polyRAD tutorial

Lindsay V. Clark
University of Illinois, Urbana-Champaign

April 19, 2018

1 Introduction

polyRAD is an R package that assists with genotype calling from DNA sequence datasets such as genotyping-by-sequencing (GBS) or restriction site-associated DNA sequencing (RAD) in polyploids and diploids. Genotype likelihoods are estimated from allelic read depth, genotype prior probabilities are estimated from population parameters, and then genotype posterior probabilities are estimated from likelihoods and prior probabilities. Posterior probabilities can be used directly in downstream analysis, converted to weighted mean genotypes for analyses of additive genetic effects, or used for export of the most probable genotypes for analyses that require discrete genotypic data.

Analyses in polyRAD center around objects of an S3 class called "RADdata". A single "RADdata" object contains the entire dataset of read depth and locus information, as well as parameters that are estimated during the course of analysis.

2 Summary of available functions

For any function named in this section, see its help page for more information. (For example by typing `?VCF2RADdata` into the R console.)

Several functions are available for import of read depth data and (optionally) alignment information into a RADdata object:

- `VCF2RADdata`
- `readTagDigger`
- `readStacks1`
- `readHMC`

More generally, the `RADdata` function is used for constructing RADdata objects; see the help page for that function for more information on what data are needed.

Several pipelines are available for genotype estimation, depending on how the population is structured (i.e. what the genotype prior probabilities should be.):

- `PipelineMapping2Parents`
- `IterateHWE`
- `IterateHWE_LD`
- `IteratePopStruct`
- `IteratePopStructLD`

Lastly, for exporting the estimated genotypes to other software:

- `ExportGAPIT`
- `Export_rrBLUP_Amat`
- `Export_rrBLUP_GWAS`
- `Export_TASSEL_Numeric`

If you need continuous numerical genotypes exported in some other format, see `GetWeightedMeanGenotypes`. Also, `GetLikelyGen` returns the most likely genotypes for a single sample.

3 Estimating genotype probabilities in a mapping population

In this example, we'll import some data from an F1 mapping population of *Miscanthus sinensis* that were output by the UNEAK pipeline. These data are from a study by Liu *et al.* (2015; doi:10.1111/gcbb.12275; data available at `http://hdl.handle.net/2142/79522`), and can be found in the "extdata" folder of the polyRAD installation. *Miscanthus* is an ancient tetraploid that has undergone diploidization. Given the ability of the UNEAK pipeline to filter paralogs, we expect most loci to behave in a diploid fashion, but some may behave in an allotetraploid fashion.

We'll start by loading polyRAD and importing the data into a "RADdata" object. The `possiblePloidies` argument indicates the expected inheritance modes: diploid (2) and allotetraploid (2 2).

```
> library(polyRAD)
> maphmcfile <- system.file("extdata", "ClareMap_HapMap.hmc.txt",
+                           package = "polyRAD")
> mydata <- readHMC(maphmcfile,
+                   possiblePloidies = list(2, c(2, 2)))
> mydata
```

```
## RADdata object ##
299 taxa and 50 loci
766014 total reads
Assumed sample cross-contamination rate of 0.001

Possible ploidies:
Autodiploid (2)
Allotetraploid (2 2)
```

We can view the imported taxa names (subsetted here for space).

```
> GetTaxa(mydata)[c(1:10,293:299)]
```

```
 [1] "IGR-2011-001"    "Kaskade-Justin"  "Map1-001"        "Map1-002"
 [5] "Map1-003"        "Map1-005"        "Map1-008"        "Map1-011"
 [9] "Map1-016"        "Map1-018"        "Map1-488"        "Map1-489"
[13] "Map1-490"        "Map1-491"        "Zebrinus-Justin" "p196-150A-c"
[17] "p877-348-b"
```

All names starting with "Map" are progeny. "Kaskade-Justin" and "Zebrinus-Justin" are the parents. "IGR-2011-001", "p196-150A-c", and "p877-348-b" aren't part of the population, but were doubled haploid lines that were used to screen for paralogous markers. We can tell polyRAD which taxa are the parents; since this is an F1 population it doesn't matter which is "donor" and which is "recurrent".

```
> mydata <- SetDonorParent(mydata, "Kaskade-Justin")
> mydata <- SetRecurrentParent(mydata, "Zebrinus-Justin")
```

The next thing we'll want to do is add our genomic alignment data. For this dataset, we have alignment data stored in a CSV file, also in the "extdata" directory of the polyRAD installation. We'll add it to the `locTable` slot of our `RADdata` object. Be sure to name the new columns "Chr" and "Pos".

```
> alignfile <- system.file("extdata", "ClareMap_alignments.csv",
+                          package = "polyRAD")
> aligndata <- read.csv(alignfile, row.names = 1)
> head(aligndata)
```

```
        Sorghum.LG Position.on.Sorghum.LG..bp.
TP5489           1                     4560204
TP13305          1                     4584260
TP18261          1                     2911329
TP18674          1                      387849
TP19030          1                     7576879
TP26698          1                     6972841
```

```
> mydata$locTable$Chr <- aligndata[GetLoci(mydata), 1]
> mydata$locTable$Pos <- aligndata[GetLoci(mydata), 2]
> head(mydata$locTable)
```

```
        Chr     Pos
TP5489    1 4560204
TP13305   1 4584260
TP18261   1 2911329
TP18674   1  387849
TP19030   1 7576879
TP26698   1 6972841
```

If you don't have alignment data in your own dataset, you can still use the pipeline described here. Just set `useLinkage = FALSE` in the code below. The advantage of including alignment data is that gentoypes at linked markers are used for imputing missing or correcting erroneous genotypes.

Now we can run the pipeline. We'll specify that we don't want to include the parents or doubled haploid lines in the estimation of allele frequencies or evaluation of genotype frequenecies (`excludeTaxa` argument). The `allowedDeviation` argument indicates how different the apparent allele frequency (based on read depth ratios) can be from an expected allele frequency (determined based on ploidy and mapping population type) and still be classified as that allele frequency. The default settings assume an F1 population, but the population type can be adjusted using the `n.gen.backcrossing`, `n.gen.intermating`, and `n.gen.selfing` arguments.

```
> mydata <- PipelineMapping2Parents(mydata,
+                                   freqExcludeTaxa = c("Kaskade-Justin",
+                                                       "Zebrinus-Justin",
+                                                      "IGR-2011-001",
+                                                      "p196-150A-c",
+                                                      "p877-348-b"),
+                                   freqAllowedDeviation = 0.06,
+                                   useLinkage = TRUE)
```

We can examine the allele frequencies. Allele frequencies that fall outside of the expected ranges will be recorded as they were estimated from read depth.

```
> table(mydata$alleleFreq)
```

```
            0.125 0.185006617653235              0.25             0.375
                1                   1                41                 1
              0.5               0.625              0.75 0.814993382346765
               12                   1                41                 1
            0.875
                1
```

Genotype likelihood is also stored in the object for each possible genotype at each locus, taxon, and ploidy. This is the probability of seeing the observed distribution of reads.

```
> mydata$alleleDepth[3,11:20]

TP26698_0 TP26698_1 TP28405_0 TP28405_1 TP28602_0 TP28602_1 TP28986_0 TP28986_1
        1         7         0        51         0        32        15        21
TP31810_0 TP31810_1
       10        13

> mydata$genotypeLikelihood[[1]][,3,11:20]

     TP26698_0    TP26698_1     TP28405_0     TP28405_1     TP28602_0
0 5.968571e-03 4.881592e-25  9.748162e-01 4.440892e-169  9.920309e-01
1 3.115636e-02 3.115636e-02  4.440892e-16  4.440892e-16  2.365849e-10
2 4.881592e-25 5.968571e-03 4.440892e-169  9.748162e-01 1.004524e-100
      TP28602_1    TP28986_0    TP28986_1    TP31810_0    TP31810_1
0 1.004524e-100 5.158357e-45 1.309455e-56 1.087526e-30 2.697672e-35
1  2.365849e-10 8.126672e-02 8.126672e-02 1.365876e-01 1.365876e-01
2  9.920309e-01 1.309455e-56 5.158357e-45 2.697672e-35 1.087526e-30

> mydata$genotypeLikelihood[[2]][,3,11:20]

     TP26698_0    TP26698_1     TP28405_0     TP28405_1     TP28602_0
0 5.968571e-03 4.881592e-25  9.748162e-01 4.440892e-169  9.920309e-01
1 2.662559e-01 3.662109e-04  4.175805e-07  2.075288e-31  1.004524e-04
2 3.115636e-02 3.115636e-02  4.440892e-16  4.440892e-16  2.365849e-10
3 3.662109e-04 2.662559e-01  2.075288e-31  4.175805e-07  5.778929e-20
4 4.881592e-25 5.968571e-03 4.440892e-169  9.748162e-01 1.004524e-100
      TP28602_1    TP28986_0    TP28986_1    TP31810_0    TP31810_1
0 1.004524e-100 5.158357e-45 1.309455e-56 1.087526e-30 2.697672e-35
1  5.778929e-20 1.233327e-02 1.746741e-05 2.592075e-02 9.787414e-04
2  2.365849e-10 8.126672e-02 8.126672e-02 1.365876e-01 1.365876e-01
3  1.004524e-04 1.746741e-05 1.233327e-02 9.787414e-04 2.592075e-02
4  9.920309e-01 1.309455e-56 5.158357e-45 2.697672e-35 1.087526e-30
```

Above, for one individual (Map1-001), we see its read depth at the first ten alleles (first five loci), followed by the genotype likelihoods under diploid and tetraploid models. For example, at locus TP26698, heterozygosity is the most likely state, although there is a chance that this individual is homozygous for allele 1 and the one read of allele 0 was due to contamination. If this locus is allotetraploid, it is most likely that there is one copy of allele 0 and three copies of allele 1. Other loci have higher depth and as a result there is less uncertainty in the genotype, particularly for the diploid model.

The prior genotype probabilities (expected genotype distributions) are also stored in the object for each possible ploidy. These distributions are estimated

based on the most likely parent genotypes. Low confidence parent genotypes can be ignored by increasing the `minLikelihoodRatio` argument to `PipelineMapping2Parents`.

```
> mydata$priorProb[[1]][,11:20]
```

```
  TP26698_0 TP26698_1 TP28405_0 TP28405_1 TP28602_0 TP28602_1 TP28986_0
0       0.0       0.5      0.25      0.25       0.5       0.0       0.5
1       0.5       0.5      0.50      0.50       0.5       0.5       0.5
2       0.5       0.0      0.25      0.25       0.0       0.5       0.0
  TP28986_1 TP31810_0 TP31810_1
0       0.0       0.5       0.0
1       0.5       0.5       0.5
2       0.5       0.0       0.5
```

```
> mydata$priorProb[[2]][,11:20]
```

```
  TP26698_0 TP26698_1 TP28405_0 TP28405_1 TP28602_0 TP28602_1 TP28986_0
0       0.0       0.0         0         0         0         0        NA
1       0.0       0.5         0         0         1         0        NA
2       0.5       0.5         1         1         0         0        NA
3       0.5       0.0         0         0         0         1        NA
4       0.0       0.0         0         0         0         0        NA
  TP28986_1 TP31810_0 TP31810_1
0        NA         0         0
1        NA         1         0
2        NA         0         0
3        NA         0         1
4        NA         0         0
```

Here we see some pretty big differences under the diploid and allotetrapliod models. For example, if TP24805 is behaving in a diploid fashion we expect F2-like segregation since both parents were heterozygous. However, if TP24805 is behaving in an allotetraploid fashion, we expect that both parents are homozygous at differing paralogous loci, resulting in no segregation.

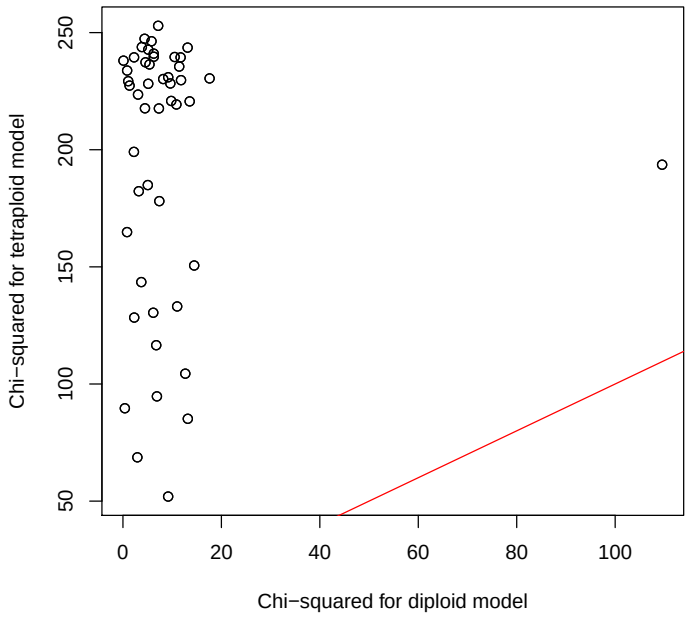
Now we want to determine which ploidy is the best fit for each locus. This is done by comparing genotype prior probabilities to genotype likelihoods and estimating a χ^2 statistic. Lower values indicate a better fit.

```
> mydata$ploidyChiSq[,11:20]
```

```
      TP26698_0  TP26698_1  TP28405_0  TP28405_1 TP28602_0 TP28602_1 TP28986_0
[1,]   6.156873   6.156873   7.390125   7.390125  17.60031  17.60031  2.107682
[2,] 130.424838 130.424838 178.066213 178.066213 230.45413 230.45413        NA
     TP28986_1   TP31810_0   TP31810_1
[1,]  2.107682   0.1076233   0.1076233
[2,]        NA 238.0402426 238.0402426
```

We can make a plot to get an overall sense of how well the markers fit the diploid versus tetraploid model.

```
> plot(mydata$ploidyChiSq[1,], mydata$ploidyChiSq[2,],
+      xlab = "Chi-squared for diploid model",
+      ylab = "Chi-squared for tetraploid model")
> abline(a = 0, b = 1, col = "red")
```



Alleles above the red line fit the diploid model better, and alleles below the red line fit the tetraploid model better. In this case it looks like everything is diploid with fairly high confidence.

Now we'll examine the posterior genotype probabilities. These are still estimated separately for each ploidy.

```
> mydata$posteriorProb[[1]][,3,11:20]
```

```
    TP26698_0   TP26698_1     TP28405_0     TP28405_1    TP28602_0    TP28602_1
0 0.00000e+00 4.95033e-28  1.000000e+00 2.131711e-174 1.000000e+00 0.000000e+00
1 1.00000e+00 1.00000e+00  2.142491e-17  2.142491e-17 2.384855e-10 2.384855e-10
2 4.95033e-28 0.00000e+00 2.131711e-174  1.000000e+00 0.000000e+00 1.000000e+00
    TP28986_0    TP28986_1   TP31810_0   TP31810_1
0 1.287625e-40 0.000000e+00 7.96211e-30 0.00000e+00
1 1.000000e+00 1.000000e+00 1.00000e+00 1.00000e+00
2 0.000000e+00 1.287625e-40 0.00000e+00 7.96211e-30
```

```
> mydata$posteriorProb[[2]][,3,11:20]
```

```
   TP26698_0  TP26698_1 TP28405_0 TP28405_1 TP28602_0 TP28602_1 TP28986_0
0 0.00000000 0.00000000         0         0         0         0       NaN
1 0.00000000 0.01442306         0         0         1         0       NaN
2 0.98557694 0.98557694         1         1         0         0       NaN
3 0.01442306 0.00000000         0         0         0         1       NaN
4 0.00000000 0.00000000         0         0         0         0       NaN
  TP28986_1 TP31810_0 TP31810_1
0       NaN         0         0
1       NaN         1         0
2       NaN         0         0
3       NaN         0         1
4       NaN         0         0
```

We can export the results for use in downstream analysis. The function below weights possible ploidies for each allele based on the results in `mydata$ploidyChiSq`, and for each taxon outputs a continuous, numerical genotype that is the mean of all possible genotypes weighted by genotype posterior probabilities. By default, one allele per locus is discarded in order to avoid mathematical singularities in downstream analysis. The continuous genotypes also range from zero to one by default, which can be changed with the `minval` and `maxval` arguments.

```
> mywm <- GetWeightedMeanGenotypes(mydata)
> mywm[c(297,2:6), 6:10]
```

```
                   TP26698_1  TP28405_1    TP28602_0    TP28986_0    TP31810_0
Zebrinus-Justin 5.110893e-01 0.50000014 2.485904e-18 3.232376e-39 5.000000e-01
Kaskade-Justin  1.521187e-06 0.49989345 5.002103e-01 5.000000e-01 1.583328e-20
Map1-001        4.998375e-01 0.98007584 1.773836e-02 5.000000e-01 4.998870e-01
Map1-002        1.126958e-02 0.47687372 4.822616e-01 1.740322e-24 1.129795e-04
Map1-003        1.126960e-02 0.03777935 1.773836e-02 5.000000e-01 1.129795e-04
Map1-005        1.126958e-02 0.50000000 4.746178e-01 1.578666e-42 4.998870e-01
```

Note that the parent weighted mean genotypes were estimated using genotype likelihood only, ignoring the priors set for the progeny. In some places they may not match the progeny genotypes, indicating a likely error in parental genotype calling.

4 Estimating genotype probabilities in a diversity panel

Pipelines in polyRAD for processing a diversity panel (i.e. a germplasm collection, a set of samples collected in the wild, or a panel for genome-wide association analysis or genomic prediction) use iterative algorithms. Essentially, allele frequencies are re-estimated with each iteration until convergence is reached.

Here we'll import a RAD-seq dataset from a large collection of wild and ornamental *Miscanthus* from Clark *et al.* (2014; doi:10.1093/aob/mcu084.

Since the data are in VCF format, we will need the Bioconductor package VariantAnnotation to load them. See `https://bioconductor.org/packages/release/bioc/html/VariantAnnotation.html` for installation instructions.

```
> library(VariantAnnotation)
> myVCF <- system.file("extdata", "Msi01genes.vcf", package = "polyRAD")
```

For your own VCF files, you will want to compress and index them before reading them. This has already been done for the file supplied with polyRAD, but here is how you would do it:

```
> mybg <- bgzip(myVCF)
> indexTabix(mybg, format = "vcf")
```

Now we can make our `RADdata` object. Because this is a small example dataset, we are setting `expectedLoci` and `expectedAlleles` to very low values; in a real dataset they should reflect how much data you are actually expecting. It is best to slightly overestimate the number of expected alleles and loci.

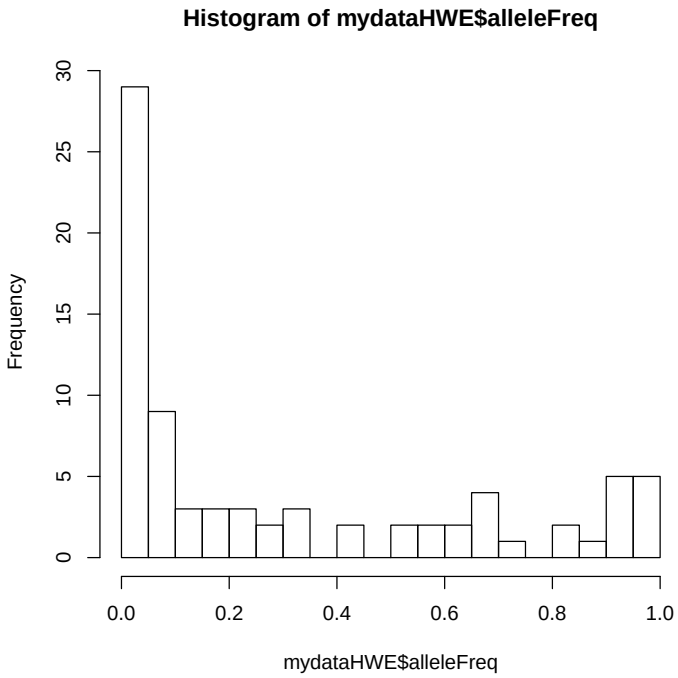
```
> mydata <- VCF2RADdata(myVCF, possiblePloidies = list(2, c(2,2)),
+                       expectedLoci = 100, expectedAlleles = 500)
> mydata

## RADdata object ##
585 taxa and 24 loci
422433 total reads
Assumed sample cross-contamination rate of 0.001

Possible ploidies:
Autodiploid (2)
Allotetraploid (2 2)
```

We can iteratively estimate genotype probabilities assuming Hardy-Weinberg equilibrium. The argument `tol` is set to a higher value than the default here in order to help the tutorial run more quickly.

```
> mydataHWE <- IterateHWE(mydata, tol = 1e-3)
```

Let's take a look at allele frequencies:

```
> hist(mydataHWE$alleleFreq, breaks = 20)
```

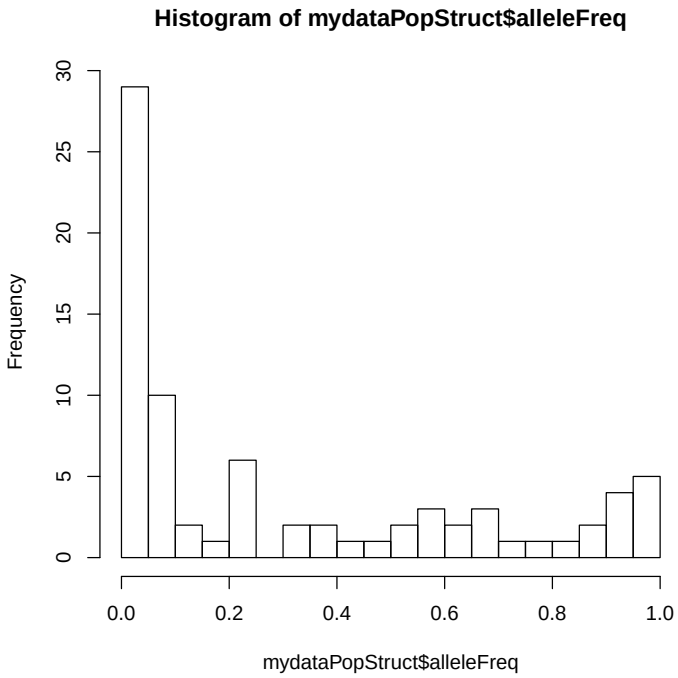


We can do a different genotype probability estimation that models population structure and variation in allele frequencies among populations. We don't need to specify populations, since principal components analysis is used to assess population structure assuming an isolation-by-distance model, with gradients of gene flow across many groups of individuals. This dataset includes a very broad sampling of *Miscanthus* across Asia, so it is very appropriate to model population structure in this case.

```
> mydataPopStruct <- IteratePopStruct(mydata, nPcsInit = 8)
```

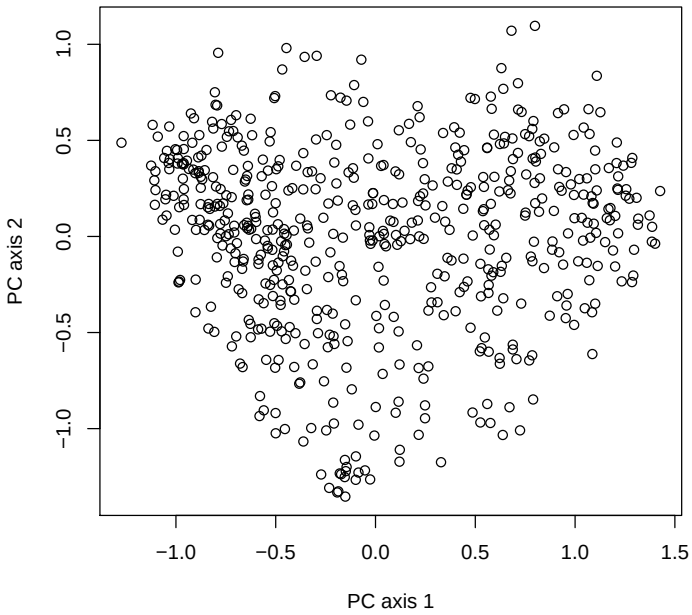
Allele frequency estimates have changed slightly:

```
> hist(mydataPopStruct$alleleFreq, breaks = 20)
```



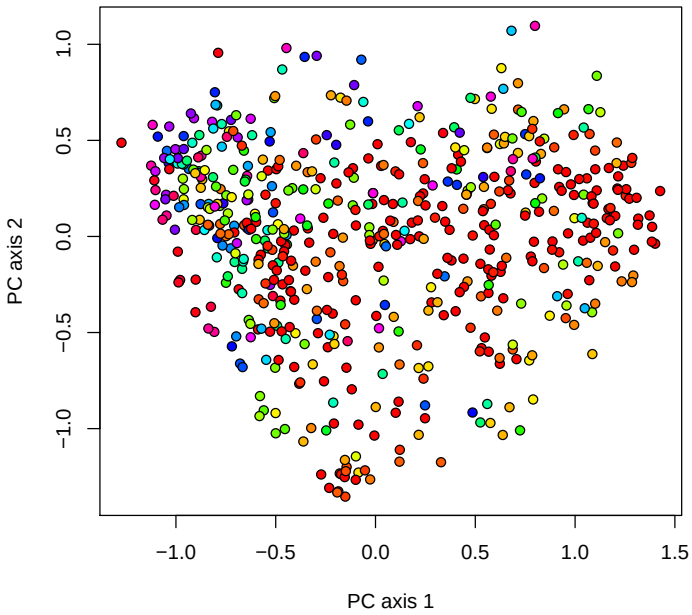
Here's some of the population structure that was used for modeling allele frequencies (fairly weak in this case because so few markers were used):

```
> plot(mydataPopStruct$PCA[,1], mydataPopStruct$PCA[,2],
+      xlab = "PC axis 1", ylab = "PC axis 2")
```



And here's an example of allele frequency varying across the environment. Allele frequencies were estimated for each taxon, and are stored in the `$alleleFreqByTaxa` slot.

```
> myallele <- 1
> freqcol <- rainbow(101)[round(mydataPopStruct$alleleFreqByTaxa[,myallele] * 100) + 1]
> plot(mydataPopStruct$PCA[,1], mydataPopStruct$PCA[,2],
+      xlab = "PC axis 1", ylab = "PC axis 2", pch = 21,
+      bg = freqcol)
```



As before, we can export the weighted mean genotypes for downstream analysis.

```
> wmgenoPopStruct <- GetWeightedMeanGenotypes(mydataPopStruct)
> wmgenoPopStruct[1:10,1:5]
```

```
                    S01_138045_G S01_139820_TT S01_139820_CT S01_139883_G
KMS207-8             4.009598e-04    0.13329403    0.06050896  0.055179238
JM0051.003           3.619411e-01    0.29614485    0.17795718  0.003882082
JM0034.001           1.000000e-04    0.00010000    0.07308096  0.022850910
JM0220.001           5.635472e-01    0.00010000    0.17198992  0.070508556
NC-2010-003-001      1.000000e-04    0.00010000    0.07122964  0.001761740
JM0026.001           1.000000e-04    0.00010000    0.11795221  0.012772340
JM0026.002           1.363591e-01    0.08578502    0.10825386  0.002586923
PI294605-US64-0007-01 1.623164e-08    0.00323793    0.03985575  0.070701723
JM0058.001           6.296405e-01    0.20767383    0.19900257  0.000100000
UI10-00086-Silberfeil 8.785122e-01    0.00010000    0.09714697  0.001359814
                    S01_150928_GG
KMS207-8             1.253158e-11
JM0051.003           1.345516e-06
JM0034.001           5.351637e-06
JM0220.001           9.306208e-10
NC-2010-003-001      5.351095e-14
```

```
JM0026.001              9.643922e-13
JM0026.002              6.828821e-10
PI294605-US64-0007-01   4.969013e-30
JM0058.001              1.217802e-07
UI10-00086-Silberfeil   3.175026e-30
```

If you expect that your species has high linkage disequilibrium, the functions `IterateHWE_LD` and `IteratePopStructLD` behave like `IterateHWE` and `IteratePopStruct`, respectively, but also update priors based on genotypes at linked loci.