Last updated: 01/25/2022 ECS

STRUCTURE Pipeline Tutorial

The following is a quick start guide to the pipeline required to generate STRUCTURE results from population genetics data. ***It is recommended that you also read the user manuals for each of the included programs, in addition to the notes included in this guide.***

Creating STRUCTURE files

1. Following filtering in GBS-SNP-CROP (step 7), export a TASSEL compatible data file. Import into TASSEL (https://www.maizegenetics.net/tassel), complete any additional post-processing filtering steps (i.e., minor allele frequency, maximum heterozygosity, etc.), and export dataset as VCF.

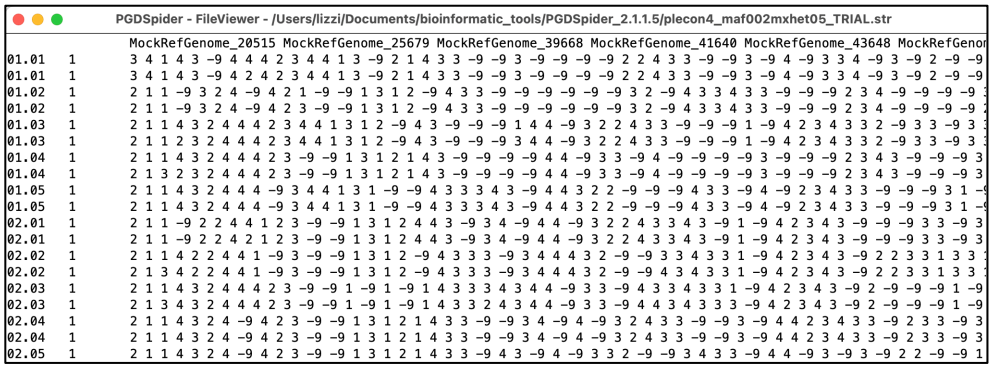
2. Convert the VCF file into a STR file using a program such as PGDSpider (http://www.cmpg.unibe.ch/software/PGDSpider/). Note, if launching PGDSpider from a Mac, `cd` into the directory that contains the JAR file and run the command: `java -Xmx1024m -Xms512m -jar PGDSpider2.jar`, otherwise PGDSpider won't recognize your file tree (this is also noted in the ReadMe file). If launching on a Windows, run the EXE file. For PGDSpider, the input format should be set to VCF, output format set to STRUCTURE. Your input file should look something like this:

PGDSpider - FileViewer - /Users/lizzi/Documents/bioinformatic_tools/PGDSpider_2.1.1.5/pleconv4_maf002mxhet05.vcf

```
##fileformat=VCFv4.0
##Tassel=<ID=GenotypeTable,Version=5,Description="Reference allele is not known. The major allele was used as reference allele"
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the reference and alternate alleles in the order listed">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth (only filtered reads used for calling)">
##FORMAT=<ID=GQ,Number=1,Type=Float,Description="Genotype Quality">
##FORMAT=<ID=PL,Number=.,Type=Float,Description="Normalized, Phred-scaled likelihoods for AA,AB,BB genotypes where A=ref and B=
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  01.01   01.02   01.03   01.04   01.05   02.01   02.02
MOCKREFGENOME   20515   MockRefGenome_20515     T       G       .       PASS    .       GT      1/1     0/0     0/0     0/0
MOCKREFGENOME   25679   MockRefGenome_25679     A       C       .       PASS    .       GT      1/1     0/0     0/0     0/0
MOCKREFGENOME   39668   MockRefGenome_39668     A       G       .       PASS    .       GT      0/0     0/0     0/0     0/1
MOCKREFGENOME   41640   MockRefGenome_41640     C       T       .       PASS    .       GT      0/0     ./.     0/1     0/1
MOCKREFGENOME   43648   MockRefGenome_43648     G       T       .       PASS    .       GT      0/0     0/0     0/0     0/0
MOCKREFGENOME   46170   MockRefGenome_46170     T       G       .       PASS    .       GT      ./.     0/0     0/0     0/0
MOCKREFGENOME   48038   MockRefGenome_48038     C       A       .       PASS    .       GT      0/0     0/0     0/0     0/0
MOCKREFGENOME   55737   MockRefGenome_55737     C       T       .       PASS    .       GT      0/1     ./.     0/0     0/0
MOCKREFGENOME   77748   MockRefGenome_77748     C       A       .       PASS    .       GT      0/0     0/0     0/0     0/0
```

In the parser questions the following should be set: 1) What is the ploidy of the data? Diploid; 2) Take most likely genotype if "PL" or "GL" is given in the genotype field? No; Do you want to exclude loci with only missing data? Yes; Do you want to include non-polymorphic SNPs? Yes; Do you want to include a file with populations definitions? No (this is something that's easier to assign manually, especially if the population ID is included in the individual ID).

In the writer questions the following should be set: Save more specific fastSTRUCTURE format? No; Do you want to include inter-marker distances? No; Specify what data type should be included in the STRUCTURE file: SNP. Your output file should look something like this:



3. Using a text editor tool (i.e., Notepad++ or VS Code) update the population IDs in the STR file. In a STR file, the first column is typically the individual ID (can contain symbols, letters, and numbers), the second column is the population ID (should be integer), and each following column is the SNP identification for each locus. Be sure not to edit the formatting of the file.

4. Once this STR file is created, genetic distance metrics can be calculated between pairs of populations. This may or may not be necessary depending on your downstream application. To do so, run the following script in R:

```
library(pegas)
library(adegenet)
library(hierfstat)
setwd("[your/file/path]")
Loci <- read.structure("[structure_file.str]",
        n.ind = [number of ind (# rows minus one divided by 2)],
        n.loc = [number of loci (number of locus columns)],
        onerowperind = FALSE, col.lab = 1, col.pop = 2,
        row.marknames = 1, NA.char = -9)
#generate various file formats
genpop <- genind2genpop(Loci)
hierfstat <- genind2hierfstat(Loci)
#calculate population-level statistics and print to files
basicstat <- basic.stats(hierfstat, diploid = TRUE, digits = 5)
write.table(basicstat$overall, file = "[output.csv]", sep = ",")
#calculate pairwise fst and print to file
Fstpairwise <- pairwise.fst(Loci, res.type = "matrix")
Fstpairwise <- pairwise.neifst(Loci, diploid = TRUE)
write.table(Fstpairwise, file= "[output.csv]", sep = ",")
#calculate Reynolds genetic distance and print to file
#(for Edwards, set method = 1)
write.table(data.matrix(dist.genpop(genpop, method = 2, diag = TRUE,
upper = FALSE)), file= "[output.csv]", sep = ",")
```

Running STRUCTURE

1. The next step is to run the STRUCTURE program itself. While it is possible to run this locally, it's much easier to run STRUCTURE on the computing cluster. You will need four files for this process if running on Coeus (https://sites.google.com/pdx.edu/research-computing/getting-started/coeus-hpc-cluster):

    a. R file containing the script to run STRUCTURE
    b. SH file to submit to the SLURM manager
    c. STR file containing SNP information
    d. TXT file to assign parallelization

    To install STRUCTURE on Coeus, follow the Unix/Linux instructions on the STRUCTURE website (https://web.stanford.edu/group/pritchardlab/structure_software/release_versions/v2.3.4/html/install.html).

2. **R file**: will look something like the following. Make sure to install ParallelStructure R package prior to submitting the job on Coeus.

```
library(ParallelStructure)
setwd("[your/file/path]")
parallel_structure(infile="structure_file.str", outpath="results/",
joblist="ParallelJobs.txt", n_cpu=20,
structure_path="/home/ech4/console/", numinds = [num ind], numloci =
[num loci], ploidy = 2, label = 1, popdata = 1, markernames = 1,
missing = -9, locprior = 1, printqhat = 1)
```

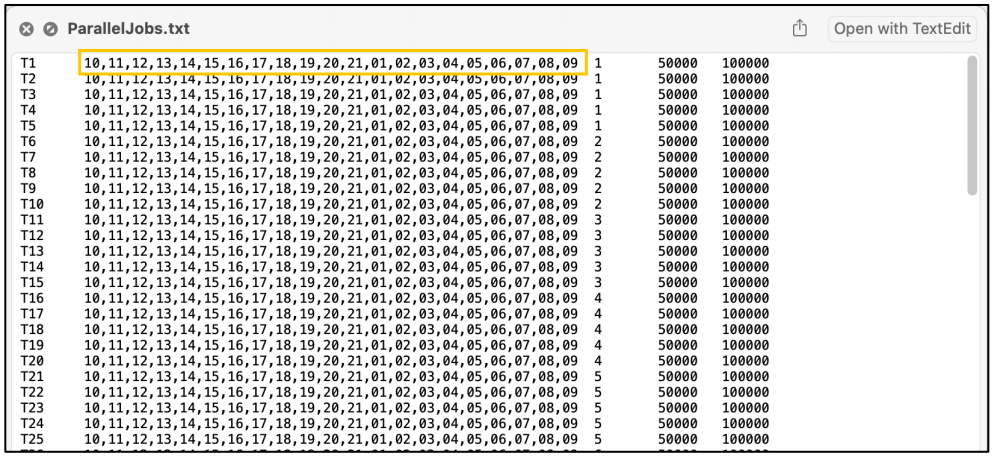
3. **SH file**: follow this general structure to write the SH file:

```
#!/bin/bash
#SBATCH --job-name str
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 1
#SBATCH --cpus-per-task 20
#SBATCH --partition long
module load R/3.5.0/openmpi-2.0/gcc7.2.0
srun --unbuffered R CMD BATCH ./parallelstr.R
```

4. **STR file**: this is the file that was produced by PGDSpider.

5. **TXT file**: this file is a bit of a hassle, and I'd recommend making a copy of an existing ParallelJobs.txt file on the I: drive for your own use (there's currently one in Lizzi's

folder: GBS/achmol_gbs/structure/ParallelJobs.txt). You will need to edit it to match your number of populations where the string of population values and the order of populations (see below image, boxed in orange) matches your STR file. Everything else should be able to stay the same.

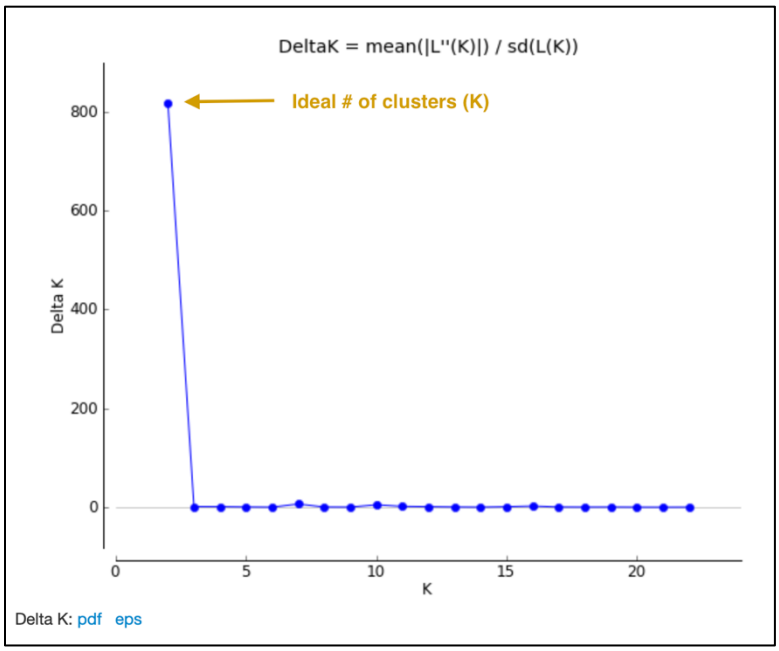


6. **Output**: STRUCTURE will output a *bunch* of files in your specified output path. This will be parsed into a more reader-friendly format in the following steps. Use the compress or zip function in Windows or Mac (or the `zip` command in Linux), zip your results file for the Structure Harvester.

Parsing STRUCTURE output

1. **Structure Harvester**: The program Structure Harvester (http://taylor0.biology.ucla.edu/structureHarvester/#) helps you interpret the STRUCTURE output and identify the ideal number of clusters. Navigate to the Structure Harvester webpage and upload your zipped results file. It'll take a few minutes to run, and the page will automatically refresh with your results. At the top of the page, there will be a download button. Download it and untar and unzip the file (`tar –xvzf archive.tar.gz`) for the next few steps.

   One of the main take-away points of Structure Harvester is the plot of Delta K and the Evanno table. This is part of the downloaded data or can also be seen in the web browser of Structure Harvester once the data are processed. Scroll down to the Delta K plot, which should look something like this:



The optimal number of clusters for the data set is at the peak of this Delta K plot (see the arrow in the figure above). This data set has an optimum of 2 clusters, which is the minimum number of clusters possible.

Another visual of the cluster patterns for these data is in the Evanno table farther down in the Structure Harvester output. It should look something like below. Note that I tested from 1 to 23 clusters, and 2 clusters was still the best option.

K	Reps	Mean LnP(K)	Stdev LnP(K)	Ln'(K)	|Ln''(K)|	Delta K
1	10	-362941.550000	8.025549	—	—	—
2	10	-358937.610000	50.335617	4003.940000	41105.140000	816.621361
3	10	-396038.810000	37254.940732	-37101.200000	45163.730000	1.212288
4	10	-478303.740000	209075.792044	-82264.930000	224151.440000	1.072106
5	10	-784720.110000	717709.037017	-306416.370000	342747.410000	0.477558
6	10	-748389.070000	375230.945260	36331.040000	36610.810000	0.097569
7	10	-748668.840000	217771.662699	-279.770000	1418615.650000	6.514234
8	10	-2167564.260000	3919357.683274	-1418895.420000	1478828.310000	0.377314
9	10	-2107631.370000	2950673.443588	59932.890000	577189.540000	0.195613
10	10	-1470508.940000	1326873.347290	637122.430000	6497268.910000	4.896676
11	10	-7330655.420000	5632955.834733	-5860146.480000	9548298.680000	1.695078
12	10	-3642503.220000	3298141.295454	3688152.200000	3405211.330000	1.032464
13	10	-3359562.350000	2187074.311689	282940.870000	1025986.920000	0.469114
14	10	-4102608.400000	3098154.372609	-743046.050000	291152.190000	0.093976
15	10	-5136806.640000	3448399.539969	-1034198.240000	3294033.610000	0.955235
16	10	-2876971.270000	1482810.798376	2259835.370000	3430155.780000	2.313279
17	10	-4047291.680000	3239785.152878	-1170320.410000	1273981.556667	0.393230
18	9	-3943630.533333	3296710.516874	103661.146667	764142.046667	0.231789
19	5	-3075827.340000	1305991.854878	867803.193333	437433.653333	0.334944
20	5	-2645457.800000	2530837.353370	430369.540000	189647.560000	0.074935
21	5	-2025440.700000	2362948.393565	620017.100000	20441.000000	0.008651
22	5	-1425864.600000	889690.074286	599576.100000	97969.260000	0.110116
23	5	-924257.760000	1068364.152755	501606.840000	—	—

2. **CLUMPP**: CLUMPP is a program that checks for biases in the STRUCTURE assignments (https://rosenberglab.stanford.edu/clumpp.html). Move the appropriate files corresponding to your optimal number of clusters out of the Structure Harvester downloaded dataset (e.g., for the above dataset, I would move files K2.indfile and K2.popfile) into the folder containing the CLUMPP executable (CLUMPP.exe; Windows executable works, I can't get the Mac executable to work though). CLUMPP is run by editing the paramfile file and then executing CLUMPP.exe. *You will have to run this program twice: once for individual data and once for population data*.

   Population data: In a text editor (e.g., Notepad++, TextEdit, or VSCode), modify the following parameters (important differences between the population data run and individual run are highlighted):
   - DATATYPE: 1
   - INDFILE: K2.indfile (or whatever your optimal Structure Harvester output is)
   - POPFILE: K2.popfile
   - OUTFILE: [pop file prefix].outfile
   - MISCFILE: [pop file prefix].miscfile
   - K: # of clusters (identified in Structure Harvester)
   - C: # of populations
   - R: number of runs (probably 10)
   - M: 1 (FullSearch method)
   - W: 1 (TRUE; weight by the number of ind in each population)
   - S: 2 (setting to 2 basically just makes sure values are between 0 and 1)
   - […] everything else can stay the same until you get to:
   - PERMUTED_DATAFILE: [pop file prefix].perm_datafile

   Individual data: In a text editor, modify the following parameters in the paramfile (important differences between the population data run and individual run are highlighted)
   - DATATYPE: 0
   - INDFILE: K2.indfile
   - POPFILE: K2.popfile
   - OUTFILE: [ind file prefix].outfile
   - MISCFILE: [ind file prefix].miscfile
   - K: # of clusters (identified in Structure Harvester)
   - C: # of individuals
   - R: number of runs (probably 10)
   - M: 1 (FullSearch method)
   - W: 1 (TRUE; weight by the number of ind in each population)
   - S: 2 (setting to 2 basically just makes sure values are between 0 and 1)
   - […] everything else can stay the same until you get to:
   - PERMUTED_DATAFILE: [ind file prefix].perm_datafile

STRUCTURE visualization

1. **distruct**: distruct is a program that makes pretty, but not easy, figures for STRUCTURE (https://rosenberglab.stanford.edu/distructDownload.html). distruct operates a very similar way to CLUMPP where you edit a parameter file and run an executable. You will need to have a program that allows you to view a PS image for this program. Mac's Preview application can open a PS image by converting it to a PDF; Windows will require an additional program to view these images (I know GIMP works, which is an open-source photo editing program (https://www.gimp.org), but it's a bit clunky so there might be better options out there).

   Copy the OUTFILEs for both the individual and population runs of CLUMPP into the distruct folder. Rename the extension on the individual file to .indivq (i.e., indK2.indivq) and the extension on the population file to .popq (i.e., popK2.popq).

   The INDIVQ file should look something like this, with one line per individual and the number of columns of the right side of the ":" symbol corresponds to the number of clusters:

```
  1        1  (10)     10 :  0.1842 0.8158
  2        2  (10)     10 :  0.1196 0.8804
  3        3  (16)     10 :  0.1288 0.8712
  4        4  (11)     10 :  0.1604 0.8396
  5        5  (12)     10 :  0.1227 0.8773
  6        6  (21)     10 :  0.0798 0.9202
  7        7  (10)     10 :  0.0999 0.9001
  8        8   (9)     10 :  0.1309 0.8691
  9        9  (14)     10 :  0.1330 0.8670
 10       10  (10)     10 :  0.0923 0.9077
 11       11  (10)     11 :  0.1055 0.8945
 12       12  (19)     11 :  0.1109 0.8891
 13       13   (7)     11 :  0.1550 0.8450
 14       14  (15)     11 :  0.0813 0.9187
 15       15   (7)     11 :  0.1114 0.8886
 16       16  (11)     11 :  0.1435 0.8565
 17       17  (32)     11 :  0.0417 0.9583
 18       18  (45)     11 :  0.0549 0.9451
 19       19  (21)     12 :  0.1035 0.8965
 20       20  (24)     12 :  0.1379 0.8621
 21       21  (34)     12 :  0.0477 0.9523
 22       22  (25)     12 :  0.1090 0.8910
 23       23  (23)     12 :  0.1098 0.8902
 24       24  (36)     12 :  0.1348 0.8652
 25       25  (27)     12 :  0.0577 0.9423
 26       26  (51)     12 :  0.0921 0.9079
 27       27  (21)     12 :  0.1218 0.8782
 28       28  (20)     12 :  0.1626 0.8374
 29       29  (22)     13 :  0.1226 0.8774
 30       30  (25)     13 :  0.1242 0.8758
```

The POPQ file should look like this, with one line per population:

```
 1:   0.1028  0.8972  10
 2:   0.1280  0.8720  3
 3:   0.1279  0.8721  9
 4:   0.1238  0.8762  10
 5:   0.1303  0.8697  9
 6:   0.1506  0.8494  8
 7:   0.1399  0.8601  10
 8:   0.1390  0.8610  10
 9:   0.1446  0.8554  9
10:   0.1249  0.8751  10
11:   0.1007  0.8993  8
12:   0.1078  0.8922  10
13:   0.0997  0.9003  4
14:   0.0896  0.9104  10
15:   0.0956  0.9044  10
16:   0.1349  0.8651  10
17:   0.1355  0.8645  8
18:   0.1366  0.8634  10
19:   0.1318  0.8682  10
20:   0.1525  0.8475  8
21:   0.1472  0.8528  9
```

You will also need to make a NAMES file (popK2.names) with the names of your populations, if you'd like to include them in your figure:

```
1 ALST
2 AH
3 BF
4 CW
5 CL
6 CMR
7 DEN
8 DCR
9 DUG
10 LTR1
11 LTR2
12 LTR3
13 MCA
14 RAD
15 TDEN
16 UTR2
17 UTRL
18 UTRR2
19 UTRR3
20 UTRW2
21 WHET
```

And finally, you'll need a PERM (popK2.perm) file with the specified color of each cluster. See the distruct documentation for color options:

```
1 yellow
2 sea_green
```

Edit the drawparams file in a text editor. Include your POPQ, INDIVQ, NAMES, and PERM files (note, I cannot get this program to add labels below the figure for whatever reason, but it will happily add them to the top of the figure. You may have to play around a little bit with the settings!). Make sure to also update K, NUMPOPS, and NUMINDS.

The rest of the drawparams file defines the figure itself. You may have to run this program multiple times and iterate through the settings until you find parameters that work for your dataset. I found the following to work well for my data:

- PRINT_INDIVS: 1
- PRINT_LABEL_ATOP: 1
- PRINT_LABEL_BELOW: 0
- PRINT_SEP: 1
- FONTHEIGHT: 8
- DIST_ABOVE: 5
- DIST_BELOW: -7
- BOXHEIGHT: 50
- INDIVWIDTH: 1.5
- ORIENTATION: 1
- XORIGIN: 300
- YORIGIN: 50
- XSCALE: 2.5
- YSCALE: 2.5
- […] defaults on the rest of the settings

2. I then cropped and rotated my figure with the following result! If any additional formatting is necessary that is not possible within distruct, I recommend a photo-editing software such as GIMP (link above, free and compatible with both Mac and Windows).