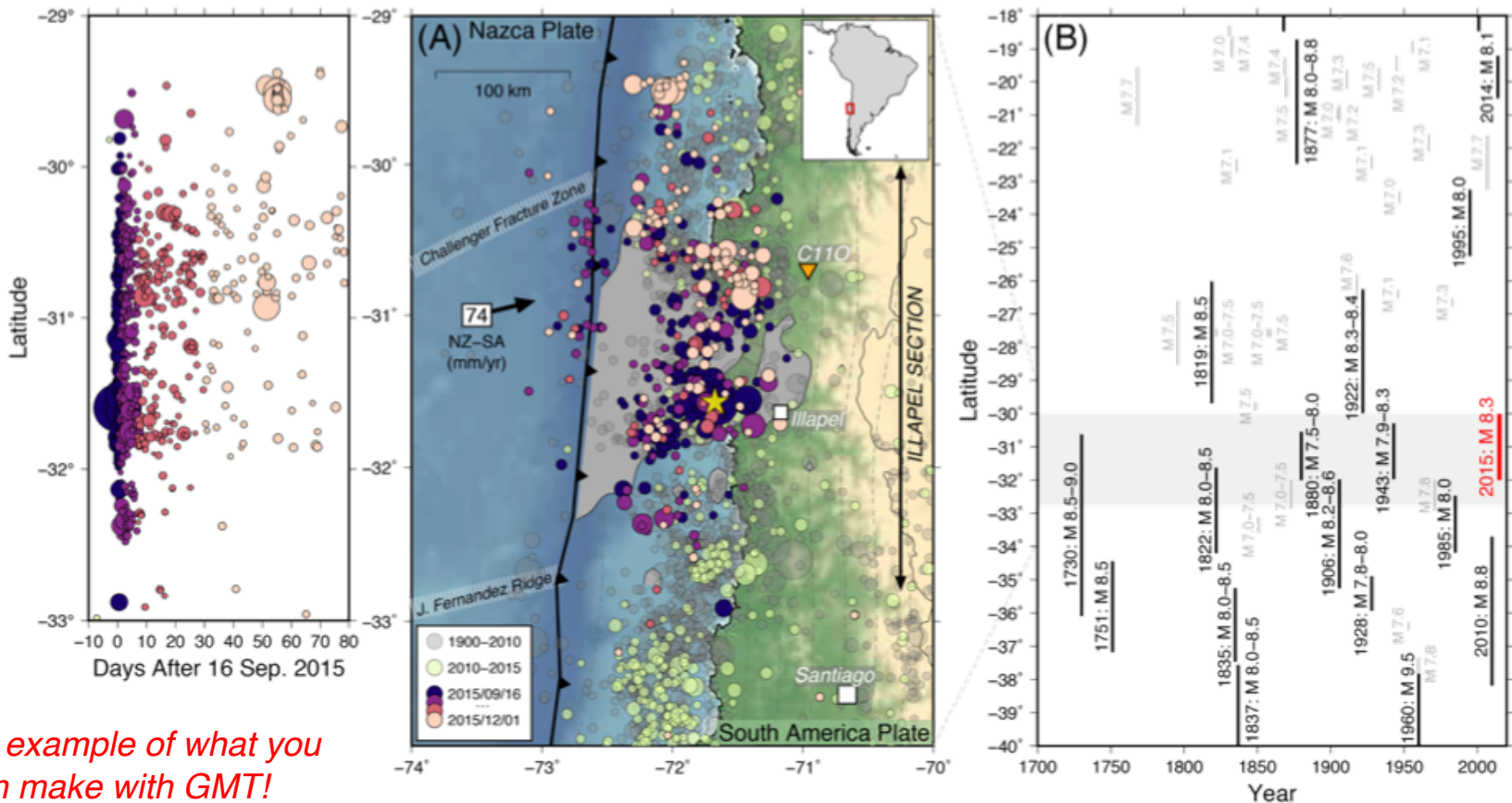


# Introduction to GMT (Part 1)



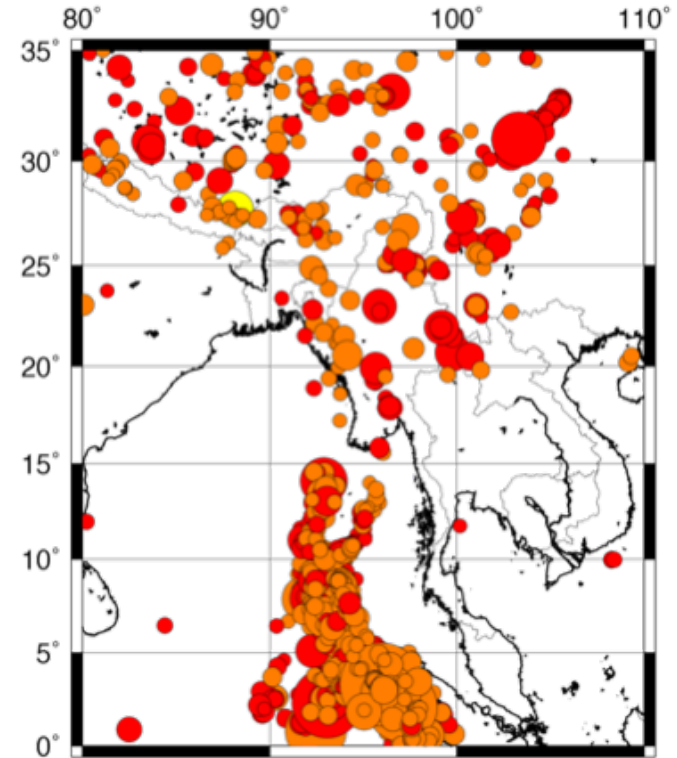
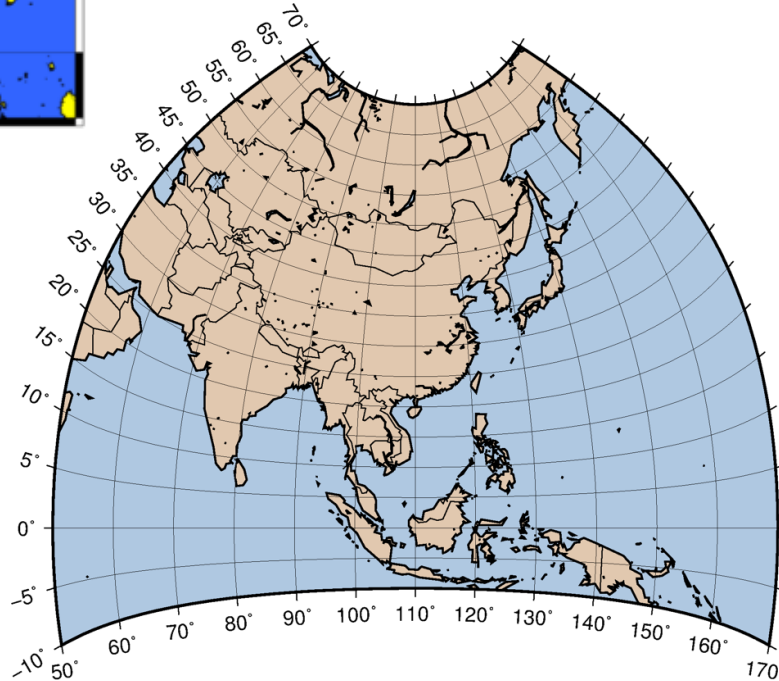
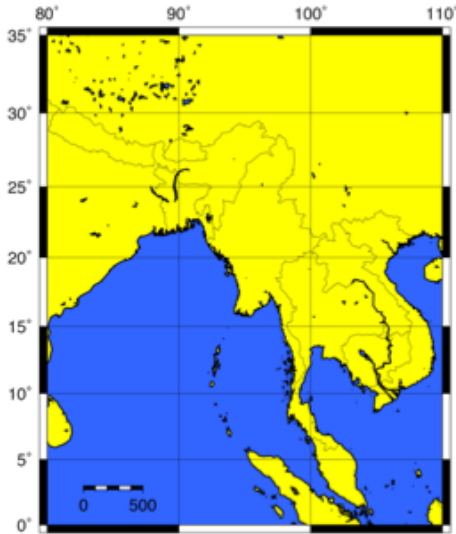
Beth Meyers  
Matt Herman

Last updated:  
31 January 2018

# Objectives

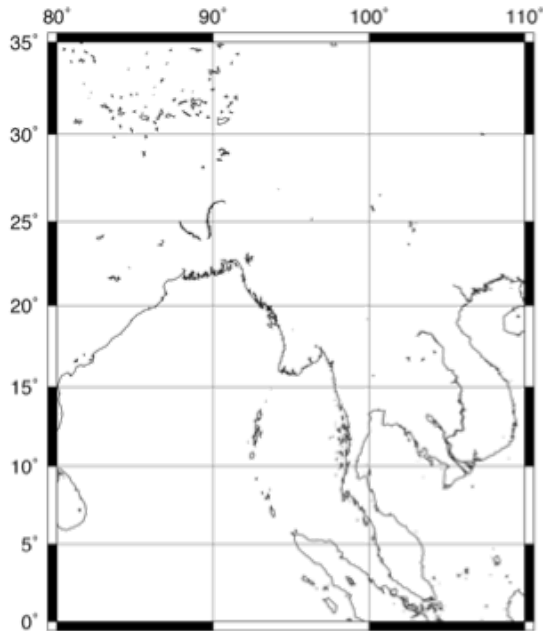
- Basic GMT commands
  - psbasemap
  - pscoast
  - psxy
- GMT syntax and command line options
- Note: this tutorial is specifically for GMT version 5

By the end of of this tutorial you will be able to  
create the following figures:

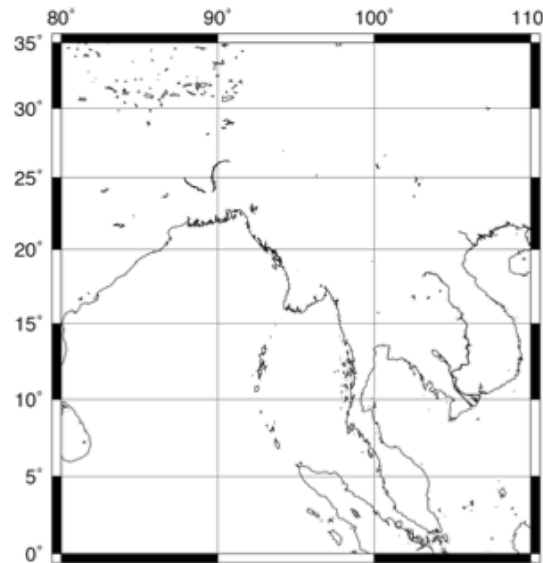


# Map Projections

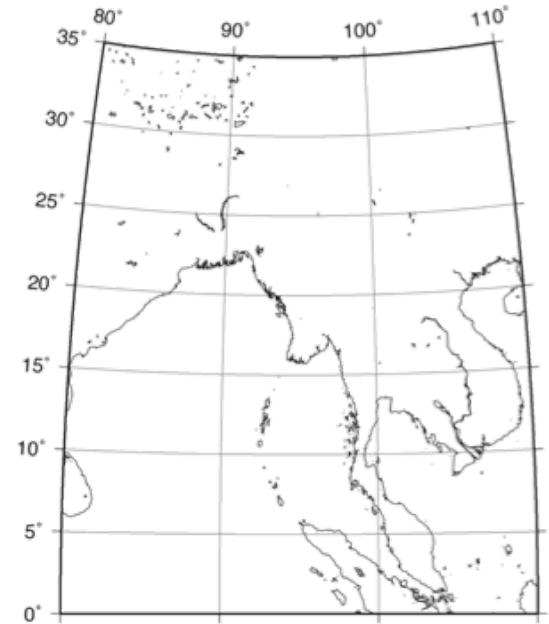
GMT plots the features of a *spherical* Earth on a *flat* surface. Note: It is impossible to unfold a sphere onto a flat surface perfectly. Therefore, any map projection distorts areas, angles, or both!



Mercator



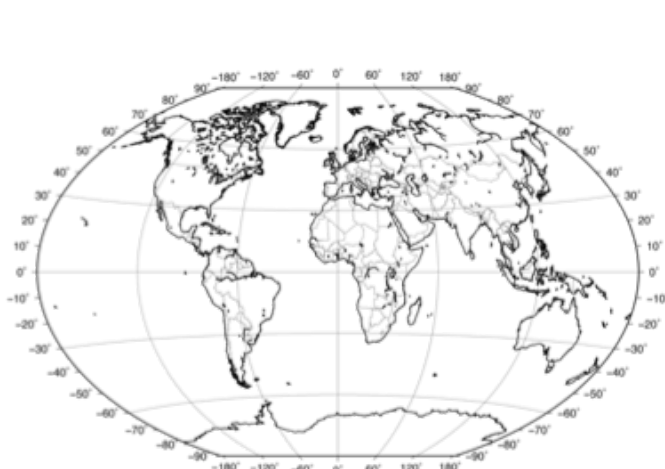
Cylindrical Equal Area



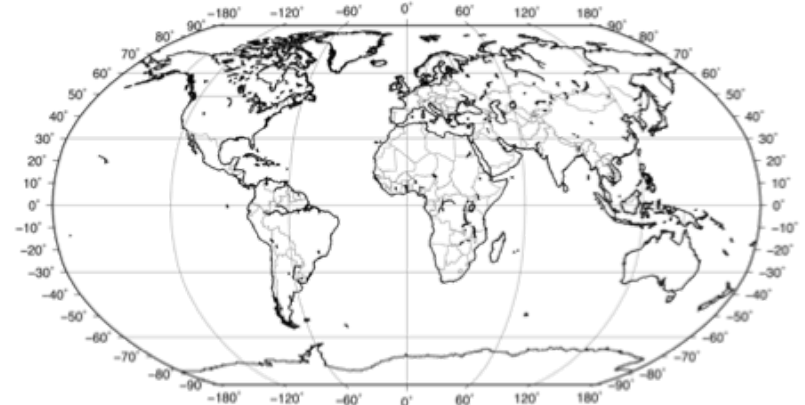
Cassini

# Map Projections

GMT plots the features of a *spherical* Earth on a *flat* surface. Note: It is impossible to unfold a sphere onto a flat surface perfectly. Therefore, any map projection distorts areas, angles, or both!



Winkel



Robinson



Mollweide

# Review of Unix Conventions and Terms

**Naming files:** Use only alphanumeric characters, “-”, and “\_”. Don’t use spaces or any other characters (\*&%\$!^~) when naming a file since they have a special meaning in Unix. Unix is case sensitive so UPPER CASE filenames are different from lower case ones. Also, try to keep filenames short so there is less to type.

**Command:** The name of a program. When you press enter, the computer performs an action.

**Options:** These are specified after the command, and are usually indicated by a dash. They modify the behavior of the command to make it do what you want. See the command “man” page (manual pages) to learn more about the capabilities of each command.

**Command line:** The prompt on an active terminal where you can write a line of code. When you want to execute multiple commands, put them into a **script** (text file) to save yourself time typing.

**Path:** List of directories that your **shell** (the program that interprets the commands) searches when you enter a command.

**Directory:** Same as folders. Everything you use in one script must be in the same directory, or in your path.

*Check out the Unix tutorial if you are not familiar with these terms!*

# Unix Special Characters

## Redirection of program outputs:

>	command > file1	output of command overwrites file1
>>	command >> file1	output of command appends to file1
	command1   command2	output of command1 is input for command2

## Directory specifications:

- . current directory
- .. directory above current directory
- ~ shorthand for the path to the home directory

## Other scripting symbols:

- # comment indicator; any characters following “#” will be ignored by script
- \ escape character; treats the character immediately following as its literal value instead of substituting a behavior; useful for multi-line commands

- Begin by opening the terminal application
- You may want to make a new directory for the exercises in this tutorial
- We start with `psbasetmap`...



# Introduction to psbasemap

Type this into the  
command prompt

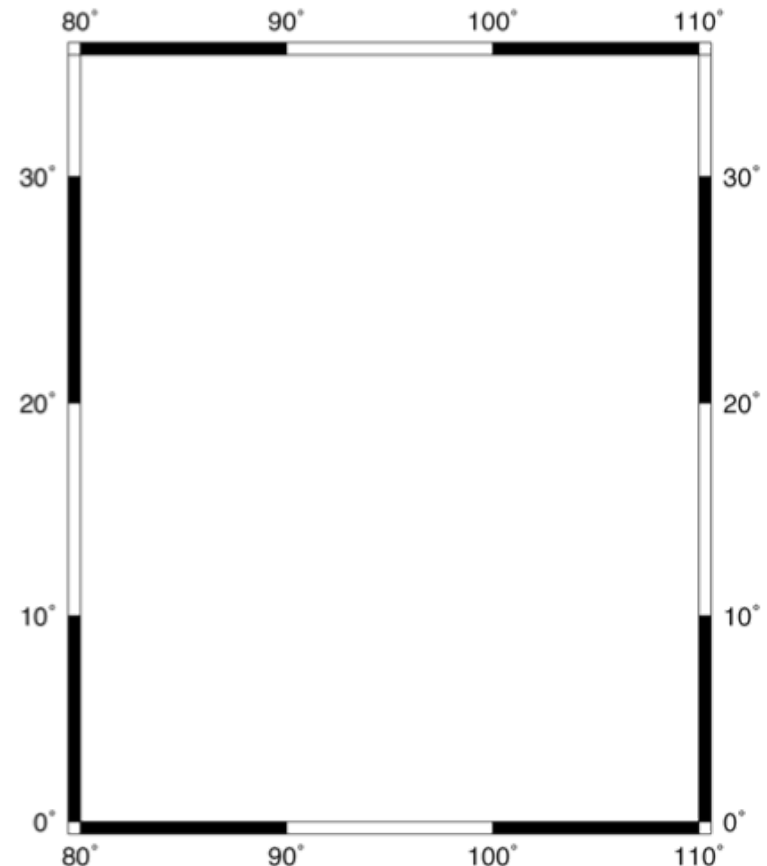
```
gmt psbasemap -JM10c -R80/110/0/35 -Ba10 > map1a.ps
```

## gmt psbasemap

All GMT5 commands begin with the main program named **gmt**. The tool **psbasemap** produces a base map, as the name suggests.

### -J -R -B

A single dash with a letter indicates an *option* (sometimes called a *flag*). Some options are required, others are added for controlling the output. Options must be separated by spaces. Some options have further specifications, e.g., **-JM10c**



# Introduction to psbasemap

```
gmt psbasemap -JM10c -R80/110/0/35 -Ba10 > map1a.ps
```

## **-JM10c**

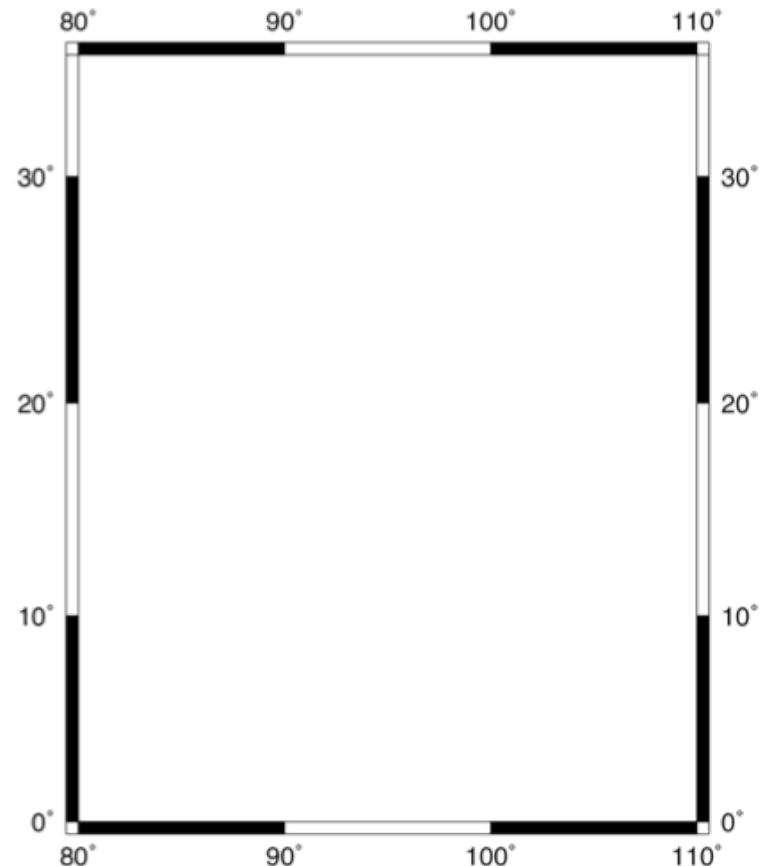
The **-J** option specifies the type of map projection. The **M** indicates Mercator. When upper case, you specify the map width afterwards: **10c** is 10 cm. You could instead use **i** for inches (e.g., **-JM5i**).

## **-R80/110/0/35**

The **-R** option specifies the map limits in the format west/east/south/north.

## **-Ba10**

The **-Ba** option specifies map boundary annotations. The number that follows is the increment, here every **10** units (degrees of longitude/latitude in this case).



# Introduction to psbasemap

```
gmt psbasemap -JM10c -R80/110/0/35 -Ba10 > map1a.ps
```

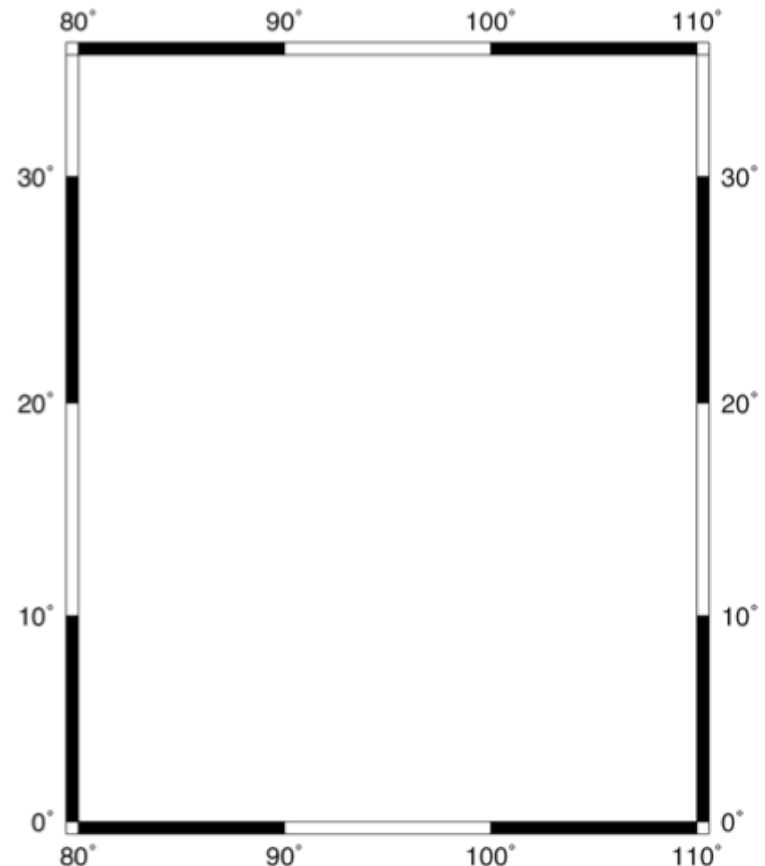
```
> map1a.ps
```

Redirect the output of the **psbasemap** command into a file called **map1a.ps**.

GMT programs generate PostScript output, which is basically impossible to decipher. Fortunately, almost any computer can open PostScript files. On a Mac, we open the PostScript file directly from the terminal by typing:

```
open map1a.ps
```

On a Linux (or Windows) computer, PostScript files can be opened with ghostview, ghostscript, or other image viewing applications.

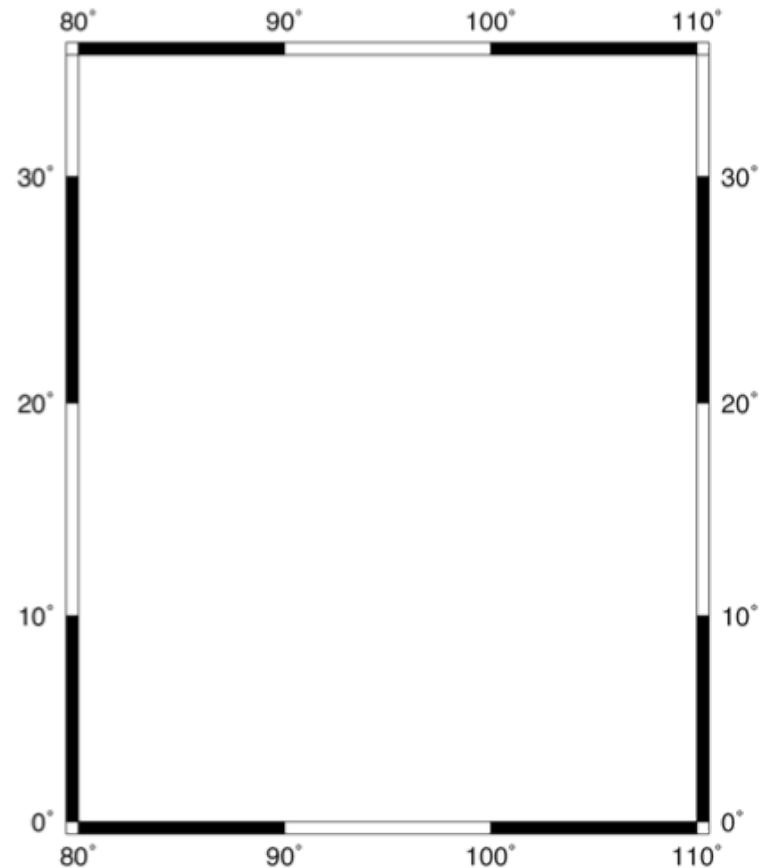


# Introduction to psbasemap

```
gmt psbasemap -JM10c -R80/110/0/35 -Ba10 -P > map1a.ps
```

## **-P**

If your map is appearing sideways add in the **-P** option. This option selects the portrait plotting mode. The default is landscape orientation.



# Introduction to psbasemap

*If you haven't yet, you should start putting all your commands into scripts. For an intro to scripts, see our [Unix Guide](#).*

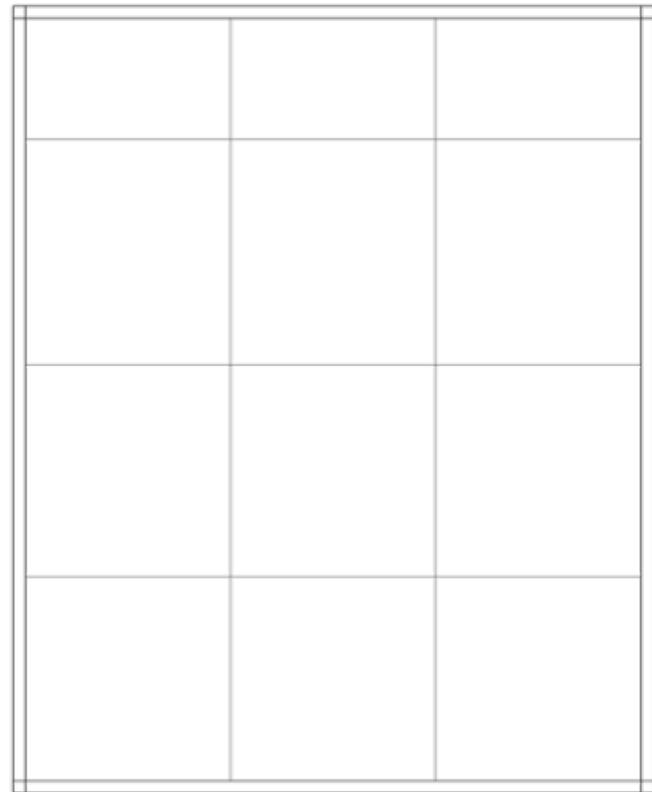
```
# change "-Ba10" to "-Bg10", "map1a.ps" to "map1b.ps"
gmt psbasemap -JM10c -R80/110/0/35 -Bg10 > map1b.ps
```

#

The first line starts with a pound sign. Everything on the line that comes after a pound sign is a comment and will not be interpreted as a command. Use comments to keep track of your work, remember what you have done, and so someone else can read your code.

**-Bg10**

Switching the **a** to **g** in the **-B** flag indicates gridline plotting instead of annotation (every **10** degrees). Since we removed the **a** we no longer have any boundary annotations.

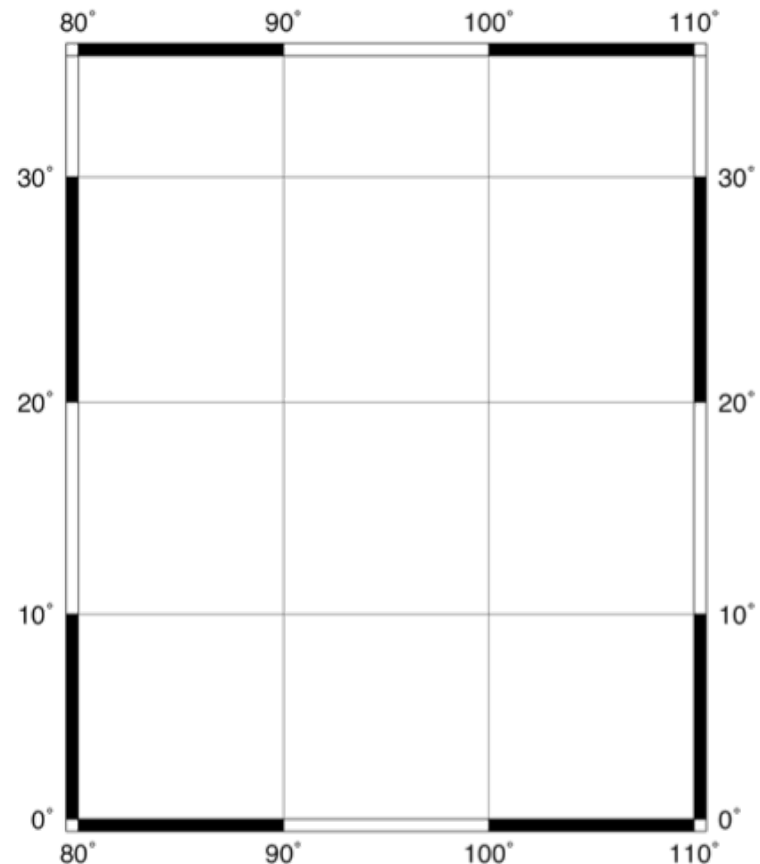


# Introduction to psbasemap

```
gmt psbasemap -JM10c -R80/110/0/35 -Ba10g10 > map1c.ps
```

## **-Ba10g10**

Here we combine **a** and **g** in the **-B** option to plot both annotations and gridlines. Since each letter has a **10** after it, each is plotted every 10 degrees.



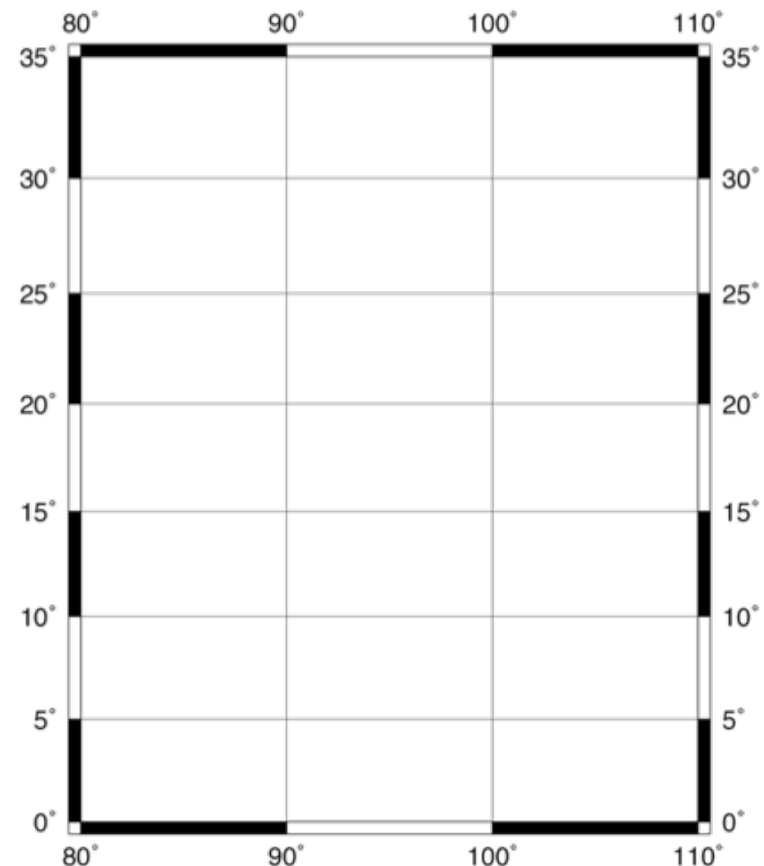
# Introduction to psbasemap

*This should all be  
on one line.*

```
gmt psbasemap -JM10c -R80/110/0/35 -Bxa10g10 -Bya5g5 >  
map1d.ps
```

**-Bxa10g10 -Bya5g5**

What if we want different annotations and gridlines for latitude and longitude? To define axes independently, we can divide the **-B** option into x and y components, using **-Bx** and **-By**. The annotation and gridline syntax is the same as before, but now they can have different values. This can be especially useful for maps with large aspect ratios, near-polar maps, or unusual projections.

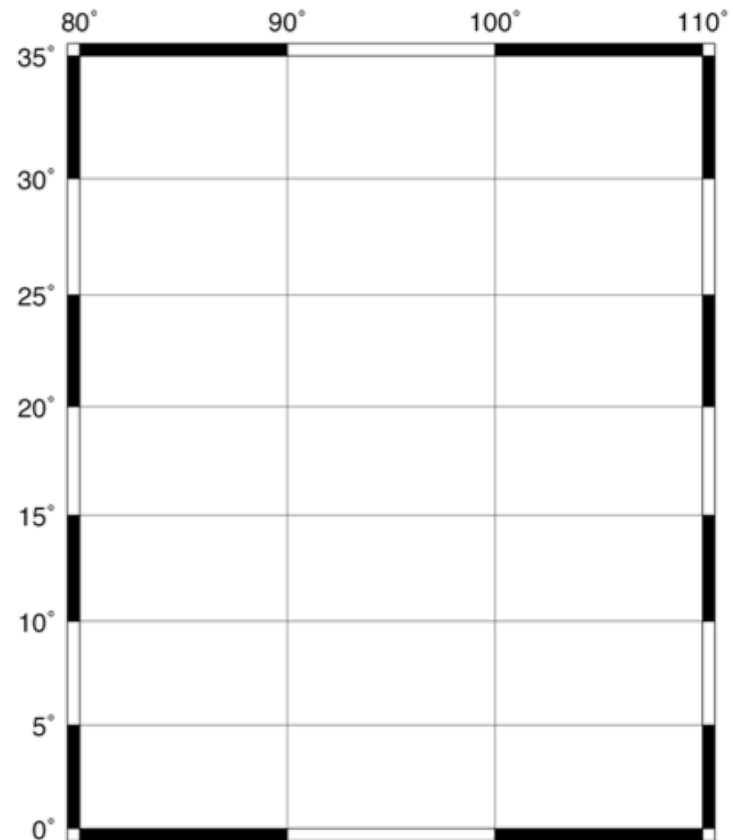


# Introduction to psbasemap

```
gmt psbasemap -JM10c -R80/110/0/35 -Bxa10g10 -Bya5g5 -BWesN >  
maple.ps
```

**-Bxa10g10 -Bya5g5 -BWesN**

Finally, we can define on what sides to plot the annotations with **-BWesN**. This specifies which edges (**W**est, **e**ast, **s**outh, **N**orth) we would like to have longitude or latitude annotations. The capital letters **W** & **N** indicate which edges we want annotated, and the lowercase letters **e** & **s** indicate that we want to draw the east and south edges, but they will be left blank. If you omit a letter, that edge will not be drawn at all.





# Recap of PSBASEMAP

- Command to draw map frame
- Produces PostScript output
- Specify map projection and map limits (**-J** and **-R**)
- Annotate map frame (**-Ba**)
- Draw gridlines (**-Bg**)
- Specify edges to draw (**-BWeSn**)

New command!

# Introduction to pscoast

```
gmt pscoast -JM10c -R80/110/0/35 -W0.5p -K > map2a.ps
```

```
gmt psbasemap -JM10c -R80/110/0/35 -Bxa10g10 -Bya5g5 -BWesN -O >> map2a.ps
```

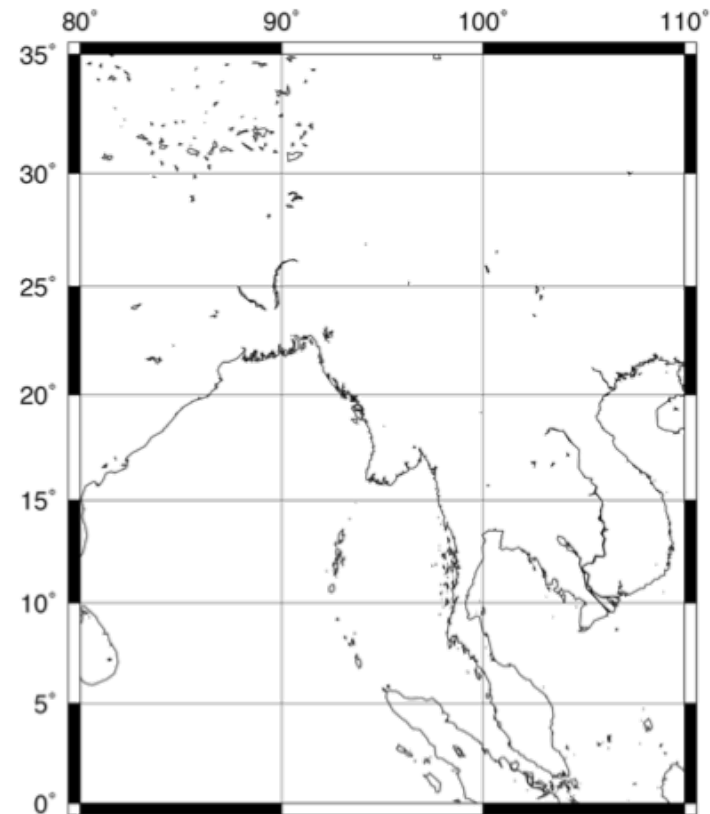
## gmt pscoast

This command draws coastlines (surprise!) and a few other geographical features such as country or state boundaries.

*Note: coastline datasets are included in the GMT package, so you do not need to find these separately unless something is wrong with the installation.*

## -JM10c -R80/110/0/35

Both **pscoast** and **psbasemap** (in fact, all gmt commands) have the same map projection and limits syntax. We want to use the same projection and syntax for both programs so the base map and the coastlines line up correctly.



# Introduction to pscoast

```
gmt pscoast -JM10c -R80/110/0/35 -W0.5p -K > map2a.ps
```

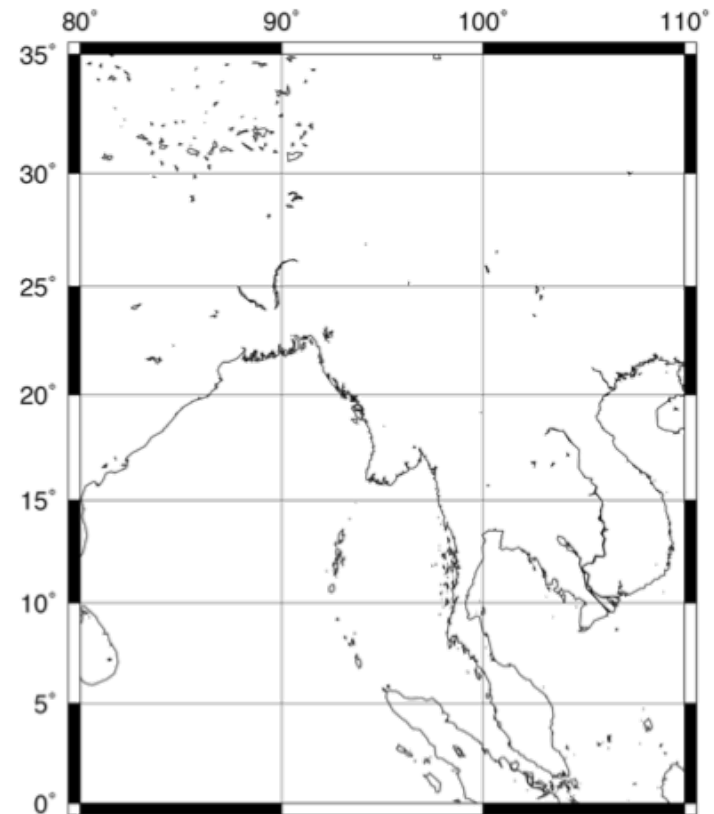
```
gmt psbasemap -JM10c -R80/110/0/35 -Bxa10g10 -Bya5g5 -BWesN -O >> map2a.ps
```

## **-W0.5p**

The **-W** option controls the pen that draws the coastline. The number specifies the thickness of the line. Here, **0.5p** means draw a line half a point thick, where a point is 1/72 of an inch (the **p** indicates a measurement in points).

## **-K**

We added this option to **pscoast**, indicating that we are going to add another layer to our map after this command. Every GMT command should include the **-K** option except the very last one.



# Introduction to pscoast

```
gmt pscoast -JM10c -R80/110/0/35 -W0.5p -K > map2a.ps
```

```
gmt psbasemap -JM10c -R80/110/0/35 -Bxa10g10 -Bya5g5 -BWesN -O >> map2a.ps
```

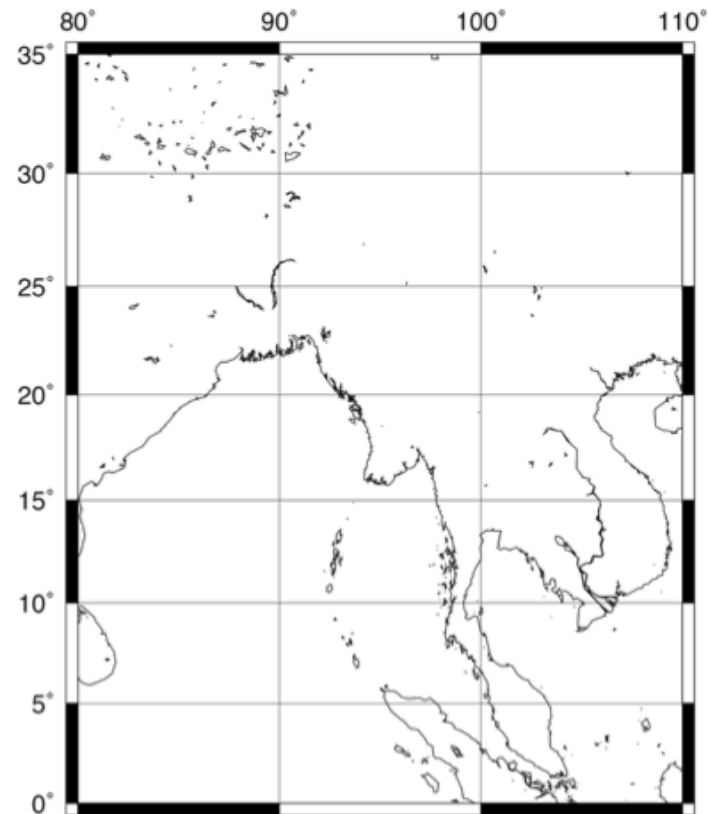
The psbasemap command is almost the same as before, with a few minor changes:

**-O**

We added the **-O** (*letter O, not number 0*) option, which stands for “overlay,” and indicates that the features in this layer will be placed on top of the previous layer instead of starting a new plot. **-O** should be included in every GMT line except for the first.

**>> map2a.ps**

Instead of the single redirect symbol (**>**), we used the double redirect symbol (**>>**). Recall: a single **>** overwrites the file with the output from the command, while the double **>>** appends the program output to the file.



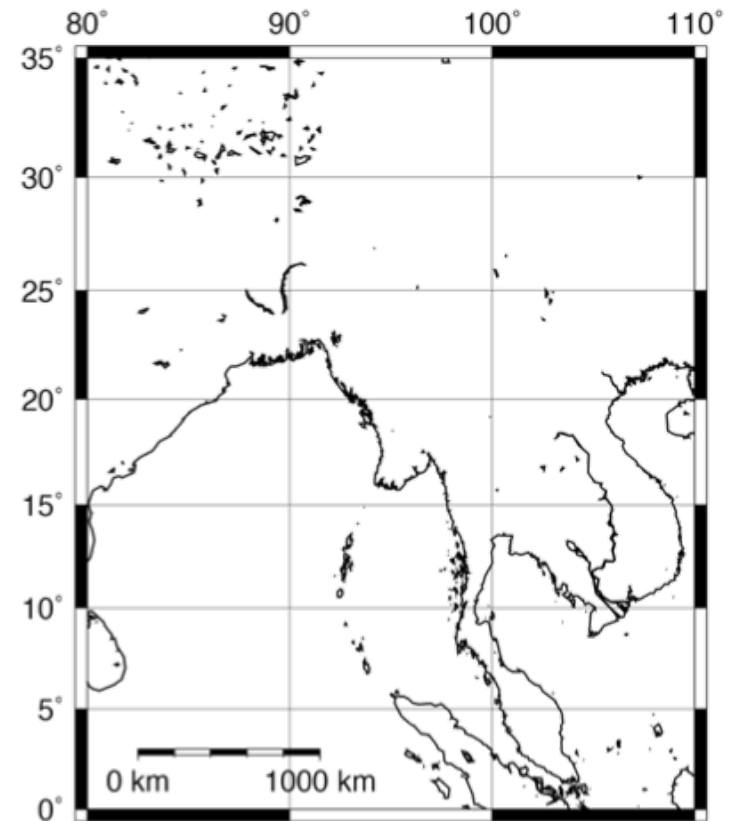
# Introduction to pscoast

*Option to add a  
scale bar*

```
gmt pscoast -JM10c -R80/110/0/35 -W1p -L87/3+c3+w1000k+u -K >  
map2b.ps
```

**-L87/3+c3+w1000k+u**

A scale bar is a useful addition to any map. The **-Llon/lat** flag creates a map scale centered at **lon/lat** (in this example, **87/3**). Most GMT options have additional options which allow for fine tuning the appearance. These are usually indicated with the **+** symbol. Here, **+c3** means that the scale will be calculated at a latitude of 3°N. The specifier **+w1000k** defines the length of the scale bar, with **k** indicating the units are km. Finally, **+u** tells GMT to plot the units on the scale bar.



# Introduction to pscoast

*Scale bar is now  
500 km long*

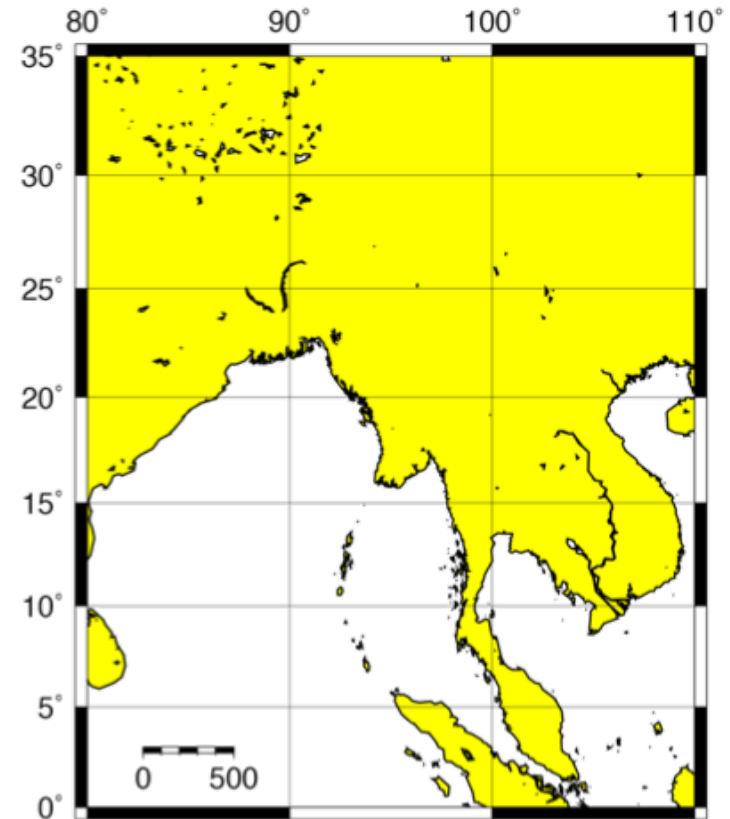
```
gmt pscoast -JM10c -R80/110/0/35 -W1p -L87/3+c3+w500k+u -G255/255/0  
-K > map2b.ps
```

Colors help to highlight or distinguish important features on your maps.

## **-G255/255/0**

The **-G** option colors in dry areas, i.e. land. The color is defined by RGB (red/green/blue) triplets, where each of R, G, and B ranges from 0 to 255. Yellow is **255/255/0**. GMT also recognizes many color names; in this case, we could have written **-Gyellow** and gotten the same result.

Search online for “gmt colors” to learn more about the built-in colors.

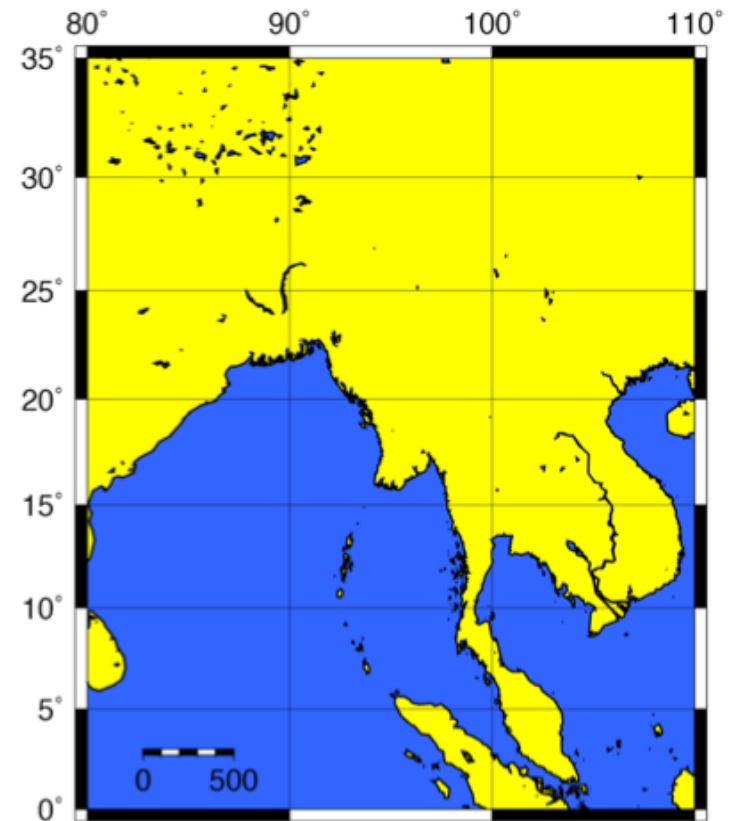


# Introduction to pscoast

```
gmt pscoast -JM10c -R80/110/0/35 -W1p -L87/3+c3+w500k+u -G255/255/0  
-S50/100/255 -K > map2b.ps
```

## **-S50/100/255**

Just like **-G** fills in dry areas, **-S** fills in the wet areas, like oceans, lakes, and rivers. The syntax is the same as in the **-G** option.



# Introduction to pscoast

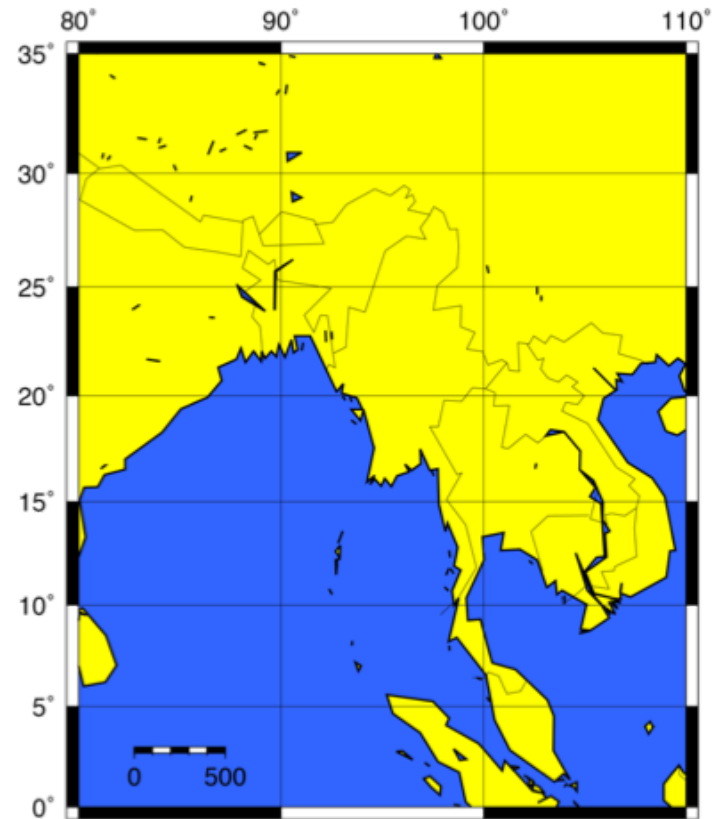
```
gmt pscoast -JM10c -R80/110/0/35 -W1p -L87/3+c3+w500k+u -G255/255/0  
-S50/100/255 -N1/0.25p -Dc -K > map2b.ps
```

## -N1/0.25p

The **-N** option draws political boundaries. The number directly afterwards specifies the type of boundary; **-N1** draws country boundaries. The number after the slash indicates the pen style (similar to the **-W** option), so here we are drawing a line 0.25 points thick.

## -Dc

You can specify the resolution of the coastline dataset, where the options are (**f**)ull, (**h**)igh, (**i**)ntermediate, (**l**)ow, and (**c**)rude. The default resolution is **i**. Lower resolution makes a smaller file size, useful for maps showing large regions, whereas high resolutions are better for showing details of zoomed-in areas.



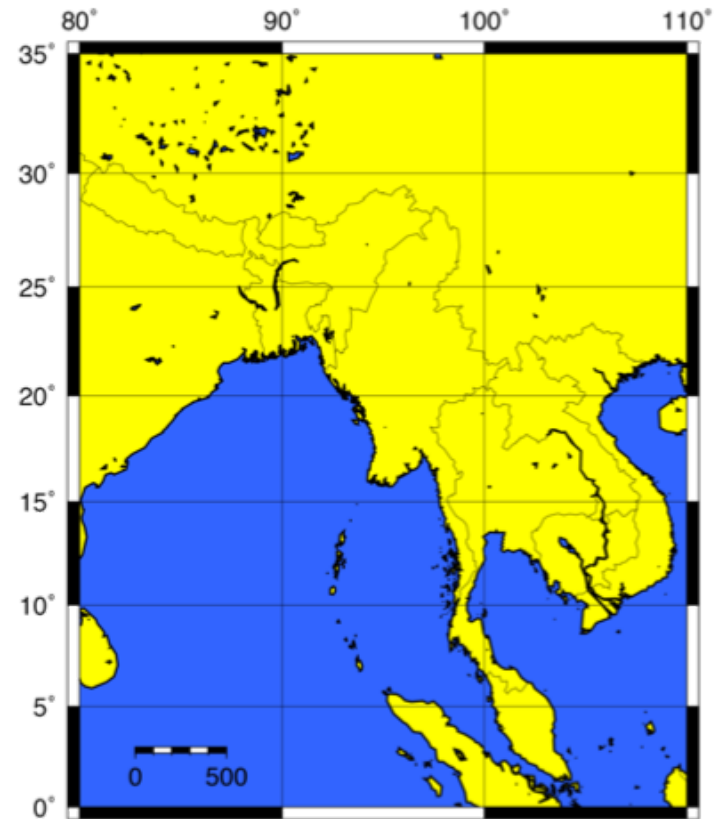


# Introduction to pscoast

```
gmt pscoast -JM10c -R80/110/0/35 -W1p -L87/3+c3+w500k+u -G255/255/0  
-S50/100/255 -N1/0.25p -Di -K > map2b.ps
```

## -Di

Intermediate resolution coastlines and country boundaries look better for our map region than crude resolution.



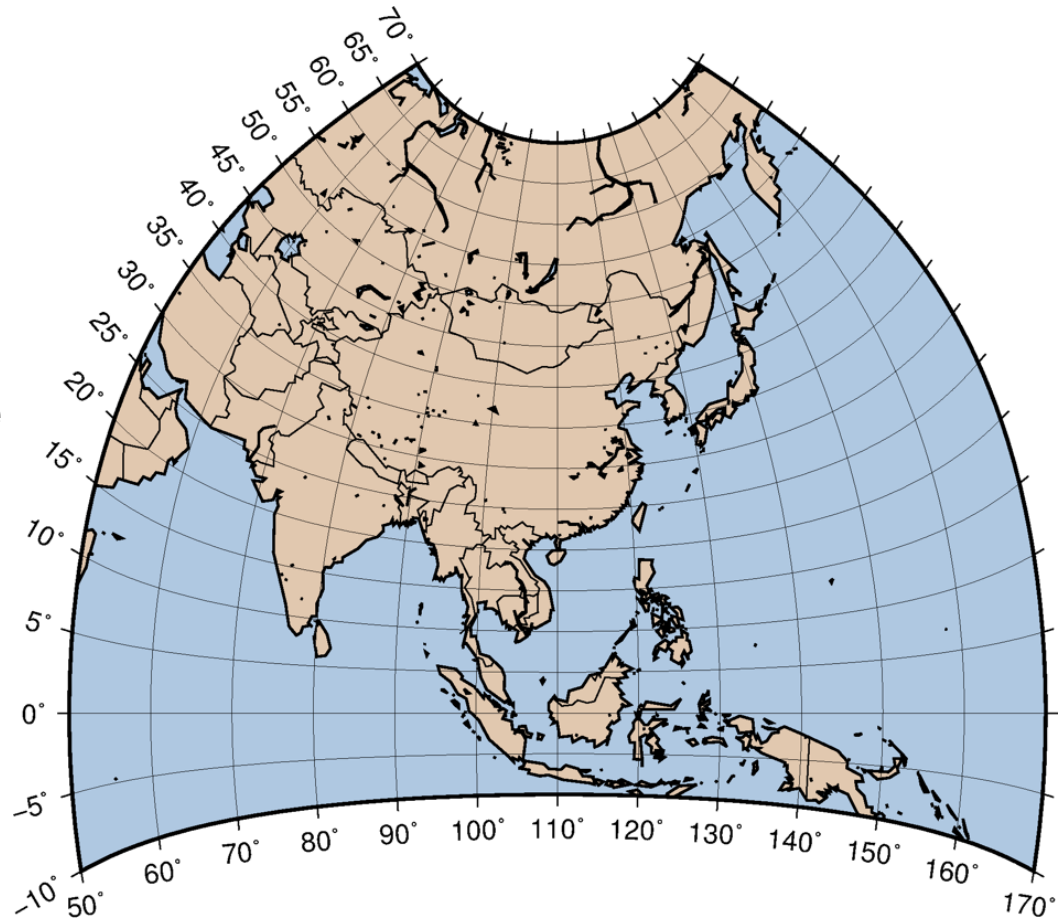
# Recap of PSCOAST

- Command to draw coastlines and country boundaries
- Specify pen (**-W**)
- Put in a scale bar (**-L**)
- Color dry (**-G**) and wet (**-S**) areas
- Draw country boundaries (**-N**)
- Change resolution (**-D**)

# Map of Asia

Now we are going to make a slightly different map of Asia using **pscoast** and **psbasemap**. We will use the relevant GMT options as well as shell scripting tools.

When we are finished, you will have the map shown at the right. Not a particularly useful map, but hey...



*We are creating a script to run GMT commands automatically instead of typing them into the terminal. If this does not make sense, see the Unix tutorial!*

# Map of Asia

Save the following as a text file called map.sh:

# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh ← First line indicates  
this a shell script.  
PROJ="-JC15c"  
LIMS="-R50/170/-10/70"  
BLUE="50/100/255"  
DRY="225/200/175"  
PSFILE="asia.ps"  
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \  
    -Dc -K > $PSFILE  
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

# Map of Asia

*Recall: variables help organize, save space, and make it so you only have to change a value in one place rather than in every instance in your script.*

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*Define GMT options  
as variables in the  
script.*

# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*Map projection: -JC means Cassini cylindrical projection*

# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*Map limits: in this case, we have chosen a much larger range than in the previous exercise*




# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*The color for wet areas*



# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```


*The color for dry areas* ←

# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*The name of the output  
PostScript file.*




# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*Put a backslash at the end of the line. This continues the options for the program to the next line.*



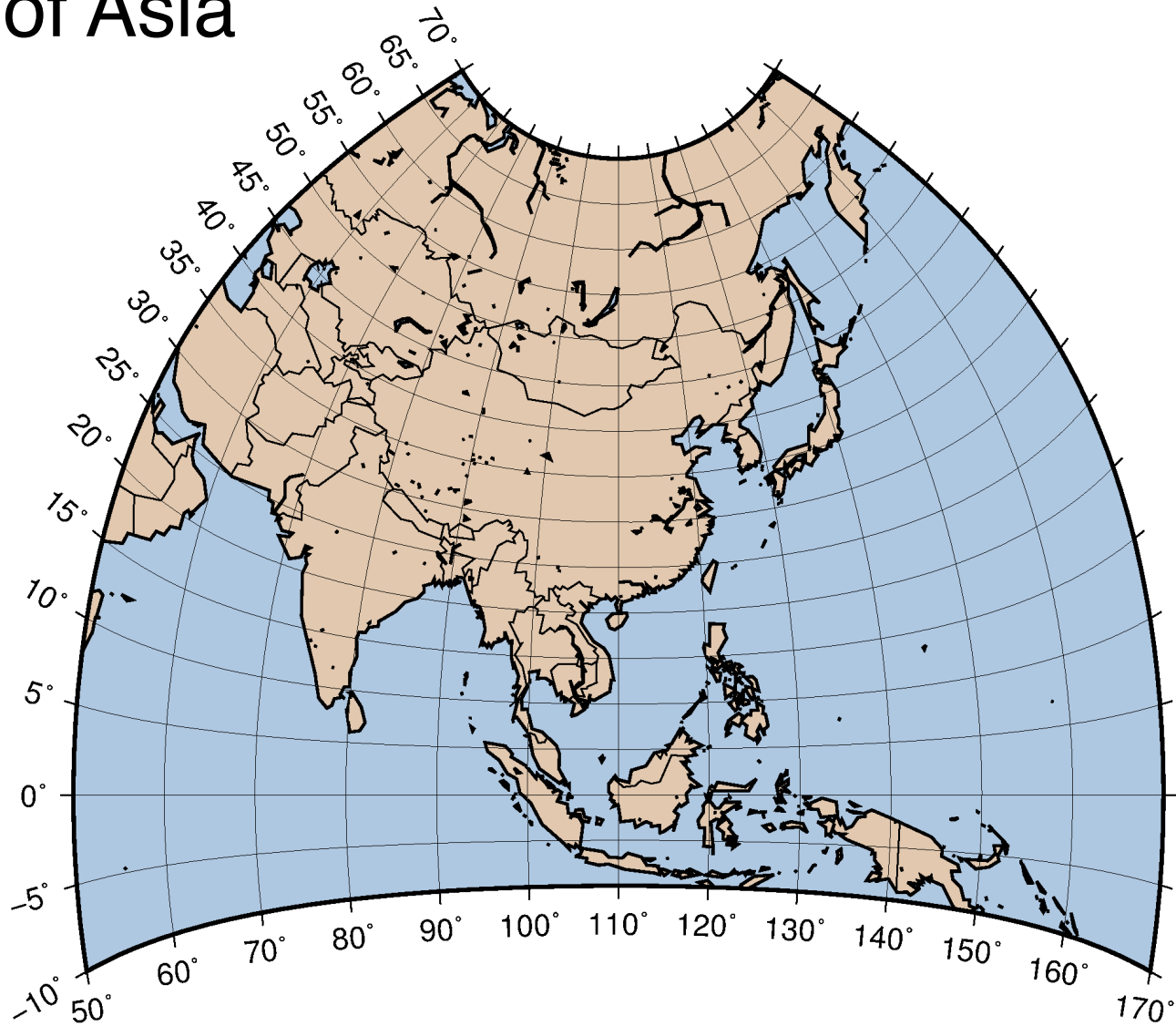
# Map of Asia

Save the following as a text file called map.sh:

```
#!/bin/sh
PROJ="-JC15c"
LIMS="-R50/170/-10/70"
BLUE="50/100/255"
DRY="225/200/175"
PSFILE="asia.ps"
gmt pscoast $PROJ $LIMS -W1p -G$DRY -S$BLUE -N1/0.25p \
    -Dc -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn -O >> $PSFILE
```

*Save your script and run it!*

# Map of Asia



# Recap of GMT Scripts

- Make scripts for using GMT! I cannot think of a single instance where typing the command directly into the terminal is better
- Use variables for options, especially options that are repeated
- The `\` “escapes” a carriage return, continuing a command to the next line

# Earthquakes in Southeast Asia

Save the following as a text file called earthquakes.sh:

```
#!/bin/sh
PROJ="-JM15c"
LIMS="-R80/110/0/35"
PSFILE="earthquakes.ps"
gmt pscoast $PROJ $LIMS -W1p -Dc -N1/0.5p -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn \
-K -O >> $PSFILE
```

*This script should look familiar: we are first generating a coastline with country boundaries and a base map.*

*Note: I included both -K and -O options in psbasemap, indicating that I am overlaying it on the previous map, and adding more afterwards.*



# Earthquakes in Southeast Asia

Save the following as a text file called earthquakes.sh:

```
#!/bin/sh
PROJ="-JM15c"
LIMS="-R80/110/0/35"
PSFILE="earthquakes.ps"
gmt pscoast $PROJ $LIMS -W1p -Dc -N1/0.5p -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn \
    -K -O >> $PSFILE
gmt psxy ....
```

*Let's add some more map  
content with psxy!*

# Introduction to psxy

```
gmt psxy stars.xy $PROJ $LIMS -Sa0.3i -W0.25p -Gred \  
-O -K >> $PSFILE
```

## gmt psxy

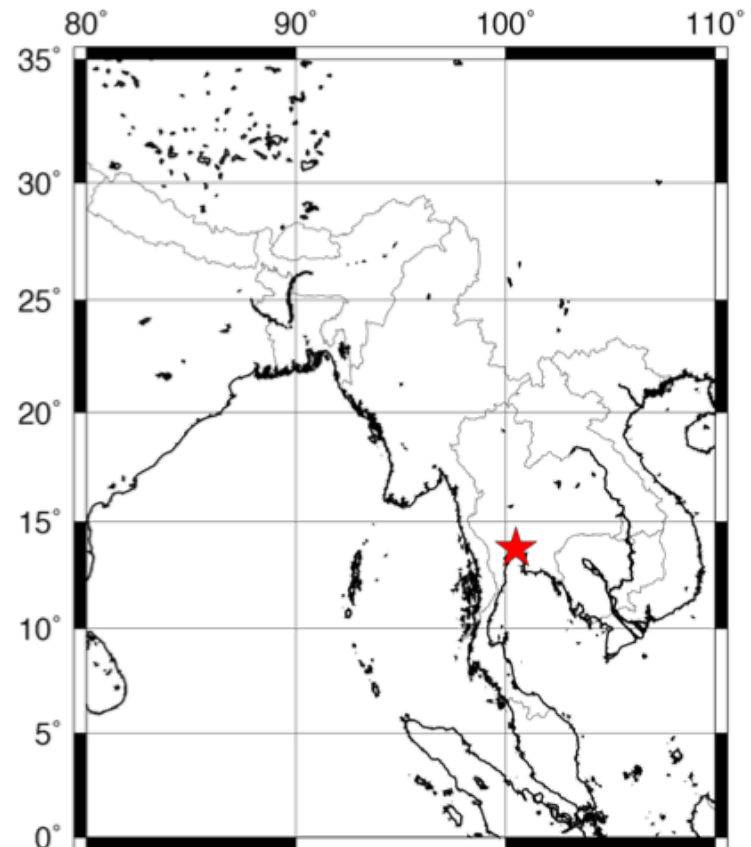
This command plots x-y data as lines or symbols on the map. If you specify a file after **psxy**, the program will look for input from the file. The data must have the first column be longitude and the second be latitude (*Note: at some point you will flip these; always check your input file*).

## stars.xy

Create a text file named **stars.xy** with the coordinates of Bangkok:

100.49 13.75

Save this file *in the same folder as your script*.



# Introduction to psxy

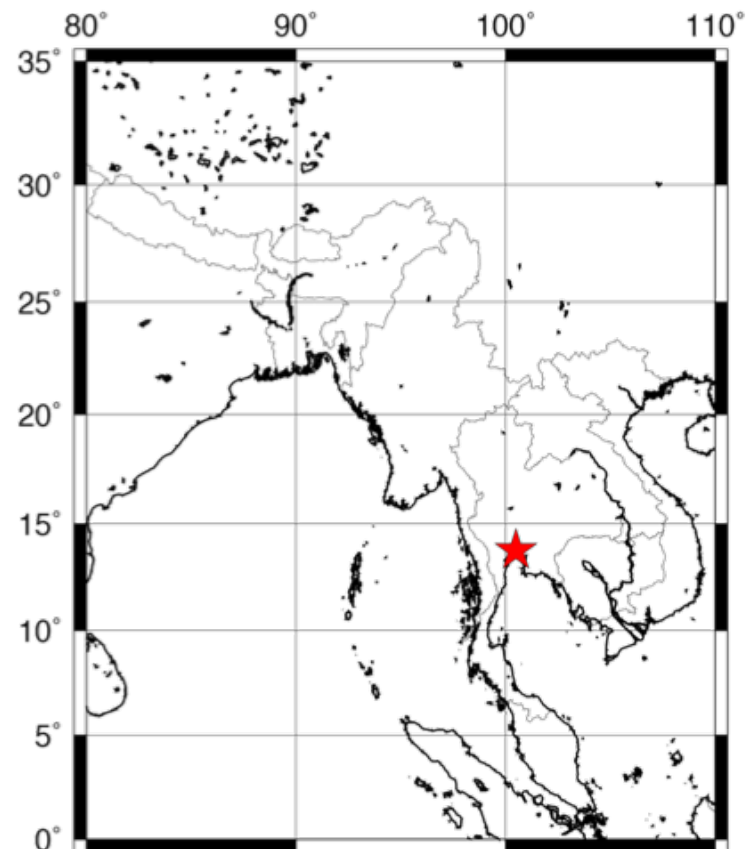
```
gmt psxy stars.xy $PROJ $LIMS -Sa0.3i -W0.25p -Gred \  
-O -K >> $PSFILE
```

## -Sa0.3i

To draw symbols, we use the **-S** option. There are many different types of symbols you can use; to see your options, look at the **psxy man** page (type **man psxy**). To draw a star that fits into a circle 0.3 inches in diameter, use **-Sa0.3i**.

## -W0.25p -Gred

To outline and color symbols, use the **-W** and **-G** options, where **-W** defines the pen style that outlines the symbol, and **-G** defines the symbol fill color. These options should look familiar from **pscoast**! At least one of these options must be used. If no fill option is selected, the symbol is transparent.



# Introduction to psxy

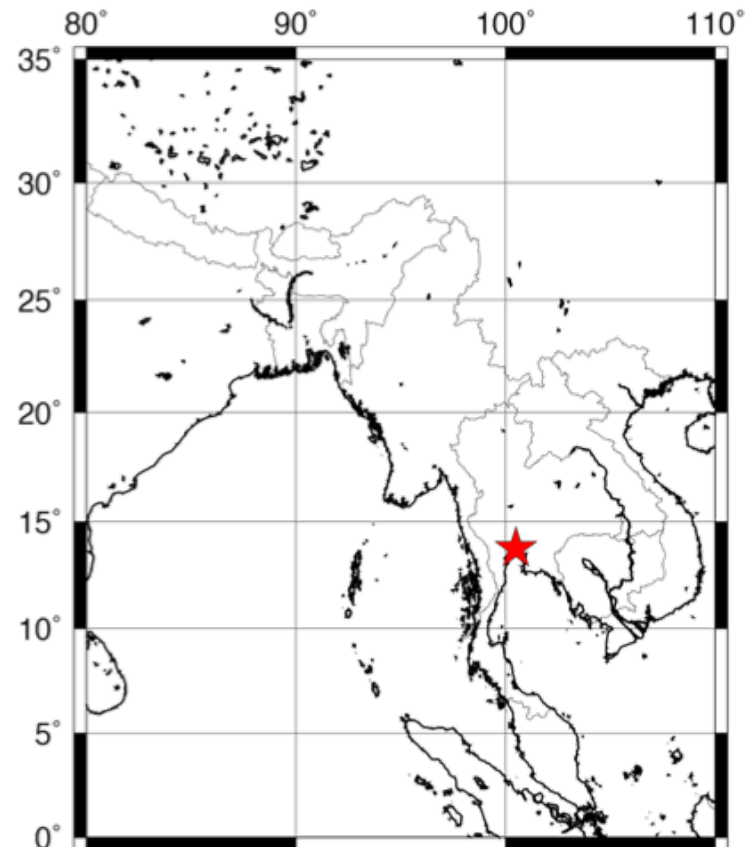
```
gmt psxy stars.xy $PROJ $LIMS -Sa0.3i -W0.25p -Gred \  
-O -K >> $PSFILE
```

## \$PROJ \$LIMS

We use the same projection and limits variables as the other commands to plot the star in the correct location on the page.

## -K -O

Because this layer is overlain on the previous layer, we must have **-O**. We are also going to add more to the map, so we also have **-K**.



# Plotting earthquake data using psxy

```
gmt psxy eqs.xy $PROJ $LIMS -Sc0.15i -W0.5p -Ggreen \  
-O -K >> $PSFILE
```

Now we are going to add earthquake locations to the map. You can download a text file with the earthquakes at:

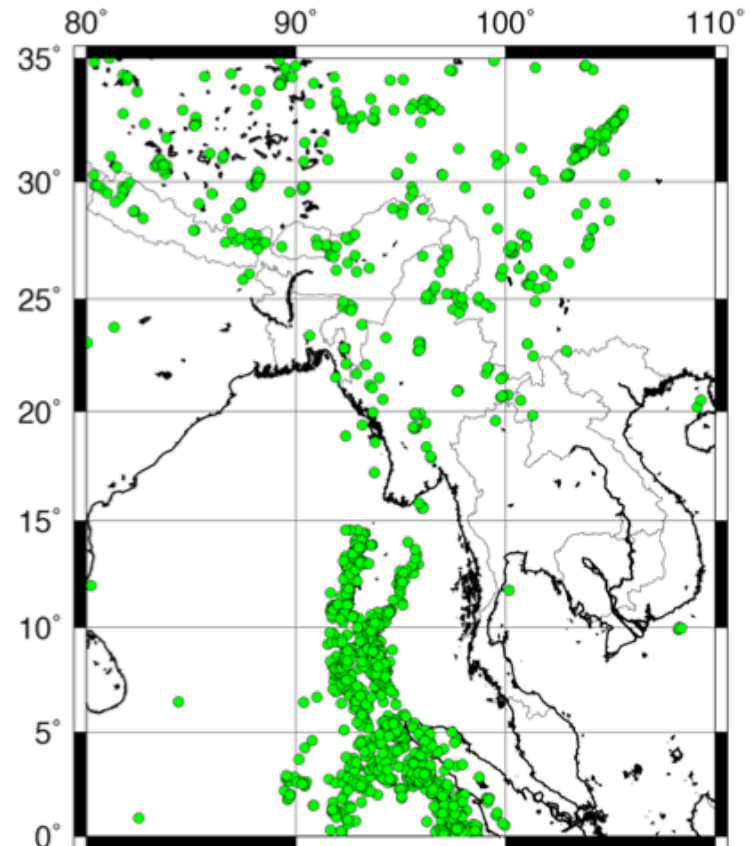
[www.matthewwherman.com/tutorials/eqs.xy](http://www.matthewwherman.com/tutorials/eqs.xy)

This text file contains magnitude 5 and greater earthquakes in the vicinity of the mapped region from 1996 to 2006.

*If for some reason the link is broken, save this subset of events as a file called eqs.xy. The columns are:*

*lon lat depth magnitude*

```
101.903 25.523 10 5.2  
95.432 2.901 36.5 5.2  
96.117 2.633 25.9 5  
91.78 8.264 35 5.1  
95.739 6.443 264 5  
93.9 1.983 10 5  
105.171 32.302 37.9 5  
93.746 3.279 10 5  
92.746 3.753 10 5.3
```



# Plotting earthquake data using psxy

```
gmt psxy eqs.xy $PROJ $LIMS -Sc0.15i -W0.5p -Ggreen \  
-O -K >> $PSFILE
```

## `eqs.xy`

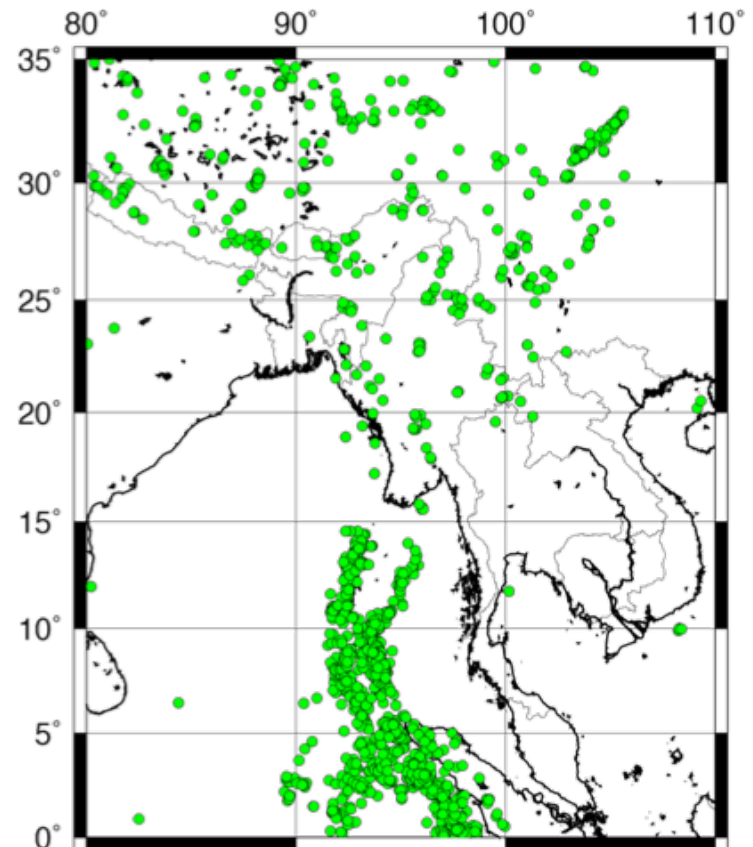
The input file for `psxy` is the earthquake file you just downloaded.

## `-Sc0.15i`

We generally plot earthquakes as circles, which are specified with `-Sc`. The diameter of the symbols is 0.15 inches.

## `-W0.5p -Ggreen`

Symbol outline and fill are chosen the same as usual. You can set the fill to any color you choose (we are going to change it soon), but here it is set to green with `-Ggreen`.



# Plotting earthquake data using psxy

```
gmt psxy eqs.xy $PROJ $LIMS -Scc -W0.5p -Ceq.cpt \  
-O -K >> $PSFILE
```

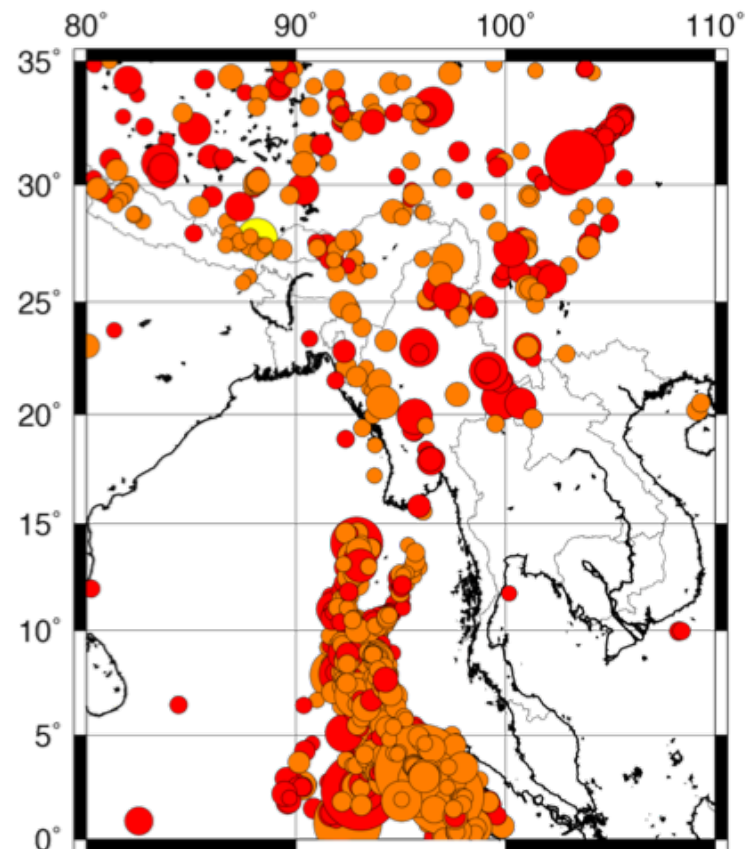
The data file also contains information about the event depth (3<sup>rd</sup> column) and magnitude (4<sup>th</sup> column). We can visualize these by adjusting the symbol color and size.

## **-Ceq.cpt**

(Changed from **-Ggreen**) This option tells **psxy** you want to color the symbols based on the value of the 3<sup>rd</sup> column of the input data, using the coloring scheme defined in the file **eq.cpt**. An earthquake depth color palette file might look like this (save it as **eq.cpt**):

0	255/0/0	10	255/0/0
10	255/155/0	25	255/155/0
25	0/255/0	50	0/255/0
50	0/0/255	100	0/0/255
B	255/0/0		
F	0/0/255		

*Do not worry too much about the details of making color palette files. They will be in another lesson.*



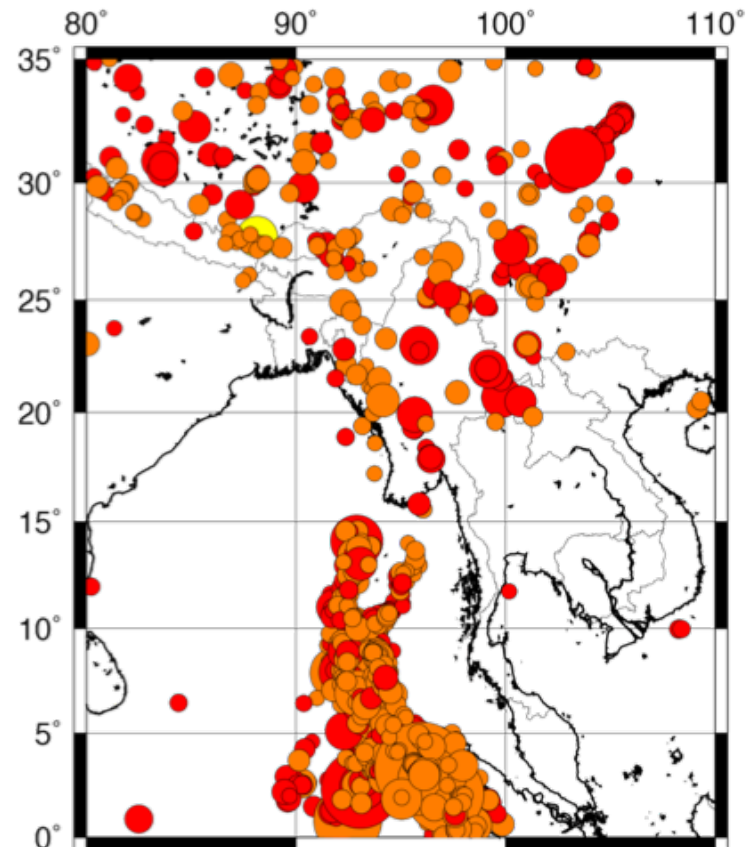
# Plotting earthquake data using psxy

```
gmt psxy eqs.xy $PROJ $LIMS -Scc -W0.5p -Ceq.cpt \  
-O -K >> $PSFILE
```

The data file also contains information about the event depth (3<sup>rd</sup> column) and magnitude (4<sup>th</sup> column). We can visualize these by adjusting the symbol color and size.

## **-Scc**

(Changed from **-Sc0.15i**) Instead of explicitly defining the size of the symbol in the **-S** option, we do not specify a value here, only a unit (**c** for cm). Doing this tells **psxy** to scale the the symbol dimension to be the value of the size column in cm.





# Plotting earthquake data using psxy

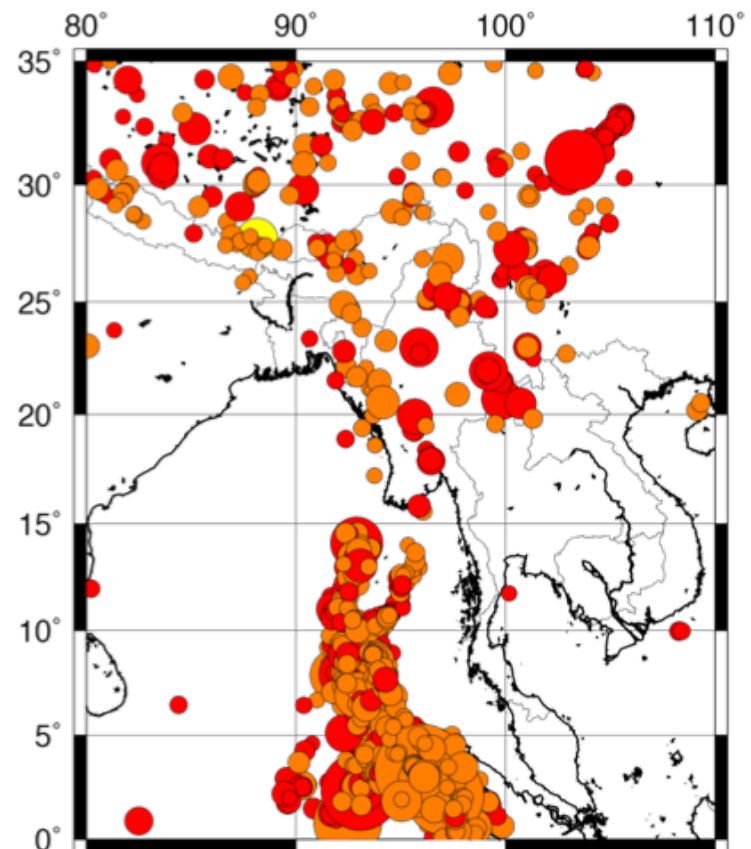
```
gmt psxy eqs.xy $PROJ $LIMS -Scc -W0.5p -Ceq.cpt \  
-O -K >> $PSFILE
```

The data file also contains information about the event depth (3<sup>rd</sup> column) and magnitude (4<sup>th</sup> column). We can visualize these by adjusting the symbol color and size.

If a color palette is defined (as it is here with **-Ceq.cpt**), then the color value *must* be in the 3<sup>rd</sup> column of the input, immediately after the x and y values. Then the symbol size is in the 4<sup>th</sup> column.

If you do not define a color palette, then the value in the 3<sup>rd</sup> column of the input is taken as the size and the color can be specified with **-G**.

To manipulate columns in files, use `awk` (see our Beginner's Guide to Awk online!).



At this point, your script should look something like this:

```
#!/bin/sh
PROJ="-JM15c"
LIMS="-R80/110/0/35"
PSFILE="earthquakes.ps"
gmt pscoast $PROJ $LIMS -W1p -Dc -N1/0.5p -K > $PSFILE
gmt psbasemap $PROJ $LIMS -Bxa10g10 -Bya5g5 -BWeSn \
-K -O >> $PSFILE
gmt psxy stars.xy $PROJ $LIMS -Sa0.3i -W0.25p -Gred \
-O -K >> $PSFILE
gmt psxy eqs.xy $PROJ $LIMS -Scc -W0.5p -Ceq.cpt \
-O -K >> $PSFILE
```

# Recap of PSXY

- **-S** indicates symbol plotting
- **-W** and **-G** specify pen and color of symbols
- The size of the symbol can be specified with **-S** or with the input file
- The color of the symbol can depend on the file if you make a color palette file and use the **-C** option

# Introduction to GMT (Part 1)

- Complete!
- Next up:
  - Plotting raster images (**xyz2grd**, **surface**, **grdcontour**, and **grdimage**)
  - Creating/drawing color scales (**makecpt** and **psscale**)
  - Writing text (**pstext**)