A *plotxy* Tutorial

Introduction

These notes aim to introduce you to the plotting program `plotxy`, written by our own Bob Parker. This program is easy to use, and also produces publication-quality output: something not found in many commercial packages. Another program that produces publication-quality plots is GMT, but it is much more complicated. Since `plotxy` was designed by and for practicing scientists, it focuses on the commonest types of scientific plots – though with additional effort, it can create many types of figures. And the program has been used for a long time, so it has been thoroughly debugged.

`Plotxy` reads a command file which tells it what to do, including reading from other files containing data. It writes to an output file, sending basic commands in the PostScript language. To make a picture from the PostScript file, you can send it straight to a printer, since all printers understand this language. if you want to view the picture on the screen, you will need to read the file into a PostScript viewer, the usual one being `ghostscript`, which you run by typing the command `gv`.

The graphics files produced by `plotxy` largely consist of two commands: one to draw a line from one place to another, and the other to fill an area. All the letters and symbols used are actually drawn from patterns stored in the software. This makes the output file easily interpretable by any PostScript device.

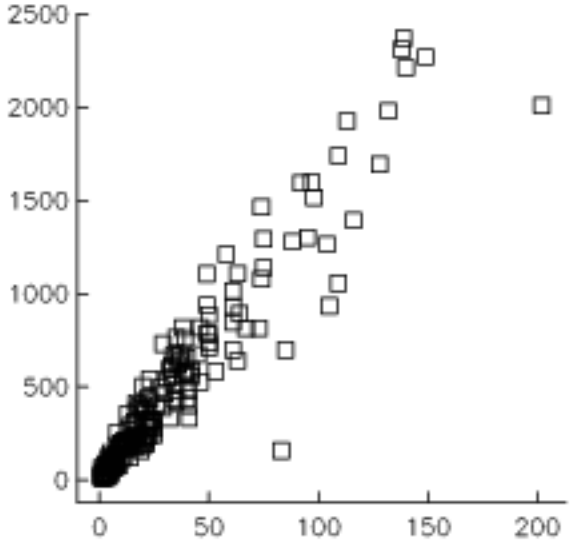
A Data Plot

We start by plotting some relevant data: the intervals, in miles and days, between Bob Parker having a flat tire on his bicycle (at least over the first 96,000 miles). The first few lines of the file look like:

```
 22  297
  2   28
  4   74
  2   38
 12  189
 74 1079
```

To plot these data, we create a file `pxy.tmp` with a text editor, which looks like

```
symbol square
file flats
read
plot
stop
```



We then run `plotxy`, list and dump the first few lines of the output file, and display it. As before, we show this using this font for what you type, and `this font` for what the computer types, including the prompt `computer%`

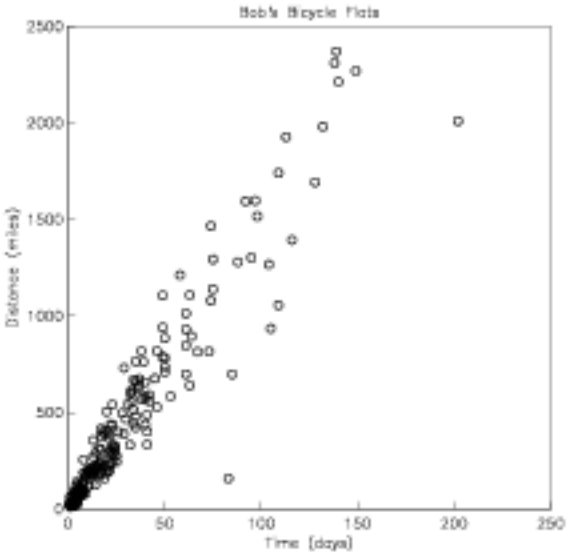
```
computer% cat pxy.tmp | plotxy
 Enter commands for graph 1

 ------------------------------------
 PostScript file written to: mypost
 ========== --------------------------
 Enter commands for graph 2
computer% ls -l mypost
-rw-r--r--    1 agnew    admin       21267 Nov  7 23:57 mypost
computer% head -6 mypost
%PS PostScript file
% Generated by plotxy
/k currentpoint stroke moveto newpath moveto def
/l lineto def
/m newpath moveto def
/n lineto currentpoint stroke moveto def
computer% gv -watch mypost
```

As you can see, the file `pxy.tmp` has only 5 commands: `symbol` says that we should plot symbols at the points given, rather than drawing lines between them, and specifies the type of symbol; `file` gives the name of the file to read; `read` causes it to be read into memory; `plot` causes the program to write PostScript to the output file, which has the default name `mypost`; and `stop` stops `plotxy`. When running, `plotxy` sends a short message to standard output. Then `ls` shows the size of the output file `mypost`, and `head -6` to shows the first few lines of PostScript. Then we run `gv` to view the file, setting the `-watch` flag so that the display will update if we modify the `mypost` file. We cannot, of course, run `gv` on this

page, but as the next best thing we show the completed plot in the margin, at about one-half actual size.

This plot is fine for a first look, but would not be for anything else, since there are no labels and no clue as to what it is about. So we add a title, and two axis labels; we also set the plot limits explicitly with `xlimit` and `ylimit`. These last two commands take three arguments: the length of the axis in inches, the lower limit, and the upper limit. We can set these limits however we want; anything that would be outside them is not plotted. To get different symbols we replace `square` with `circle`, and make the symbols smaller by explicitly setting their size in the second argument of the `symbol` command: in this case to 0.08 inches. We also use the `frame` command to draw a full box.

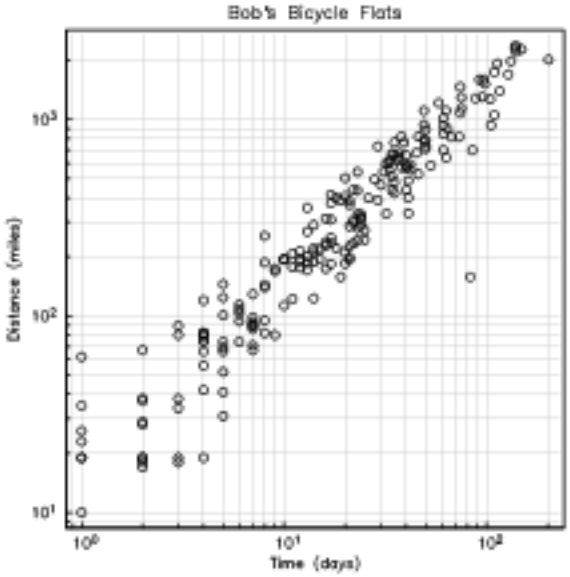



```
frame
title Bob's Bicycle Flats
xlabel Time (days)
ylabel Distance (miles)
xlimit 5 0 250
ylimit 5 0 2500
symbol circle .08
file flats
read
plot
stop
```

One reason `plotxy` can make professional-looking plots is the ability to vary the character size and line weight. The first is varied with the `character` command, which sets the character size in inches; the second is varied with the `weight` command, which sets the line weight in thousandths of an inch: a value of 10 corresponds to a line thickness of $0.010''$. In our next iteration we make the title larger than the axis labels, with line weights scaled to match. We use the `grid solid` option in `frame` to draw a light (gray) grid in the background of the plot. We set both axis scales to be logarithmic with the `logxy` command, which has the options `loglin` (x-axis log), `linlog` (y-axis log), `loglog` (both axes log). This spreads the data out more evenly over the area of the plot.

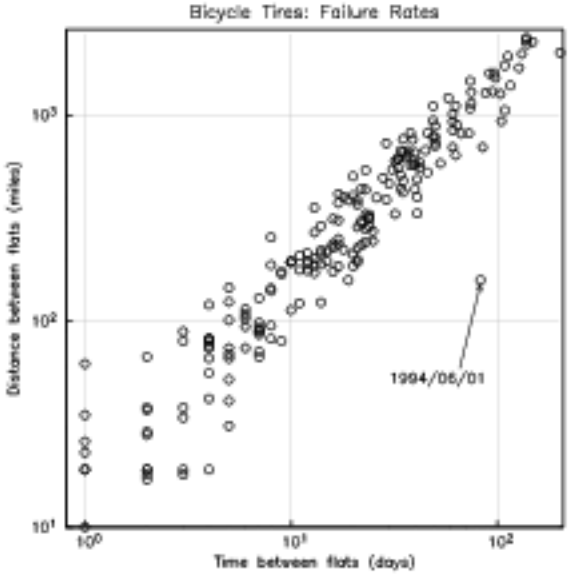
```
frame grid solid
character .12
weight 12
```

```
title Bob's Bicycle Flats
character .10
weight 10
logxy loglog
xlimit 5
ylimit 5
symbol circle .08
file flats
read
xlabel Time (days)
ylabel Distance (miles)
plot
stop
```



For the final version, we set the limits explicitly to make best use of the space available. We also add a third, negative, argument to the `xlimit` and `ylimit` commands to suppress all the labels that are not powers of ten. Using a log-log plot shows that there are a few outliers, and we label one of them with a `note` command. This consists of the word `note`, followed by one or two pairs of coordinates inside parentheses. If there are two pairs, as here, the first gives the location of an the head of an arrow whose tail is located near some text; the second coordinate pair gives the lower left corner of the text. The closing parenthesis is followed by the text of the note. The program then decides where to place the tail of the arrow: close to the text, but in a location closest to the arrowhead.

```
character .12
weight 12
title Bicycle Tires: Failure Rates
character .10
weight 10
frame grid solid
logxy loglog
xlimit 5 .81 210 -1
ylimit 5 10 2600 -1
symbol circle .08
file flats
read
note (83 150 30 50)1994/06/01
xlabel Time between flats (days)
```



```
ylabel Distance between flats (miles)
plot
stop
```

Plotting a Function

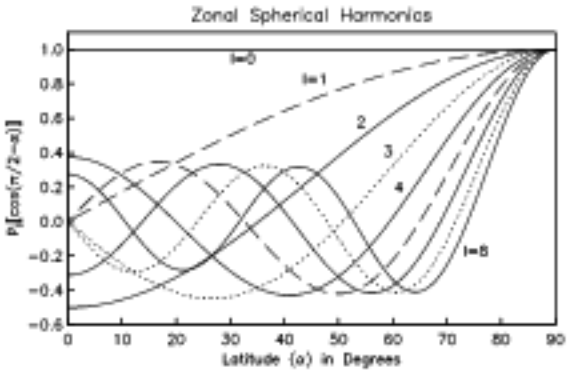
Next we see how to plot a function: actually, several functions, namely the Legendre functions $P_l(\cos(\alpha))$ which are the basis for the zonal spherical harmonics. We assume we have a table of these, the first few lines of which look like

```
90.0 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
89.9 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.9999
89.8 1.0000 1.0000 1.0000 1.0000 0.9999 0.9999 0.9999 0.9998 0.9998
89.7 1.0000 1.0000 1.0000 0.9999 0.9999 0.9998 0.9997 0.9996 0.9995
```

where the first column is the latitude ($\pi/2 - \alpha$, given in degrees), and the next nine columns are the values of P_l for l from 0 through 9. The commands to make the plot are:

```
frame
weight 12
character .12
title Zonal Spherical Harmonics
weight 10
character .1
xlimit 5 0 90
ylimit 3 -0.6 1.1
file ltable
%
% read column 1 for x, and columns 2 through 10 for y
%   with N from 0 through 9
%
mode 20 1 2
read
dash            %  dashed line
mode 20 1 3
read
dash 0 0        %  solid line
mode 20 1 4
read
dash .02 .05    %  dotted line
mode 20 1 5
read
```

```
dash 0 0        %  solid line
mode 20 1 6
read
dash
mode 20 1 7
read
dash 0 0
mode 20 1 8
read
dash .02 .05
mode 20 1 9
read
dash 0 0
mode 20 1 10
readxlabel Latitude (\alpha\) in Degrees
ylabel P\subl[cos(\pi\/2-\alpha\)]
note (30 .92)l=0
note (48 .80 r)l=1
note (55 .55 r)2
note (60 .38 r)3
note (62 .17 r)4
note (73 -.2)l=8
plot
stop
```



This introduces some new commands. First of all, any line starting with a `%` is a comment; indeed, anything on a line an after a `%` will be ignored (unless the `nocomment` command overrides this). To read two specific columns from a table, we use `mode 20 m n` before the `read`; the column numbers in this case run from 1 through 9. For example `mode 20 1 10` puts the first column as the x-values, and the tenth column as the y-values.

We have used the `dash` command to distinguish the lines. Setting just `dash` means that any lines plotted afterwards have dashes that are $0.1''$ long, separated by spaces of $0.2''$. Likewise, the command `dash 0.02 0.05` gives dashes $0.02''$ long, and spaces of $0.05''$; `dash 0 0` reverts to a solid line.

The `note` command demonstrates an additional element; a `r` at the end of the positioning point means that this will be the bottom right end of the text; a `c` would center the text.

One of the useful features of `plotxy` is being able to include math in a text string – and do so in a reasonably attrac-

tive way. The `ylabel` shows an example. To get a subscript you type `\sub{}`, which subscripts anything between the {}; to get a Greek letter, you type its name surrounded by \. There are also special codes for special characters; see the reference summary for more information.

Filling a Polygon

We next use *plotxy* to make a simple map, not so much because it is a great map-making program, as to show some additional features. In this case we have two files: one, of the coastline of the island of Hawai'i; the other, the locations and magnitudes of all earthquakes of magnitude 4 and above since 1948.[1] The coastline (file `coast`) comes as latitude-longitude pairs, and the earthquake file as latitude, longitude, and magnitude triplets (file `earthquakes`).

For a local map, we can make an adequate map projection by scaling the two axes so that a degree of latitude and a degree of longitude are scaled to relative lengths appropriate to the latitude. If the two latitudes are θ_S and θ_N, the two longitudes are ϕ_W and ϕ_E, and the length of the x-axis is x inches, the length of the y-axis is

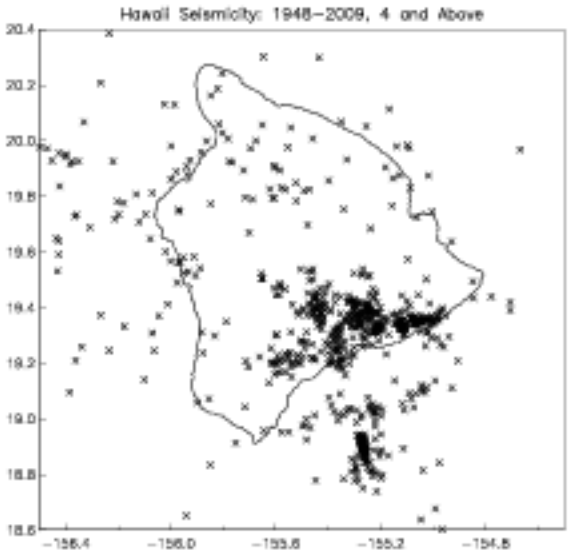
$$y = x * \left(\frac{\theta_N - \theta_S}{(\phi_W - \phi_E)\cos[1/2(\theta_N + \theta_S)]} \right)$$

So using this, and plotting the coastline as a line, and the earthquakes as crosses, our *plotxy* script is

```
frame
weight 12
character .12
title Hawaii Seismicity: 1948-2009, 4 and Above
weight 10
character .10
xlimit 6.00 -156.5 -154.5
ylimit 5.73 18.6 20.4
mode 20 2 1
file coastline
read
```

[1] Data before 1960 are from F. W. Klein and T. L. Wright (2000), Catalog of Hawaiian earthquakes, 1823-1959, *U. S. Geol. Surv. Prof. Pap.* **1623**, and since then from the ANSS website.

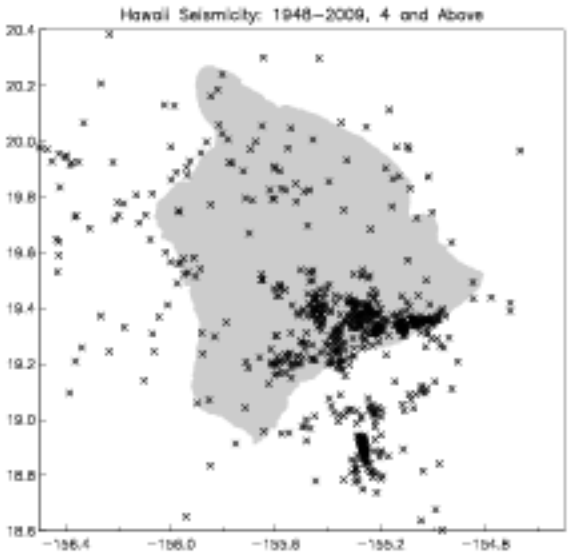
```
file earthquakes
symbol cross .1
mode 20 2 1
read
plot
stop
```



Note that we have used `mode 20 2 1`, which makes the latitude-longitude pairs into x-y pairs, and also to avoid reading the third column (magnitudes) of the `earthquakes` file; forgetting to do this is a common source of error.

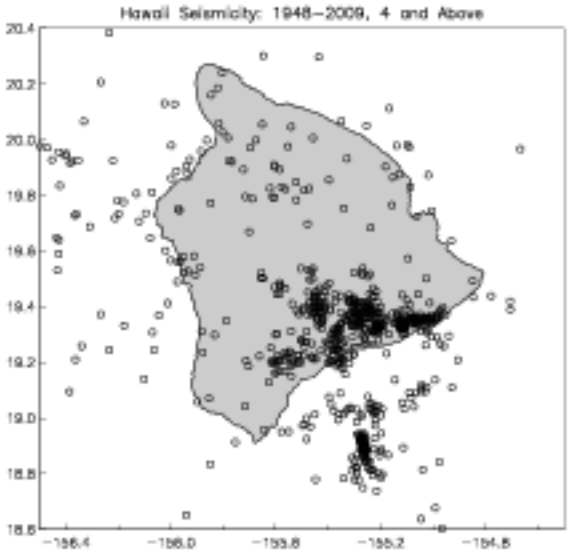
This map is a reasonable start, but the earthquakes and the coast are difficult to distinguish. We can do better by recognizing that the coastline is just a polygon (a very irregular one), so instead of drawing it, we can fill it. We also use the `color` command to make the fill color a pale gray, since if we did not it would be filled black, which would make the plot worse not better. We also need to do this before we plot the earthquakes, since if we did not this gray region would cover all the black symbols: an example of the general rule that series are plotted in order. And then we need to set `color black` again to make the earthquakes, and the frame and title, come out in black.

```
frame
weight 12
character .12
title Hawaii Seismicity: 1948-2009, 4 and Above
weight 10
character .10
xlimit 6.00 -156.5 -154.5
ylimit 5.73 18.6 20.4
mode 20 2 1
file coastline
color light gray
fill
read
color black
file earthquakes
symbol cross .1
mode 20 2 1
read
plot
stop
```



Our next variation on this black and white map, is done not so much as to show a feature of `plotxy` as to make a graphical point. We add to the plot above by plotting the coastline a second time, as a black line that divides the gray from the background white. Perhaps surprisingly, this makes the gray and white areas much more distinct – so much so, that the gray may seem darker than before. But it isn't. If you look at most professional maps, you will see this technique used: it relies on a feature of our visual system, namely that we (our eyes and our brains) are very good at spotting edges and other abrupt changes in tone.

```
frame
weight 12
character .12
title Hawaii Seismicity: 1948-2009, 4 and Above
weight 10
character .10
xlimit 6.00 -156.5 -154.5
ylimit 5.73 18.6 20.4
mode 20 2 1
%
%  Read coastline twice: once for fill,
%                        once for line
%
file coastline
color light gray
fill
read
color black
read
file earthquakes
symbol circle .07
mode 20 2 1
read
plot
stop
```



In all the plots so far, we have left the earthquake symbols the same size, but *plotxy* allows us to vary them. If we invoke `symbol` with two arguments, the second one will be taken to be the size of the symbol in inches, *unless* the value is larger than 10: something so large that *plotxy* assumes that we want symbols of variable size. To get different sizes, we read in a

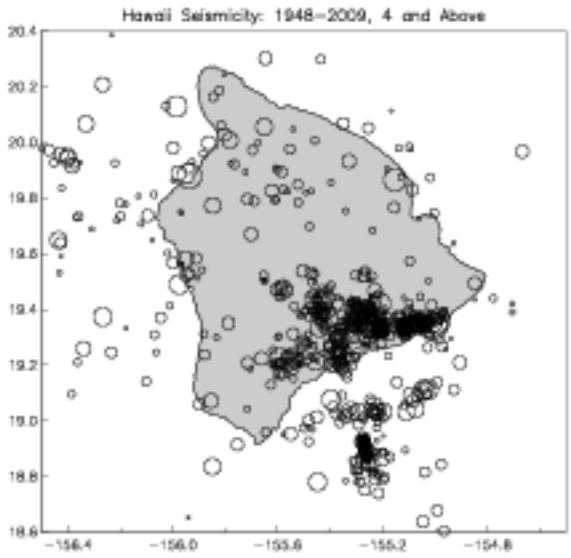
three-column file, in which the third column is the symbol size in inches, which is done by specifying `mode 3` (if there are only three columns) or `mode 30` (if we need to specify the columns explicitly.[2]

Of course, we do not want to use the actual magnitudes for the size in inches; instead we scale them so the smallest earthquakes (magnitude 4) become very small, and the area is proportional to the magnitude above 4; we also set it to make the largest earthquake in the catalog (magnitude 7.2) have unit size before scaling to inches. The following script uses `awk` to do the job; the scaling constant is set outside the `awk` part so that it can be, in the original script, `sc=$1`, allowing us to modify it easily to decide what works best.

```
cat earthquakes |\
awk '{
   printf"%s %s %.3f\n",$1,$2,sc*sqrt($3-3.95)/sqrt(7.2-3.95)
}' sc=.09 > tmpe
```

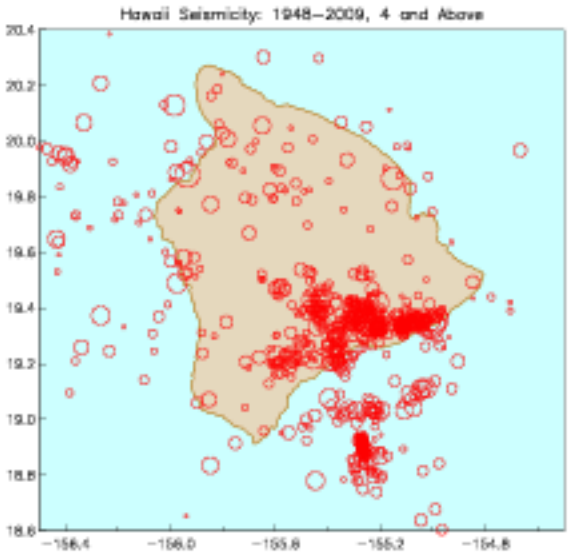
Then the *plotxy* file is

```
frame
weight 12
character .12
title Hawaii Seismicity: 1948-2009, 4 and Above
weight 10
character .10
xlimit 6.00 -156.5 -154.5
ylimit 5.73 18.6 20.4
mode 20 2 1
%
%  Read coastline twice: once for fill,
%                        once for line
%
file coastline
color light gray
fill
read
color black
read
%
```



[2] If the symbol size is less than 10 inches, this mode will plot the third column as error bars in y; see the reference sheet for other options.

```
%  Plot earthquakes with variable-sized circles
%
mode 30 2 1 3
symbol circle 20
file tmpe
read
plot
stop
```



Colors

It is easy to modify the above to make a colored plot. First, we use the `background` command to fill the body of the plot with some color suggesting ocean, and specify colors for the `fill` and subsequent line-drawing that will suggest land. It is good practice to use the lightest colors possible for coverage of large areas, so we have used `cyan` and `brown`, both modified by the adjective `pale`; we use regular `brown` for the coastline, and `red` for the earthquakes, not forgetting to reset to `black` before the end, so that the frame and axis numbers will be in that color.

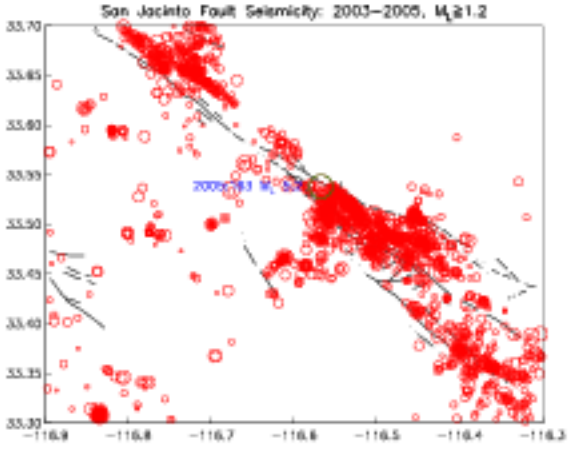
```
background pale cyan
frame
weight 12
character .12
title Hawaii Seismicity: 1948-2009, 4 and Above
weight 10
character .10
xlimit 6.00 -156.5 -154.5
ylimit 5.73 18.6 20.4
%
%  Read coastline twice: once for fill, once for line
%
file coastline
color pale brown
fill
read
color brown
read
%
%  Plot earthquakes with variable-sized circles
%
```

```
color red
file tmpe
symbol circle 20
mode 30 1 2 3
read
color black
plot
stop
```

Using White

By judicious use of line weights and the color `white` it is possible to clarify a figure considerably. To show how this works, we make another set of seismicity maps, these for the San Jacinto fault in the Anza area. There was a magnitude 5.2 earthquake on this fault in 2005 which was the largest in some time; suppose we want to make a map to show it. We suppose that we have three files: `faults`, containing line segments for the surface fault `tmpe`, containing the earthquake locations and magnitudes for the earthquakes from 2003 through 2005, and `tmpm`, containing the location and magnitude for the large earthquake. For our first plot we try putting each of these in a different color, with the label for the largest earthquake in yet another:

```
frame
weight 12
char .12
titl San Jacinto Fault Seismicity: 2003-2005, M\sub{L}\1391\1.2
weight 10
char .10
xlim 6.00 -116.9 -116.3
ylim 4.80 33.3 33.7
mode 20 2 1
file faults          % faults in black
read
color red
file tmpe     % variable-size earthquakes in red
symbol circle 20
mode 30 2 1 3
read
color green          % mainshock in green
file tmpm
```



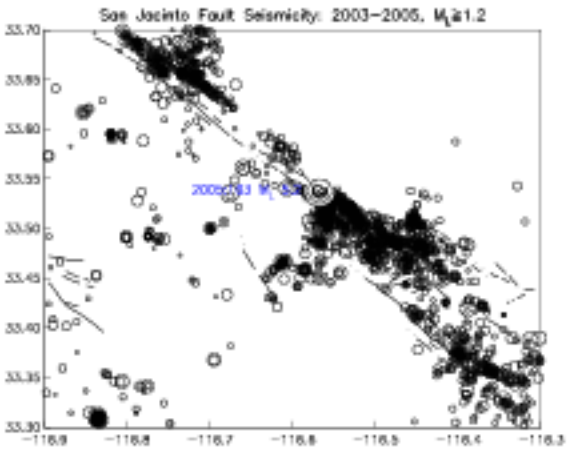
```
read
%                       label in blue
color blue
note (-116.59 33.535 r)2005:163 M\sub{L} 5.2
color black
plot
stop
```

One change you should notice is that we have used, in some cases, only the first four characters of a command, as for example `char` instead of `character`. We can in fact do this for all *plotxy* commands.

This colored plot does not work very well: the many small earthquakes hide the fault, and it is difficult to pick out the label as well. It also has the disadvantage, admittedly now diminishing, that color reproduction is more expensive and difficult than plain black and white (including shades of gray).

We can actually do better, and use fewer colors, by utilizing the edge-detection approach mentioned above when discussing coastlines: shapes can be easily distinguished from those around them if they are surrounded by a small amount of white space. Such a surround is produced by plotting a shape twice: first, with the color set to `white` and a large line weight, to paint "white" over anything else nearby; second, with a more usual color, and a smaller line weight.

Our next plot shows this method being used to distinguish the earthquake of interest from all the others, even when the same color is used for both:




```
frame
weight 12
char .12
titl San Jacinto Fault Seismicity: 2003-2005, M\sub{L}\1391\1.2
weight 10
char .10
xlim 6.00 -116.9 -116.3
ylim 4.80 33.3 33.7
mode 20 2 1
file faults
read
file tmpe
symbol circle 20
mode 30 2 1 3
```

```
read
weight 70
color white
file tmpm
read
weight 10
color black
read
color blue
note (-116.59 33.535 r)2005:163 M\sub{L} 5.2
color black
plot
stop
```

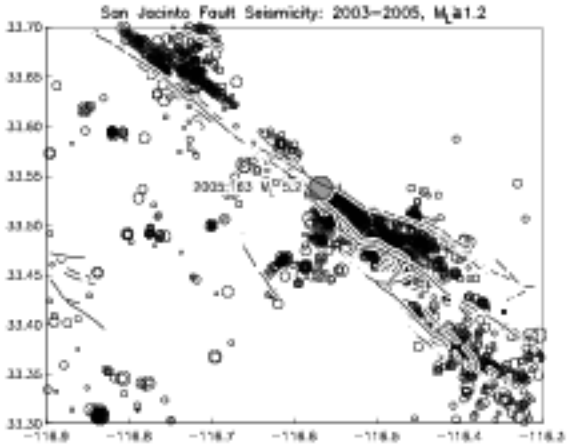
Having done this much, we can apply this same approach to separating the label and the faults from the many earthquakes plotted. We plot, in order:

1. All the earthquakes, in black, with a regular line weight.
2. The faults, in white, and a large line weight.
3. The earthquake of interest, in white, and a large line weight.
4. The earthquake of interest, in gray, and a regular line weight – and also with the symbol filled.
5. The earthquake of interest, in black, and a regular line weight.
6. The faults, in black, and a regular line weight.
7. The label, in white, and a large line weight.
8. The label, in black, and a regular line weight.

The result is a plot that uses no color other than a gray shade, but in which all the elements are distinct.

```
frame
weight 12
char .12
titl San Jacinto Fault Seismicity: 2003-2005, M\sub{L}\1391\1.2
weight 10
char .10
```

```
xlim 6.00 -116.9 -116.3
ylim 4.80 33.3 33.7
file tmpe
symbol circle 20
mode 30 2 1 3
read
symbol -1
weight 70
color white
mode 20 2 1
file faults
read
symbol circle 20
mode 30 2 1 3
file tmpm
read
weight 10
color gray
symbol filled circle 30
read
color black
symbol circle 30
read
symbol -1
mode 20 2 1
file faults
read
weight 100
color white
note (-116.59 33.535 r)2005:163 M\sub{L} 5.2
weight 10
color black
note (-116.59 33.535 r)2005:163 M\sub{L} 5.2
plot
stop
```



Plotting Time Series

We close with an example of how to plot "one-dimensional" data, such as a time series; this allows us to introduce new commands, and variations on old ones. In this particular case we have values sampled every minute of tilt in the vault at

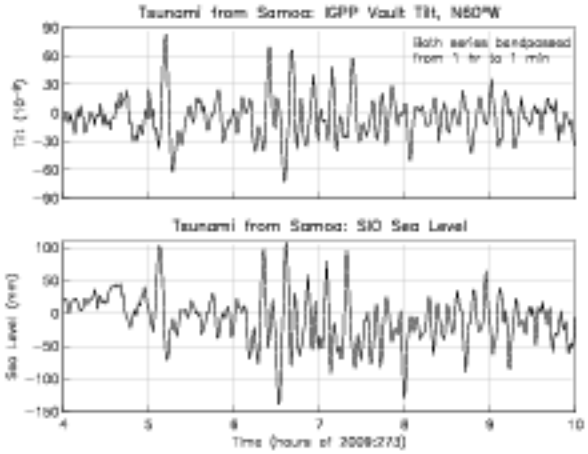
IGPP, and sea level at the end of the SIO pier, at the time of arrival of the tsunami generated by the Samoa earthquake of September 29, 2009.[3] The data form two columns in a file called `tsundata`. These look similar because the loading caused by the extra ocean water causes a tilt in the direction of the ocean.

We could add times to these data and read them as x-y pairs, but this is not necessary. If we read them in using `mode 10`, they are assumed to be y values separated by an x-interval of 1.0, and with the first point having an x value of 1.0. To put the x values in terms of time, we use the `affine` command. This takes the values of x and y the series originally contain, and converts them to plot units using the equations

$$x_{pl} = c_x x + x_0 \qquad \text{and} \qquad y_{pl} = c_y y + y_0$$

where the command is `affine cx x0 cy y0`. The first time point is at 2009:273:04:00 (day 273 is September 30); we can scale the x-units to hours of day 273 by taking $c_x = 1/60 = 0.016666$ and $x_0 = 4 - 1/60 = 3.983333$ (so the first point will be at $x_{pl} = 4.0$). We also use the `affine` command to scale the sea-level data from cm to mm.

```
frame grid solid
char .12
weight 12
titl Tsunami from Samoa: IGPP Vault Tilt, N60\1429\W
char .10
weight 10
ylab Tilt (10\sup{-9})
xlim 6 4 10
ylim 2 -90 90 30
affine 0.016666666 3.983333 1 0
mode 10 1
file tmp3
read
color black
note (8.1 70)Both series bandpassed
```



[3] The original data in both cases was sampled at every second; tides were removed, and both series filtered to remove energy outside of the period range from one hour to one minute. The tilt data are from an Askania tiltmeter; the sea level data are from a pressure gauge operated by Scripps' Coastal Data Information Program.

```
note (v)from 1 hr to 1 min
frame -xnum
plot 1.5 4
note
char .12
weight 12
titl Tsunami from Samoa: SIO Sea Level
char .10
weight 10
frame +xnum
ylim 2 -150 110
ylab Sea Level (mm)
xlab Time (hours of 2009:273)
affine 0.016666666 3.983333 10 0
mode 10 2
file tmp3
read
color black
plot 0 -2.5
stop
```

We have used some other features of *plotxy* for the first time as well; setting `frame -xnum` for the first plot removes the (redundant) numbering information; we restore it with `frame -xnum` after the first `plot` command. The `note (v)` usage shows how to stack notes vertically; the `note` with no arguments after the first `plot` command keeps the note from being repeated on the second plot. We have used a pair of `plot` commands to stack a pair of plots, the first above the second. The values in the first `plot` command put the lower left corner of the plot frame 1.5 inches to the left and 4 inches up relative to the lower left corner of the overall display region; the second `plot` command moves the corner of the second frame 0 inches to the left and 2.5 inches down, relative to the previous corner.

Converting *plotxy* Files to Encapsulated PostScript

As written, *plotxy* output files cannot be imported into LaTeX; to make this possible, they need to be converted to encapsulated PostScript (eps) form. An eps file contains, along with

the PostScript commands, a line at the start that gives the dimensions of the plot on the second line; the plot file would begin as

```
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox:     x_ll  y_ll    x_ur   y_ur
```

where the x and y values give the location of the lower left and upper right corner of the plot, relative to the lower left corner of the page, in units of "Adobe points" (72 to the inch). For example, the last plot we showed had

```
%%BoundingBox:     59.8     76.8    545.8    449.7
```

Since *plotxy* does not know what is coming at the time it writes the start of the output file (default `mypost`) it cannot write the `BoundingBox` in the place needed; but by keeping track of where it is putting ink, it does write this at the end of the output file. The following shell script will create an encapsulated PostScript file by moving the end to the start; the name is given as the first (and only) argument.

```
#!/bin/sh
#  converts plotxy output file   mypost to .eps file
#
case $# in
1)
tail -2 mypost > $1.eps
sed -n '2,$p' mypost >> $1.eps
;;
*)
cat << XXX

Usage: mypost2eps name.for.eps

                where

name.for.eps is the name (no .eps) of the encapsulated'
     PostScript file that will be created

XXX
;;
esac
```