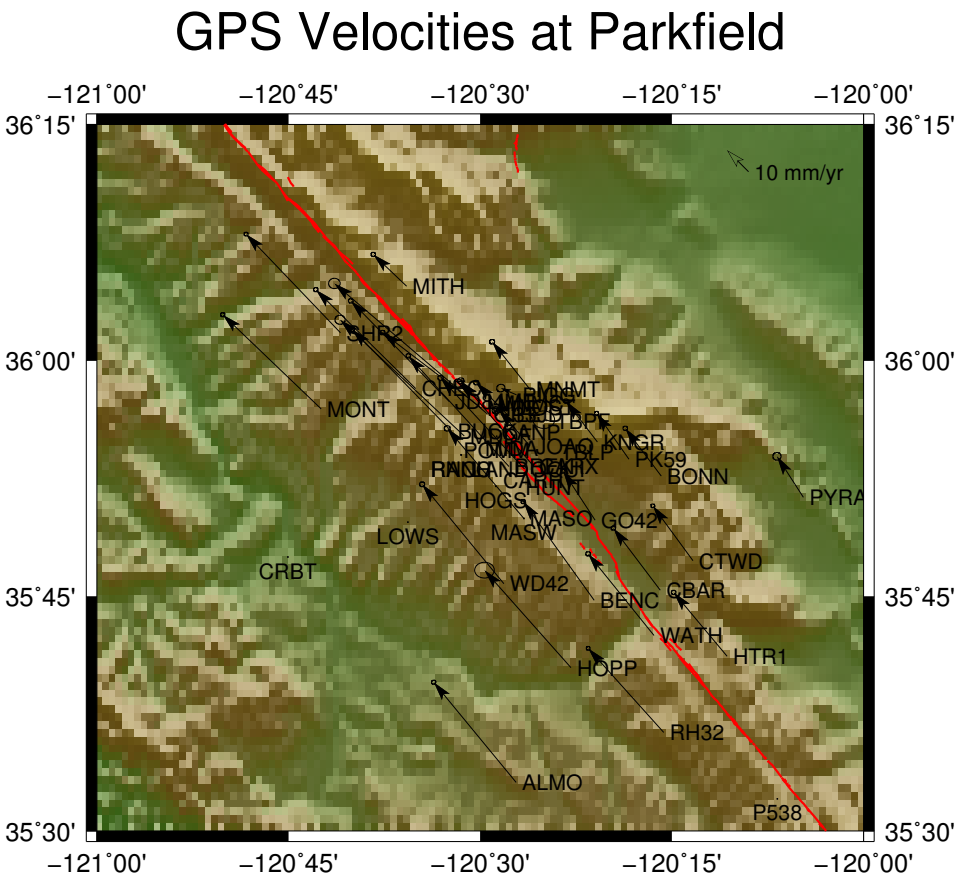
CERI 7104/8104 Homework 4 Solutions

Here are my solutions to HW 4. I have included the shell script to produce the plot "hw4.csh" as well as my AWK script for reformatting the fault trace data "kml2txt.awk." Please see those files for details.

1. Here is my map of the Parkfield section of the San Andreas Fault. I use the same global 30s dataset that we looked at in lab, though its resolution is a bit coarse when viewed at this scale. There are higher resolution datasets available through the USGS web site that I would download and use if this map were to go in a publication, but for the purposes of this homework assignment, this map is fine. Many of you showed a much larger area than this, which made it hard to read the GPS vectors, but since I did not specify a region there were no points lost as long as you displayed everything correctly.



Here is how I parsed the file to make the plot in the solutions. I used two regular expressions to accomplish this – one to idenfity coordinates, and one to identify the beginning of line segments to determine where to insert the > characters. The regular expressions are quite long, so while I attempt to explain them in the comments in the file, I explain them here as well:

- `/<LineString.*-?[0-9]{1,3}\.[0-9]*,-?[0-9]{1,2}\.[0-9]*/` Is used to identify the beginning of each segment. The KML tag for the start of a line segment begins with "<LineString" so I am looking for lines containing this set of characters. Following the tag for a LineString, I include `.*`, which represents an arbitrary number of arbitrary characters that may exist between the LineString tag and the actual coordinates. Then I have a series of characters that are meant to match a general set of geographic coordinates: first is an optional minus sign, then 1-3 numeric characters, a decimal point, then an arbitrary number of numeric characters, all of which represents the longitude coordinate. Then comes a comma, and a similar set of characters to represent the latitude coordinate. Once I find these lines, I print out `>` on its own line, then the longitude and latitude. I need to use the `substr` function to do this, since the field separator does not find the start of the longitude coordinate. I use the coordinate part of the regular expression as the match condition to find the position of the coordinate within the string.
- `/^-?[0-9]{1,3}\.[0-9]*,-?[0-9]{1,2}\.[0-9]*/` is similar to the regular expression above, except the geographic coordinates must come at the beginning of a line. This is for all other coordinates, so we just simply need to print out the latitude and longitude for this case.

All of you were able to do a reasonable job of parsing the KML file, though most of you could have documented your script a little bit more. Note in my comments I explicitly mention in detail what my expressions are and why I need them. Many of you achieved your results with simpler expressions (which is okay) but did not say anything about why you could use them.

Feel free to use my AWK script in the future if you need to parse a KML file for GMT. You can run it from the command line using (for example)

```
$ ./kml2txt.awk Historic.kml > historic.txt
```

Alternatively, you can just pipe the result directly into `psxy`, which is what I did in my shell script solution. Otherwise, this problem was much like the shaded topography and GPS velocity vector examples that we did in class.

2. Your backprojection results were good. All of you wrote your script to accept the appropriate inputs, so you should have no trouble reusing your script on your final. Some of you used AWK to find the map limits from the input text file, which is an alternative to giving the map limits as inputs to the script. I accepted either version. I did not specify whether the map should be rectangular or have lines of constant latitude/longitude as the map borders, so either was acceptable.