Scenario: You've discovered that one of the data files you were given by your Chief Scientist is a navigation file. How do you examine and clean these data?

One of the first data processing software packages you will use when you start playing with data is the Generic Mapping Tools (or GMT), which was developed by UH’s own Paul Wessel (and his partner in crime Walter Smith). GMT has dozens of programs that are useful for processing and displaying 1-, 2- and 3-D data, and it is very well documented. There’s a terrific GMT reference guide (cookbook) that you can access on the web at: www.gmt.edu. The best part about GMT is that it’s open source software, so you can modify the code and even contribute new functionality to the package. Furthermore, many, many academic institutions use the GMT tools to make gridded data files and PostScript plots, so the output from GMT is pretty transportable. Almost all ships in the academic fleet have computers with the GMT system installed – you can even download versions for your Mac PowerBooks.

The make a quick plot of a navigation file, use the GMT command `psxy`. This command creates a PostScript rendering of x,y pairs of data. Recall the format of our navigation file, `Lecture01.nav`:

```plaintext
1999 132 04 05 48 000 *nav 84.620350 10.78654
1999 132 04 11 48 000 *nav 84.593900 10.74714
1999 132 04 17 48 000 *nav 84.567680 10.70505
1999 132 04 23 48 000 *nav 84.541600 10.66544
1999 132 04 29 48 000 *nav 84.515050 10.63230
1999 132 04 35 48 000 *nav 84.488350 10.59917
1999 132 04 41 48 000 *nav 84.461680 10.57346
1999 132 04 47 48 000 *nav 84.434870 10.54607
1999 132 04 53 48 000 *nav 84.408070 10.51882
```

The “x,y” data in this file are the longitudes (found in column 9) and the latitudes (found in column 8). So, the first thing we need to do is strip this information out of the file. This is accomplished using the UNIX command `awk`:

```plaintext
awk '{print $9,$8}' | more
```

yields:

```plaintext
10.78654 84.620350
10.74714 84.593900
10.70505 84.567680
10.66544 84.541600
10.63230 84.515050
10.59917 84.488350
10.57346 84.461680
10.54607 84.434870
10.51882 84.408070
```
If we pipe these data into the psxy command instead, we can generate a plot:

```bash
awk '{print $9,$8}' | psxy -JL15/82.9/82.25/83.5/7 -R10/20/82.4/83.4 -Ba10f5/a30mf15m \
  -W3/0/0/0 -V >> raw_nav.ps
```

The command line arguments are described in the psxy manual page. In a nutshell:

- `-J` is used to determine the map projection. `-JL15/82.9` means use a Lambert Conic Conformal projection centered about 15° (E) longitude and 82.9° (N) latitude. By capitalizing the “L” we are requesting that the largest dimension of the plot be “7”, the number specified at the end of the `-JL` argument. The two intervening numbers, 82.25 and 83.5 are the latitudes at which the scale will be “true.”

- `-R` is used to specify the plot boundaries. In this case the plot will range from longitude 10° (E) to 20° (E) and 82.4° (N) to 83.4° (N).

- `-B` sets the annotations along the plot margin. In this example “a10f5” specifies that longitudes are to be annotated every 10° with a change in the frame every 5°; “a30mf15m” likewise means that the latitudes should be annotated every 30’ with frame changes every 15’.

The resulting plot looks like this:
Suppose we find out that this plot actually represents the data for a ship, but we need to plot the navigation for a towfish that was pulled on a cable behind the ship. As can be seen in the figure at right, it can take some time for a turn experienced by the ship to translate to the towfish behind it.

Figure 1 – Schematic showing ship (black) and the towfish behind it at two points during a 180° turn. As the ship approaches the first 90° turn the towfish is following directly behind it, but after the turn the towfish’s path will be a combination of its forward momentum and the pull to the right that the ship is exerting on it. Hence, the ship’s path is described by the black line, but the red line is a more accurate representation of the towfish path.

Before we attempt to filter the data to smooth them, we need to remember some basic statistics and the concept of mean (or average) versus median (or middle). In the frequency distribution plot below:

The **mode** is the value that occurs with the greatest frequency.  
The **median** is the value midway in the frequency distribution. That is, one-half of the area under the distribution curve is to the right of the median and one-half is to the left.  
The **mean** is the arithmetic average, or the sum of all observations divided by the number of observations.

There are other types of filters available in GMT as well, but for the purposes of our class let’s do a simple experiment where we see what the differences are between using a median and a mean (averaging or boxcar) filter. In order to do this we will use the GMT program **filter1d**.
As its name implies, **filter1d** operates on a one-dimensional data stream(s). In our case this will be longitude or latitude, but not both at the same time. We thus need to strip one column at a time from our original navigation file and then merge them back together to make more plots. This can be accomplished using the UNIX **awk** command as before, but a more direct approach is to use the UNIX commands **cut** and **paste**. These commands allow columns to be stripped from datafiles and then merged back together in row-wise order (i.e., column1/row1 column2/row1, column2/row2 column2/row2, etc).

Here is the exercise that we completed in class that allowed us to filter the Lecture01.nav longitudes and latitudes in twelve different ways, using standard boxcar –(Fb) and median –(Fm) filters and using robust boxcar (-FB) and median –(FM) filters with filter lengths of 3, 7 and 11 data points. Note that because filter1d requires a “time variable” we will use the line number as a proxy time stamp.

Filter the longitudes:

```
awk '{print NR,$9}' Lecture01.nav | filter1d -Fm3 | cut -f2 > mef1d.lon.m03.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -Fb3 | cut -f2 > mef1d.lon.b03.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FM3 | cut -f2 > mef1d.lon.M03.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FB3 | cut -f2 > mef1d.lon.B03.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -Fm7 | cut -f2 > mef1d.lon.m07.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -Fb7 | cut -f2 > mef1d.lon.b07.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FM7 | cut -f2 > mef1d.lon.M07.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FB7 | cut -f2 > mef1d.lon.B07.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -Fm11 | cut -f2 > mef1d.lon.m11.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -Fb11 | cut -f2 > mef1d.lon.b11.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FM11 | cut -f2 > mef1d.lon.M11.nav
awk '{print NR,$9}' Lecture01.nav | filter1d -FB11 | cut -f2 > mef1d.lon.B11.nav
```

Filter the latitudes:

```
awk '{print NR,$8}' Lecture01.nav | filter1d -Fm3 | cut -f2 > mef1d.lat.m03.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -Fb3 | cut -f2 > mef1d.lat.b03.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FM3 | cut -f2 > mef1d.lat.M03.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FB3 | cut -f2 > mef1d.lat.B03.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -Fm7 | cut -f2 > mef1d.lat.m07.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -Fb7 | cut -f2 > mef1d.lat.b07.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FM7 | cut -f2 > mef1d.lat.M07.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FB7 | cut -f2 > mef1d.lat.B07.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -Fm11 | cut -f2 > mef1d.lat.m11.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -Fb11 | cut -f2 > mef1d.lat.b11.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FM11 | cut -f2 > mef1d.lat.M11.nav
awk '{print NR,$8}' Lecture01.nav | filter1d -FB11 | cut -f2 > mef1d.lat.B11.nav
```

Merge the filtered data back together:

```
paste mef1d.lon.m03.nav mef1d.lat.m03.nav > mef1d.m03.nav
paste mef1d.lon.b03.nav mef1d.lat.b03.nav > mef1d.b03.nav
paste mef1d.lon.M03.nav mef1d.lat.M03.nav > mef1d.M03.nav
paste mef1d.lon.B03.nav mef1d.lat.B03.nav > mef1d.B03.nav
```
paste mef1d.lon.m07.nav mef1d.lat.m07.nav > mef1d.m07.nav
paste mef1d.lon.b07.nav mef1d.lat.b07.nav > mef1d.b07.nav
paste mef1d.lon.M07.nav mef1d.lat.M07.nav > mef1d.M07.nav
paste mef1d.lon.B07.nav mef1d.lat.B07.nav > mef1d.B07.nav
paste mef1d.lon.m11.nav mef1d.lat.m11.nav > mef1d.m11.nav
paste mef1d.lon.b11.nav mef1d.lat.b11.nav > mef1d.b11.nav
paste mef1d.lon.M11.nav mef1d.lat.M11.nav > mef1d.M11.nav
paste mef1d.lon.B11.nav mef1d.lat.B11.nav > mef1d.B11.nav

Now if we modify the command that we used to make our “raw” navigation map, we can compare the results of the various filters:

```
#!/bin/csh -f
#
set Rp=10/20/82.4/83.4
set Jp=L15/82.9/82.25/83.5/7
set Bp=a10f5:."median":/a30mf15m
#
p psyxy mef1d.m11.nav -J$Jp -R$Rp -B$Bp -M -W3/255/0/0 -V -K >! mef1d.m.ps
psxy mef1d.m07.nav -J$Jp -R$Rp -B$Bp -M -W3/0/255/0 -V -O -K >> mef1d.m.ps
psxy mef1d.m03.nav -J$Jp -R$Rp -B$Bp -M -W3/0/0/255 -V -O >> mef1d.m.ps
#
set Bp=a10f5:."Median":/a30mf15m
#
psxy mef1d.M07.nav -J$Jp -R$Rp -B$Bp -M -W3/0/255/0 -V -O -K >> mef1d.M.ps
psxy mef1d.M03.nav -J$Jp -R$Rp -B$Bp -M -W3/0/0/255 -V -O >> mef1d.M.ps
#
set Bp=a10f5:."boxcar":/a30mf15m
#
p psyxy mef1d.b11.nav -J$Jp -R$Rp -B$Bp -M -W3/255/0/0 -V -K >! mef1d.b.ps
psxy mef1d.b07.nav -J$Jp -R$Rp -B$Bp -M -W3/0/255/0 -V -O -K >> mef1d.b.ps
psxy mef1d.b03.nav -J$Jp -R$Rp -B$Bp -M -W3/0/0/255 -V -O >> mef1d.b.ps
#
set Bp=a10f5:."Boxcar":/a30mf15m
#
psxy mef1d.B07.nav -J$Jp -R$Rp -B$Bp -M -W3/0/255/0 -V -O -K >> mef1d.B.ps
psxy mef1d.B03.nav -J$Jp -R$Rp -B$Bp -M -W3/0/0/255 -V -O >> mef1d.B.ps
```

In this case I have type the commands into an executable shell so that I can quickly edit the similar command lines and generate all of the plots by making the script executable using the `chmod` command, and then executing this script. The results are shown on the following pages.
boxcar

median