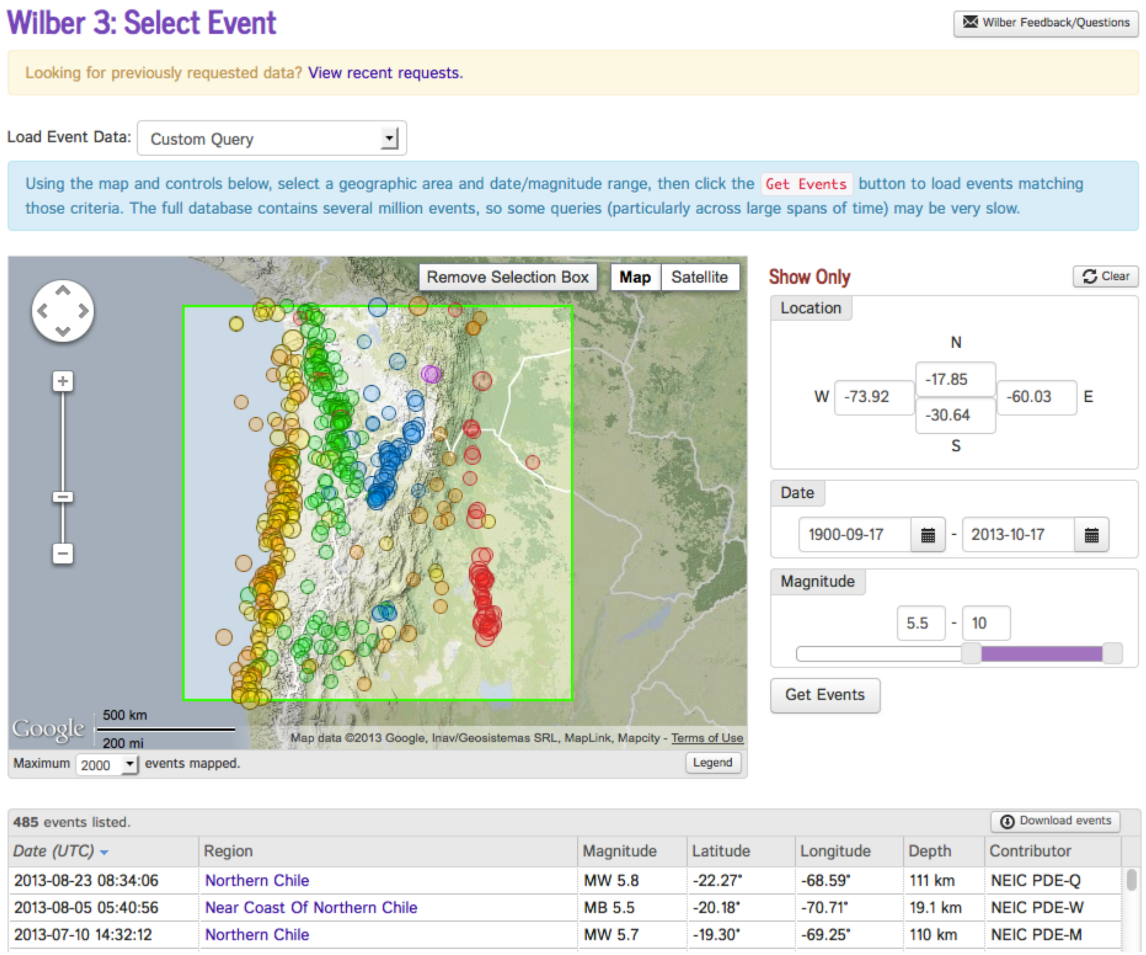
Data Analysis in Geophysics
ESCI 7205

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

Tu/Th - 13:00-14:30
CERI MAC (or STUDENT) LAB
Lab – 15, 17/10/13

Go to IRIS DMC (http://www.iris.edu/wilber3/find_event) and download a list of events with M≥5.5 for the region shown below (the USGS web pages were shut down when I prepared the class). I selected the data using the selection box (not typing in the weird lat/long limits, so you should get your own limits).


Wilber 3: Select Event

Looking for previously requested data? View recent requests.

Load Event Data: Custom Query

Using the map and controls below, select a geographic area and date/magnitude range, then click the `Get Events` button to load events matching those criteria. The full database contains several million events, so some queries (particularly across large spans of time) may be very slow.

Show Only

Location: N -17.85, S -30.64, W -73.92, E -60.03

Date: 1900-09-17 - 2013-10-17

Magnitude: 5.5 - 10

Maximum 2000 events mapped.

485 events listed.

Date (UTC)	Region	Magnitude	Latitude	Longitude	Depth	Contributor
2013-08-23 08:34:06	Northern Chile	MW 5.8	-22.27°	-68.59°	111 km	NEIC PDE-Q
2013-08-05 05:40:56	Near Coast Of Northern Chile	MB 5.5	-20.18°	-70.71°	19.1 km	NEIC PDE-W
2013-07-10 14:32:12	Northern Chile	MW 5.7	-19.30°	-69.25°	110 km	NEIC PDE-M

Wilber Feedback/Questions
Remove Selection Box
Map
Satellite
Clear
Get Events
500 km
200 mi
Map data ©2013 Google, Inav/Geosistemas SRL, MapLink, Mapcity - Terms of Use
Legend
Download events

The “download events” button brings up a list of events.

```
#EventID | Time | Latitude | Longitude | Depth/km | Author | Catalog | Contributor | ContributorID | MagType | Magnitude | MagAuthor | EventLocationName
4224414|2013-08-23T08:34:06|-22.274|-68.593|111.0|NEIC|NEIC PDE|NEIC PDE-Q||MW|5.8|WCMT|NORTHERN CHILE
4223021|2013-08-05T05:40:56|-20.181|-70.712|19.1|NEIC|NEIC PDE|NEIC PDE-W||MB|5.5|NEIC|NEAR COAST OF NORTHERN CHILE
4221773|2013-07-10T14:32:12|-19.302|-69.248|110.0|NEIC|NEIC PDE|NEIC PDE-M||MW|5.7|GCMT|NORTHERN CHILE
4221374|2013-07-02T20:04:55|-23.851|-66.516|198.9|NEIC|NEIC PDE|NEIC PDE-M||MW|5.6|WCMT|JUJUY PROVINCE, ARGENTINA
4219882|2013-06-08T12:25:08|-22.588|-66.691|215.0|NEIC|NEIC PDE|NEIC PDE-M||MW|5.6|WCMT|JUJUY PROVINCE, ARGENTINA
4212015|2013-02-22T12:01:58|-27.932|-63.097|575.2|NEIC|NEIC PDE|NEIC PDE-M||MW|6.1|WCMT|SANTIAGO DEL ESTERO PROV., ARG.
3278379|2011-02-21T06:58:36|-27.1293|-64.698|13.5|ISC|ISC|ISC|01963232|MW|5.6|GCMT|SANTIAGO DEL ESTERO PROV., ARG.
4252115|2011-01-01T09:59:37|-26.8683|-63.2194|600.0|ISC|ISC|ISC|01764720|mb|5.7|ISC|SANTIAGO DEL ESTERO PROV., ARG.
3207531|2011-01-01T09:56:58|-26.8513|-63.2373|584.3|ISC|ISC|ISC|01764719|MW|7.0|GCMT|SANTIAGO DEL ESTERO PROV., ARG.
3180564|2010-10-23T01:38:15|-29.5649|-71.1702|54.9|ISC|ISC|ISC|01345818|MW|5.6|GCMT|NEAR COAST OF CENTRAL CHILE
3180560|2010-10-22T19:31:37|-20.9722|-68.5707|130.9|ISC|ISC|ISC|01345795|MW|5.8|GCMT|CHILE-BOLIVIA BORDER REGION
```

We will now plot this with color as a function of depth and symbol size as a function of magnitude.

First look at input data file to find automatic way to process it. Look at header line and then data lines to figure out how to make “free format” “fields” for something like awk.

Fields come separated by “|”, and some are empty – but this is OK since the “|” is the separater. (Missing fields typically a big problem for awk/awk/gawk.)

Check getting magnitude

```
507 $ awk 'BEGIN {FS="|"} {print $11}' fdsnws-event_2013-10-17T03_33_41.txt
 Magnitude
5.8
5.5
5.7
5.6
...
```

No missing values from looking at output.

What are we doing with awk?

We are executing the awk commands from the command line. The awk commands are contained in single quotes. The use of quotes is complicated.

Single quotes are “strong” and “protect” everything inside the quotes from the shell, which interprets them as regular old characters.

Double quotes are “weaker”. They do the same as single quotes except that they allow for variable expansion – so if you have a $MYVARI inside double quotes it gets replaced by the value of MYVARI, and allow for command substitution (the backwards quotes), which behaves normally.

BEGIN followed by a block of code, the stuff in the braces, to be done before reading the file (there is a similar command, END, for a block of code to be executed when done reading file).

Here we are redefining the field separator, FS, to be the vertical bar “|”, rather than a space (based on what we found when we looked at the data we downloaded).

If you want several field separators: space, comma, semicolon for example, you put them all in the double quotes “ ,;”. Field separators are a single character, e.g. in the example the field separator is not the three characters in a row, it is any one of them.

So all we are doing here is redefining the field separator to “|” and then printing out the 11th field. The “action” is given in the braces, which represents a block of code (these can be nested { some lines {some more lines}} when needed). (the $11 here is how to specify that, $0 is the whole line.)

Do same for depth, which is in the 5th field.

```
545 $ awk 'BEGIN {FS="|"} {print $5}' fdsnws-event_2013-10-17T03_33_41.txt
 Depth/km
111.0
19.1
110.0


198.9
215.0
575.2
```

Now we have some missing values. Most unix programs read “free format” not a strictly column based (“Fortranny”, or “holorith” based, empty field interpreted as 0).

How to fix this?

A) Throw out the few events with no depth.
B) Replace blanks with zero.

Lets do it both ways to see how to do it.

While it is probably quicker to edit the file by hand if you only have to do it once, that is not an acceptable solution if you need to make this map every day.

(There are only 3 of them! But that is enough to break the program, so they have to be fixed.)

First – throw them out. Use awk to test the value of depth and use the result of the test to print the line out or not.

```
nawk 'BEGIN {FS="|"} !($5 ~ /^0?$/) {print $5}' fdsnws-event_2013-10-17T03_33_41.txt
 Depth/km
111.0
19.1
110.0
```

```
198.9
215.0
575.2
```

Now we are introducing a new feature of awk – testing something on the line and using the result to determine how to process the line.

The "/" is used to define the start and stop of the test. Here we are testing that field 5 ($5) is empty (nothing in it, field 5 is defined by the two separators without any character between them). The string to test against is contained in double quotes. Once the test is done, it is negated by the leading exclamation point. So this will print out the lines that don't have an empty 5th field.

Test to see we really removed the three lines

```
609 $ nawk 'BEGIN {FS="|"} !($5 ~ /^0?$/) {print $5}' \
fdsnws-event_2013-10-17T03_33_41.txt | wc
     483     483    2604
610 $ wc fdsnws-event_2013-10-17T03_33_41.txt
     486    1854   52355 fdsnws-event_2013-10-17T03_33_41.txt
```

To make life a bit more interesting we can use a variable in the test (or pretty much anywhere else in the awk program).

```
611 $ TF='^0?$'
613 $ nawk 'BEGIN {FS="|"} !($5 ~ /'$TF'/) {print $5}' \
fdsnws-event_2013-10-17T03_33_41.txt
 Depth/km
111.0
19.1
110.0
198.9
215.0
```

Now I’ve defined a variable TF that has the test field in it. I used the strong, single quote so the weaker, double quote does not get interpreted by the shell.

Now we put on our unix thinking caps. Everything is going along fine till we come to the '$TF'. The single quote “closes” the protection provided by the first one, so the $TF is no longer protected. So the shell replaces it by the value of the variable TF, which is “”. The next single quote turns the protection back on, and everything else till the next single quote is protected from the shell.

How did we find the “test” value? In the old days you needed a UNIX wizard. Now we have the internet. Even then, it usually takes some effort.

If we were looking to replace or find something something easy – a “regular” character string we would just have put that in (look in the notes from last time where we were looking for a line that started “ PDE” (the ^ in that example means beginning of line, also note the space before the string PDE). But now we are looking for an empty field. This is a bit tricky.

Unix has a way to look for just about anything. It is very powerful. It is also very confusing.

I did a google search for “awk test for empty field” and the following link had how to do it.

http://stackoverflow.com/questions/10935049/search-empty-value-in-awk

How to define what we are searching for (in terms of strings) is handled by something called “regular expressions”. They are rife with the symbols found over the number keys. They are also complicated because they have to handle every possible case that can happen. For example – removing empty lines – an empty line can have nothing (two consecutive carriage returns) or have any number of spaces (nothing to us, but something to the computer).

It is very easy now to find help for searching for things using the web. In the old days you had to find a unix wizard (unix is NOT free!).

In our shell script we could store this output in another file, or if we only need it once, process it on the fly. We will do it on the fly.

To replace missing depths with a value (0) use the stream editor, sed.

```
sed 's/||/|0|/g' fdsnws-event_2013-10-17T03_33_41.txt
```

sed uses the same commands as vi (actually, following the unix philosophy, both sed and vi use the same commands as ed). So this says find the string "||" and replace it with the string "|0|" and do it globally with respect to individual input lines (else it just gets the first instance if there are more than one) and with respect to the whole file (else it just does it on the first line it finds it on). Since we already know that in the input file we got from IRIS it only occurs once when it occurs, we don't really need the g.

We will use the "throw them out" method here.

Next we have to print out the input that gmt wants. Looking at the psxy man page we will need

Lat, lon, the field that control the color = depth, and the field that controls the symbol size = magnitude. The symbol size has to be the last field going into gmt. Look at the input data to decide which fields (columns) we need to print out.

```
MAGRESCALE=4.5
nawk 'BEGIN {FS="|"} !($5 ~ /^0?$/) {print $3, $4, $5, ($11-'$MAGRESCALE')^2/8}' \
fdsnws-event_2013-10-17T03_33_41.txt
```

Notice that we are printing out fields 3, 4, the lat and lon, and 5, the depth, and then doing something with field 11, the magnitude. Assuming the data in the field is a number we can do calculations with it. AWK has a limited number of mathematical operations and functions it can do (+,-,*,/, ^, sin, etc.). Our data set has events M≥5.5 and the largest is 8. We want to use the magnitude to scale the symbol size. A nice looking range of sizes can be obtained by subtracting 4.5, so all the values are ≥1, squaring this, and then dividing by a scale factor. This is to spread out the values a bit for the sizing.

Look in the notes from last time to see what to do if the magnitude was reported as zero (what we would have had to do if we put zero in, instead of throwing them away). We still want to plot those points, but would not see a symbol with zero size. We use an inline-if statement construction in awk (also appears in C, bash, etc.)

```
($9>0)?(($9-‘$MAGRESCALE’)^2/8):"0.01"
```

This says to compare the value in field 9 to zero and then, if the result of the test is true do what is between the "?" and the ":" (see above), else do what is after the ":", print out the string "0.01". We need the parentheses to group the test and the arithmetic.

One can do multiple tests by using logical or ("||") and logical and ("&&").

```
nawk '$3<60&&($3!=0||$3!=33) …
nawk '/LHCL/||/AREQ/||/AZUL/ …
nawk '{print (-2100<$3&&$3<2100)?$0:$1,$2,"0"}'
nawk '$3<60&&($3!=0||$3!=33) {print $0}'
```

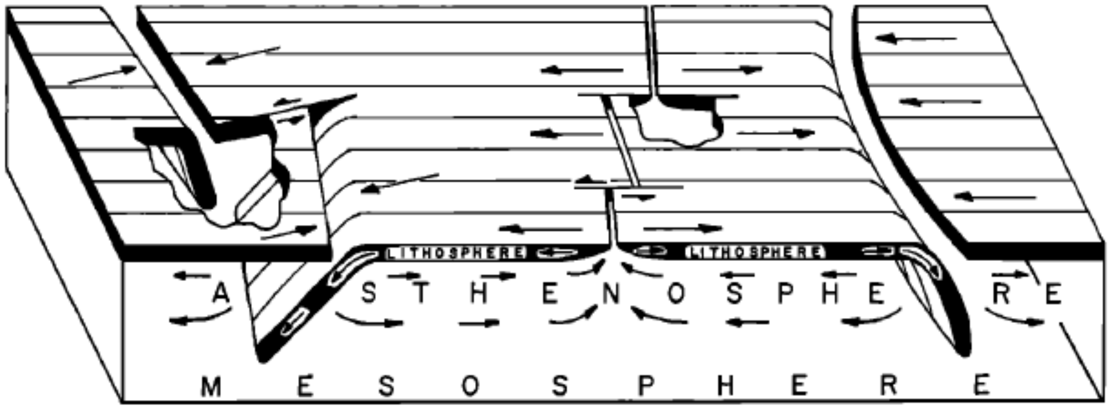
We also need them sorted by depth so the deeper ones appear "underneath" the shallower ones. This data set is small enough that we don't have a problem with lots of small earthquakes covering over the bigger ones, so we only have to deal with sorting by depth.

```
nawk 'BEGIN {FS="|"} !($5 ~ /^0?$/) {print $3, $4, $5, ($11-‘$MAGRESCALE’)^2/8}' \
fdsnws-event_2013-10-17T03_33_41.txt | sort -k 3 -n -r
```

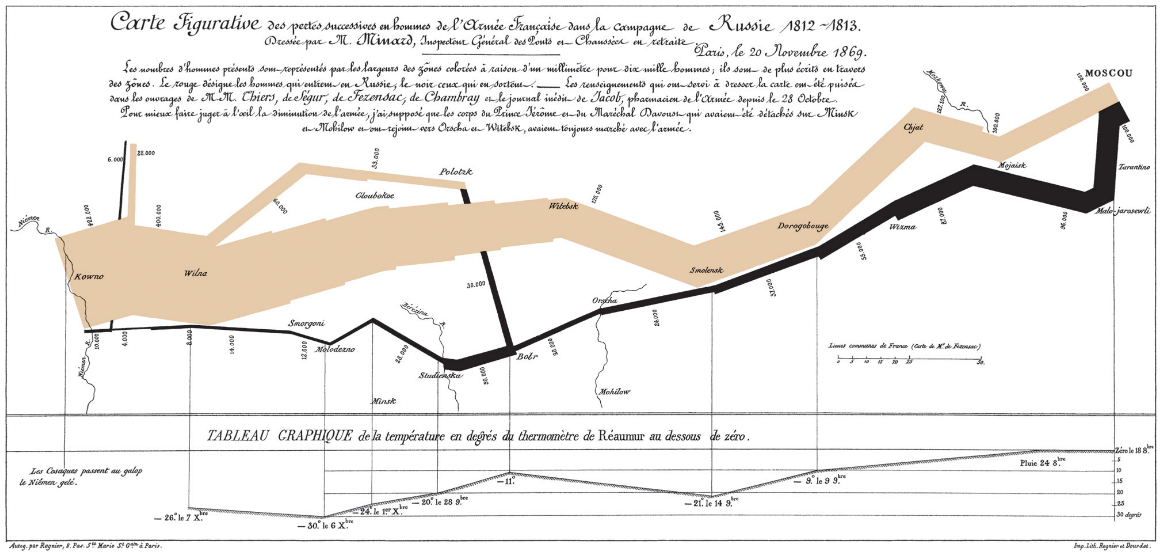
Finally we send (pipe) it to psxy and into the file that will have all the parts of the figure.

```
nawk 'BEGIN {FS="|"} !($5 ~ /^0?$/) {print $3, $4, $5, ($11-‘$MAGRESCALE’)^2/8'} \
fdsnws-event_2013-10-17T03_33_41.txt | sort -k 3 -n -r | psxy $REGION $PROJ  \
$SYMBOL $CPT $OUTLINE $LLSWAP $CONTINUE >> $OUTFILE
```

Note that you may have to make a few practice figures before you get how to present the information you want. Figures are important. Make a good figure and it will become iconic.



The only important plate tectonic interaction missing from the Isacks, Oliver and Sykes 1968 figure above was subduction of ridges and the associated slab windows in the down going plate. These guys were masters of graphical presentation.



Charles Minard's 1869 chart showing the losses in men, their movements, and the temperature of Napoleon's 1812 Russian campaign. Lithograph, 62 x 30 cm.
Source: http://en.wikipedia.org/wiki/File:Minard.png

Next make a cross section (and could also show where it will go on map). For this we will need to project the data – find its distance along a line on the map (modify the part above to plot this line – easy in our case since we need endpoints and is easy to get the far endpoint, in general is challenging).

Look up the gmt command "project" on the web. We need to state a starting point, azimuth and direction for the cross section. We also need to find the new limits to plot (can look at file produced by project or base it on what you already know about the data and your projection parameters).

So we will have to assign a bunch of new variables/values and build the commands to project and plot the data.