1) Compile EEMs and bed2diffs by following the documentation. Dependencies:
    a) Eigen
    b) Boost
    c) Plinkio (necessary for bed2diffs, used to compute the diff matrix)

2) Once bed2diffs is compiled, create a bedfile in the proper format to convert to a diff matrix
    a) Plink doesn't like the chromosome names in the original vcf. You can tell it to preserve them, but this causes problems with bed2diffs down the line. Better to just change chromosome names to numbers now using sed:

```
$ sed -i 's/NC_035780.1/1/g'  reallyLongName.vcf
```

(repeat for each chromosome,NCBI tells which chrom is which)

** I went back and created a bash script to do this --- renameChroms.sh (takes 1 positional param, the name of the vcf file)**

b) Run plink on the edited VCF to convert it to bed/bim/fam files:

```
$ plink --vcf SNP.TRSdp5g95FnDNAmaf05_10KThinnedRandomSNPs13PCs.vcf  --const-fid
```

3) Calculate the difference matrix on the newly created bed/bim/bam files using bed2diffs:

```
$ /path/bed2diffs_v1 --bfile reallyLongName --nthreads 2
```

4) use createCoodsForEEMS.R to create the coord file. It uses the "measurements" library to convert latitude and longitude to decimal degrees. Unknown latitudes and longitudes are manually entered based on google maps data. Besides converting the metadata into a coordinates file, it also removes the non-wild samples from the diff matrix.

Usage:

```
$ Rscript createCoordForEEMS.R <file prefix>
```

The prefix should correspond to the name that was used for the .diffs file (but without .diffs at the end)

5) Make the .outer file using google maps. Go to google maps -> your places -> maps -> create a map. Make a copy of the .coords file and change the extension to .csv, now you can load the coordinates onto the map. Add a new layer, use the "draw a line" tool to draw the habitat boundaries and save the polygon. Then export the map to a .kml file. Use a regex to extract the coordinates, which are between <coordinates>...</coordinates> in the .kml file (or just copy and paste).

I wrote a quick perl script that uses perl regex to extract and format the coordinates: extractCoord.pl

6) Run run_eems_snps. This is the parameter file I tried first:

```
datapath = (long data name)
mcmcpath = (long data name-chain1)
nIndiv = 53
nSites = 5996
nDemes = 200
diploid = true
numMCMCIter = 2000000
numBurnIter = 1000000
numThinIter = 9999
```

** **note on ndemes** -- the runtime increases rapidly with more demes: it went from ~15 minutes with 200 demes to almost 5 hours with 500 demes. You can start run_eems_snps then immediately cancel it to see how many demes the samples occupy, probably a good idea to do this a few times, adjusting ndemes each time, until you find the smallest number of demes that separates the data in the way you want**

*note - I had to go back and change the order of the columns in the .coord file to match the .outer file -- the exported polygon gave longitude first and latitude second. Per the EEMs documentation, order does not matter as long as it matches between these two files and the .coord file was easier to change

Output:

```
 Ending iteration 2000000 with acceptance proportions:
            (90981/125722) = 72.37% for proposal type "qTileRate",          with proposal variance "qEffctProposalS2"
            (36860/125194) = 29.44% for proposal type "qTileMove",          with proposal variance "qSeedsProposalS2"
            (56703/125492) = 45.18% for proposal type "qBirthDeath"
            (280996/374207) = 75.09% for proposal type "mTileRate",         with proposal variance "mEffctProposalS2"
            (109079/249596) = 43.70% for proposal type "mMeanRate",         with proposal variance "mrateMuProposalS2"
            (206145/375304) = 54.93% for proposal type "mTileMove",         with proposal variance "mSeedsProposalS2"
            (122179/374812) = 32.60% for proposal type "mBirthDeath"
            (156779/249673) = 62.79% for proposal type "degrees of freedom"
 and effective degrees of freedom = 463.19
       number of qVoronoi tiles = 8
       number of mVoronoi tiles = 25
        Log prior = 44.30
        Log llike = 2840.96
Final log prior: 44.30
Final log llike: 2840.96
```

Looks like qTileRate and mTileRate are getting accepted too often, according to the documentation I should increase qEffectProposalS2 and mEffectProposalS2

New params for chain 2 (all other params the same except mcmcpath):

```
mEffctProposalS2 = 0.200000
 qEffctProposalS2 = 0.010000

 Ending iteration 2000000 with acceptance proportions:
		(57469/125296) = 45.87% for proposal type "qTileRate",		with proposal variance "qEffctProposalS2"
		(36239/124457) = 29.12% for proposal type "qTileMove",		with proposal variance "qSeedsProposalS2"
		(84237/125215) = 67.27% for proposal type "qBirthDeath"
		(257344/374821) = 68.66% for proposal type "mTileRate",		with proposal variance "mEffctProposalS2"
		(108939/250492) = 43.49% for proposal type "mMeanRate",		with proposal variance "mrateMuProposalS2"
		(207539/374977) = 55.35% for proposal type "mTileMove",		with proposal variance "mSeedsProposalS2"
		(146823/374605) = 39.19% for proposal type "mBirthDeath"
		(157846/250137) = 63.10% for proposal type "degrees of freedom"
 and effective degrees of freedom = 445.92
	number of qVoronoi tiles = 18
	number of mVoronoi tiles = 19
	 Log prior = 64.98
	 Log llike = 2846.34
Final log prior: 64.98
Final log llike: 2846.34
```

Still too high….

New params for chain 3:

```
mEffctProposalS2 = 0.500000
qEffctProposalS2 = 0.030000

 Ending iteration 2000000 with acceptance proportions:
		(37990/125108) = 30.37% for proposal type "qTileRate",		with proposal variance "qEffctProposalS2"
		(37020/125498) = 29.50% for proposal type "qTileMove",		with proposal variance "qSeedsProposalS2"
		(86076/125157) = 68.77% for proposal type "qBirthDeath"
		(215632/374724) = 57.54% for proposal type "mTileRate",		with proposal variance "mEffctProposalS2"
		(109092/249256) = 43.77% for proposal type "mMeanRate",		with proposal variance "mrateMuProposalS2"
		(207123/374672) = 55.28% for proposal type "mTileMove",		with proposal variance "mSeedsProposalS2"
		(175846/375779) = 46.80% for proposal type "mBirthDeath"
		(157610/249806) = 63.09% for proposal type "degrees of freedom"
 and effective degrees of freedom = 465.75
	number of qVoronoi tiles = 16
	number of mVoronoi tiles = 21
	 Log prior = 61.70
	 Log llike = 2843.54
Final log prior: 61.70
Final log llike: 2843.54
```

qTileRate is good, mTileRate still a bit high

Finally decided on:

mEffctProposalS2 = 0.9
qEffctProposalS2 = 0.03

```
Iteration 2000000 of 2000000...
 Ending iteration 2000000 with acceptance proportions:
                (39334/125595) = 31.32% for proposal type "qTileRate",              with proposal variance "qEffctProposalS2"
                (56132/125268) = 44.81% for proposal type "qTileMove",              with proposal variance "qSeedsProposalS2"
                (90381/125282) = 72.14% for proposal type "qBirthDeath"
                (140192/374168) = 37.47% for proposal type "mTileRate",             with proposal variance "mEffctProposalS2"
                (102869/249758) = 41.19% for proposal type "mMeanRate",             with proposal variance "mrateMuProposalS2"
                (177479/374777) = 47.36% for proposal type "mTileMove",             with proposal variance "mSeedsProposalS2"
                (116372/375474) = 30.99% for proposal type "mBirthDeath"
                (155395/249678) = 62.24% for proposal type "degrees of freedom"
 and effective degrees of freedom = 659.80
         number of qVoronoi tiles = 18
         number of mVoronoi tiles = 14
          Log prior = 67.10
          Log llike = 3052.31
Final log prior: 67.10
Final log llike: 3052.31
```

7) Plot results. Had to install rgdal to plot the diagrams on a world map. Install.packages initially failed on Ubuntu for this package, I had to install libgdal-dev and libproj-dev first on the command line:

```
$ sudo apt-get install libgdal-dev libproj-dev
```

After those completed successfully I was able to install the package in R.

There’s lots of different options to use in plotting and the ‘code’ for it is basically just eems.plots(<options>). The script ‘plot_eems_results.R’ includes the options that I used but it is not meant to be run from the command line or without making adjustments for the current dataset and desired output.