

# Tutorial for: Training to Improve Applications of Genetics and Genomics for Biodiversity Conservation

Jared Grummer

2023-07-22

## Read in the wolf data

These data were taken from a publication of wolf reintroduction in 1997, titled “Genetic structure and migration in native and introduced Rocky Mountain Wolf populations, by Forbes and Boyd. Gordon will discuss this dataset more during day 2. But briefly, wolves were collected from four sites in the Rocky Mountains (US + Canada) and introduced to Yellowstone National Park in an attempt to re-establish wild populations in Yellowstone. In these scenarios, genetic analyses can be a good tool to quantify how divergent introduced populations have become from their source population founders, how genetically diverse the introduced populations are, etc. We will use data from the Montana source population and Yellowstone NP introduced population to see how genetically differentiated the populations are through structure, principal components, and discriminant analysis of principal components analyses. These same data will be used later in assignment tests to see if we can assign introduced individuals to their source populations.

The dataset consists of 31 individuals from Yellowstone and 66 from Montana; all individuals were sequenced at 10 microsatellite loci. The dataset is in GenePop format, generally where individuals are rows and loci are columns. Individuals are assigned to populations (could be sampling sites, sex, all individuals in one populations, etc.). The GenePop format is as follows: First line: any characters (could be information on your dataset) Lines 2 - (# loci + 1): names of markers, one name per line Line (# loci + 2): enter the word “POP” (or “Pop” or “pop”) The following lines are individual-level data, that look like this: ind1, 0101 0103 0202...

The individual name is listed first, then the two alleles are listed for each locus, followed by a space, then the two alleles for the next locus, etc. Following all individuals of one population, the word “POP” appears again, under which the individual-level data for that population is entered.

The R package *adegenet* can read in data files with the GenePop format, and we will use the *read.genepop* function to do so.

## See a summary of the dataset

We can take a peek at the dataset to make sure it was read in properly, and see some summary-level information for number of individuals, loci, etc.

---

**Q: How many individuals are in this dataset?**

**Q: How many loci, and alleles per locus, are in this dataset?**

---

```
## /// GENIND OBJECT //////////  
##  
## // 97 individuals; 10 loci; 50 alleles; size: 39.6 Kb
```

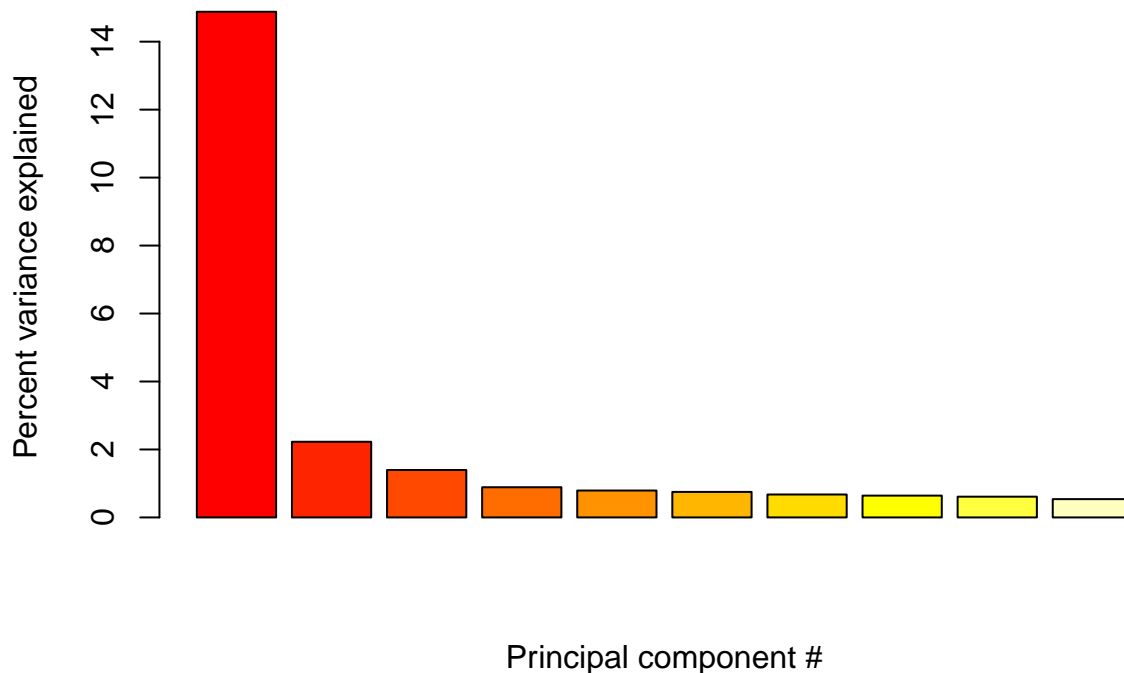
```
##
## // Basic content
##   @tab: 97 x 50 matrix of allele counts
##   @loc.n.all: number of alleles per locus (range: 2-8)
##   @loc.fac: locus factor for the 50 columns of @tab
##   @all.names: list of allele names for each locus
##   @ploidy: ploidy of each individual (range: 2-2)
##   @type: codon
##   @call: read.genepop(file = "~/Documents/UM/Conferences/2023_ICCB/Workshop/wolves/Yellowstone Mont
##         ncode = 2L, quiet = FALSE)
##
## // Optional content
##   @pop: population of each individual (group size range: 31-66)
```

## Perform a PCA on the data

To see how individuals and populations are structured on the landscape, we will first perform a principal component analysis (PCA) on the allele frequencies.

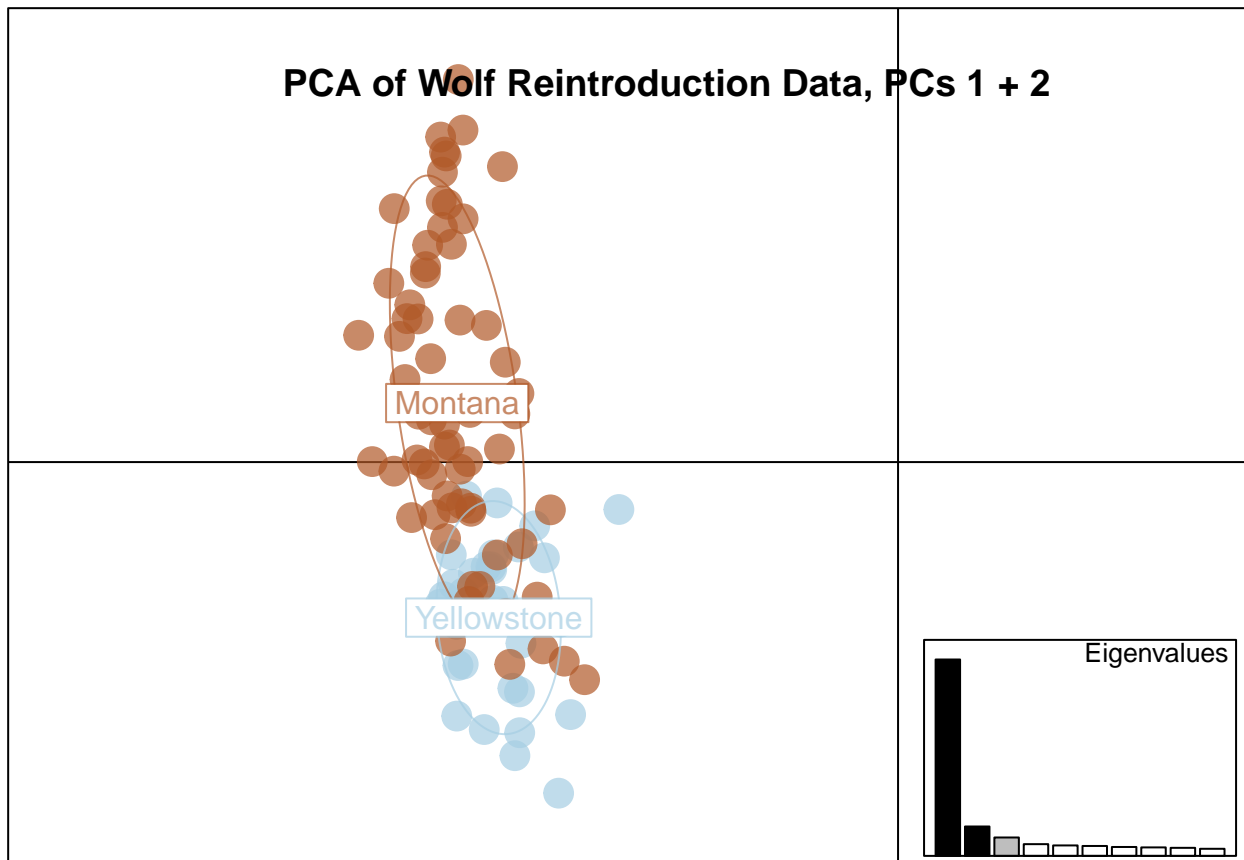
It's often useful to see how the variance in the dataset is partitioned among the principal components (PCs). Ideally, the first 2-3 PCs explain most of the variance (~75%+) in allele frequencies. Let's look at the amount of variance represented by each of the first ten principal components:

### PCA eigenvalues

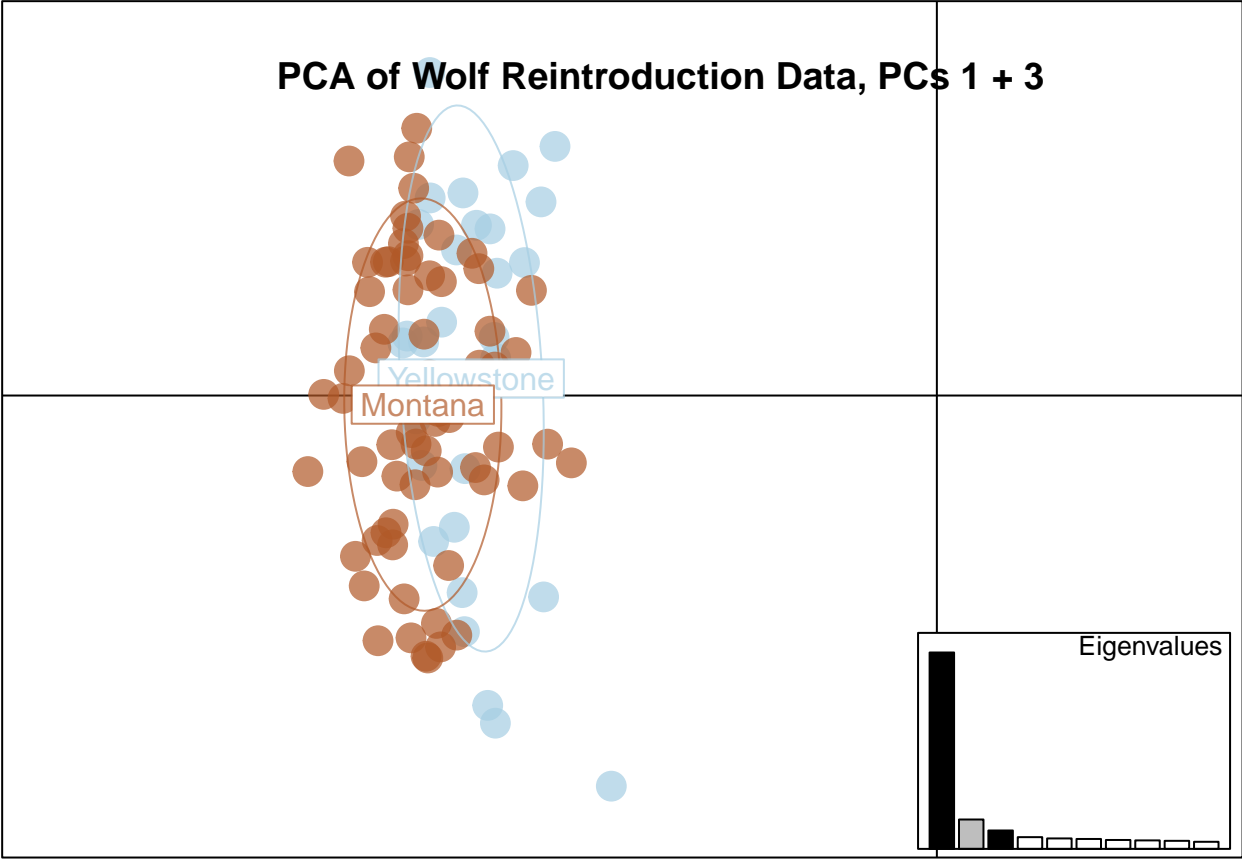


Note that the majority of variance explained in allele frequencies is in the first three (and really first one) axes.

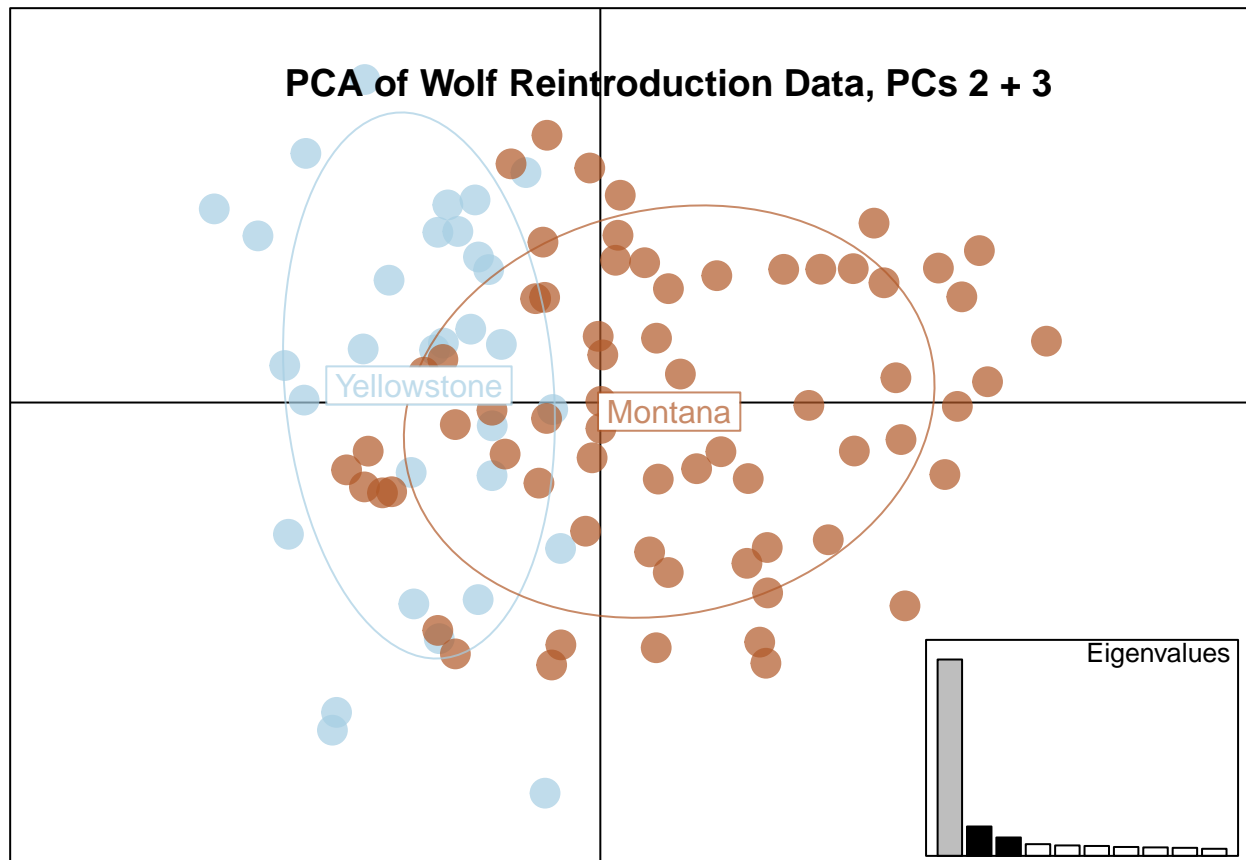
Now, we will make of plot of the first two PCs to see how genetically distinct (or not) these populations are.



Because we can only visualize the values of two principal components at a time, it is sometimes useful to view plots of PCs 1 + 3 or 2 + 3, depending on how the variance is partitioned among the top PCs. Here, let's look at 1 + 3:



And now at PCs 2 + 3:



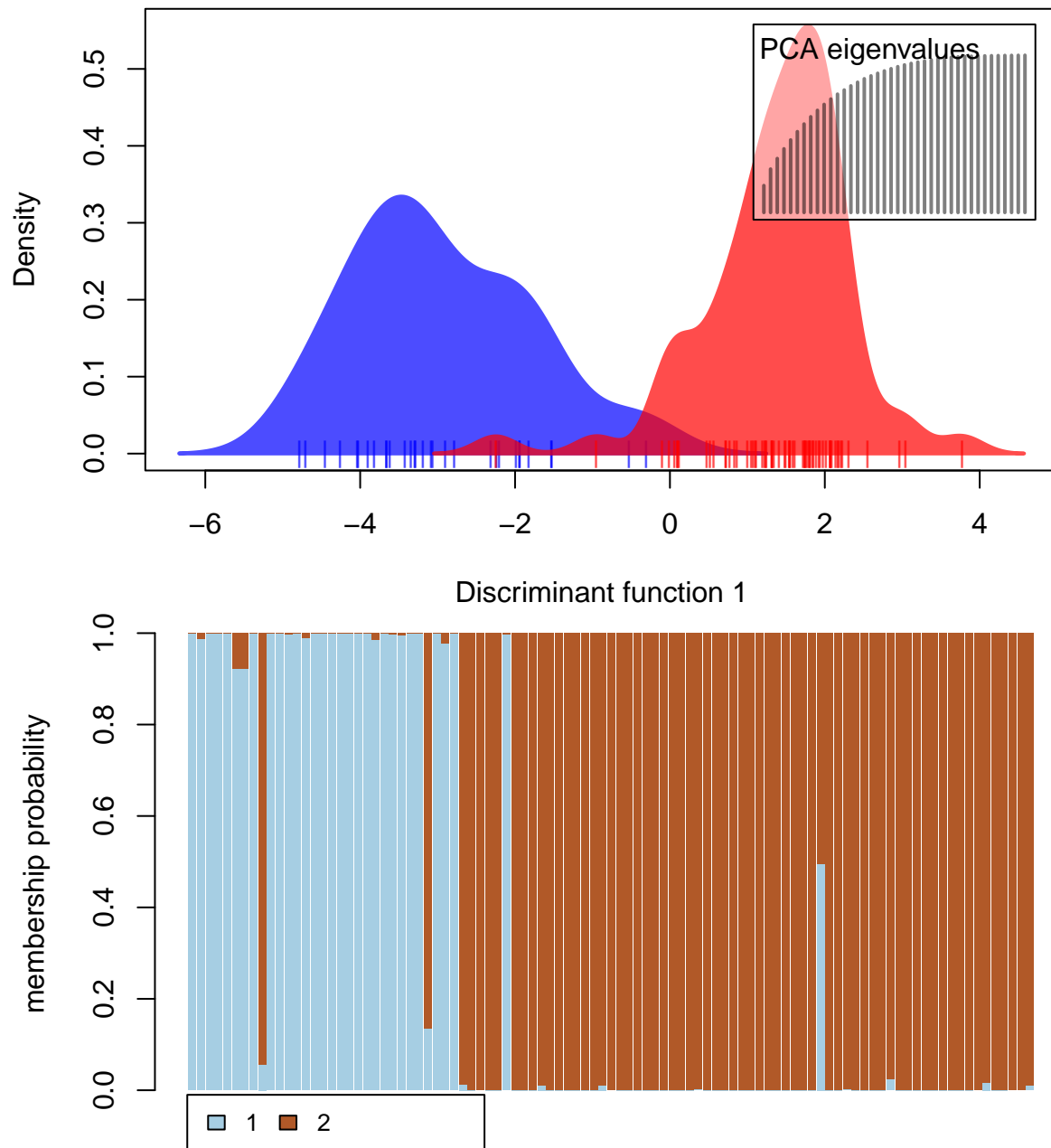
**Q:** Does the relationship between/structure of populations change based on which PCs are visualized? Why might this be?

## Discriminant Analysis of Principal Components (DAPC)

A DAPC analysis is similar to PCA, however, the DAPC will maximize separation between the groups (or clusters/populations) and can therefore be useful in some cases. We are going to look at performing a DAPC in two ways. First, we will use the *a priori* group membership assignments based on where the individuals were sampled. Next, we will use the “*find.clusters()*” command within *ade4* to partition the dataset into genetically distinct clusters.

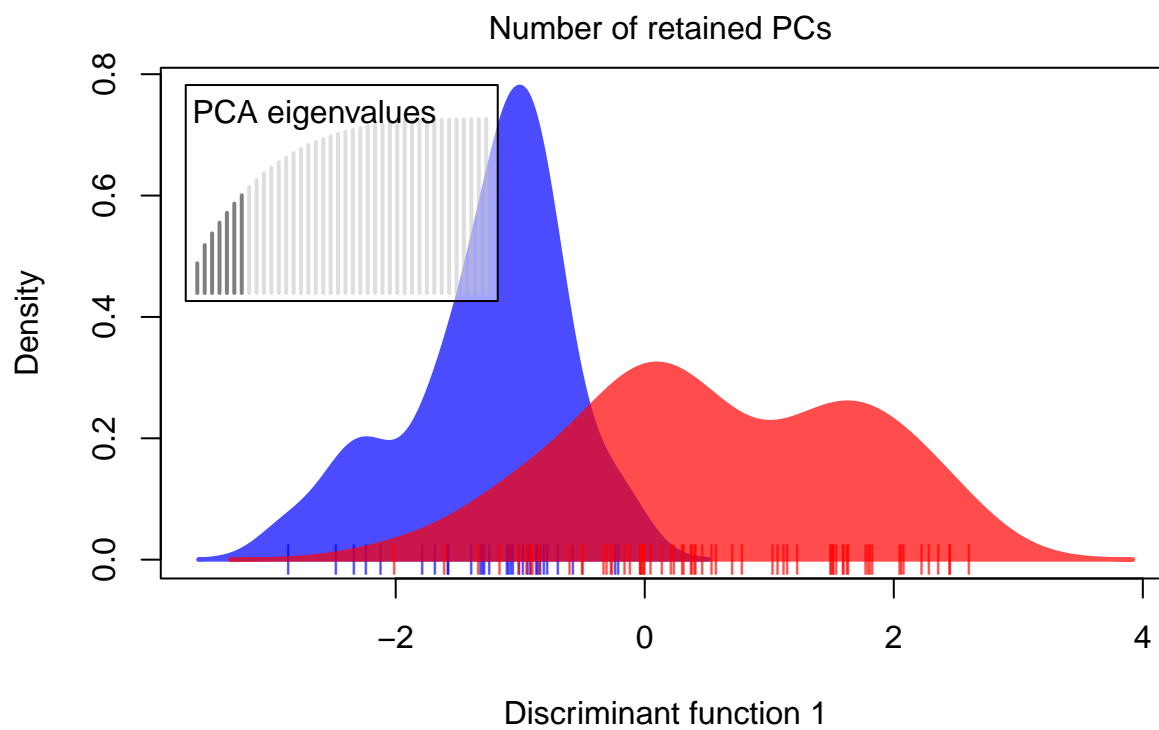
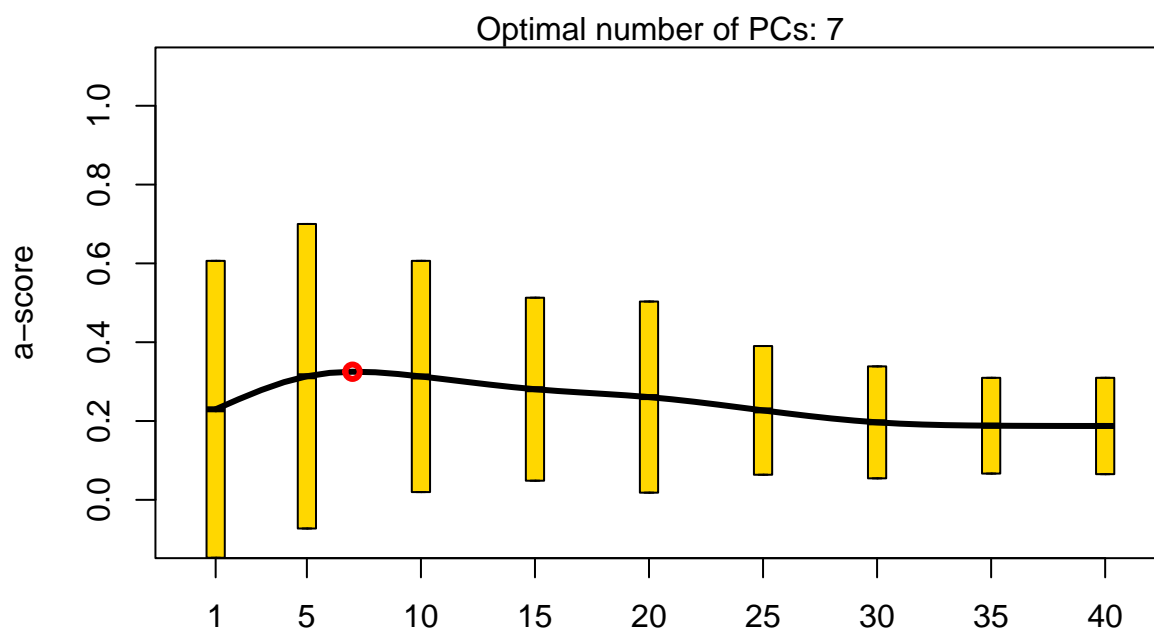
We will perform the DAPC in *ade4*, which behind-the-scenes uses the *ade4* package for the analysis. For the code below, set `max.n.clust` to highest number of biologically plausible populations

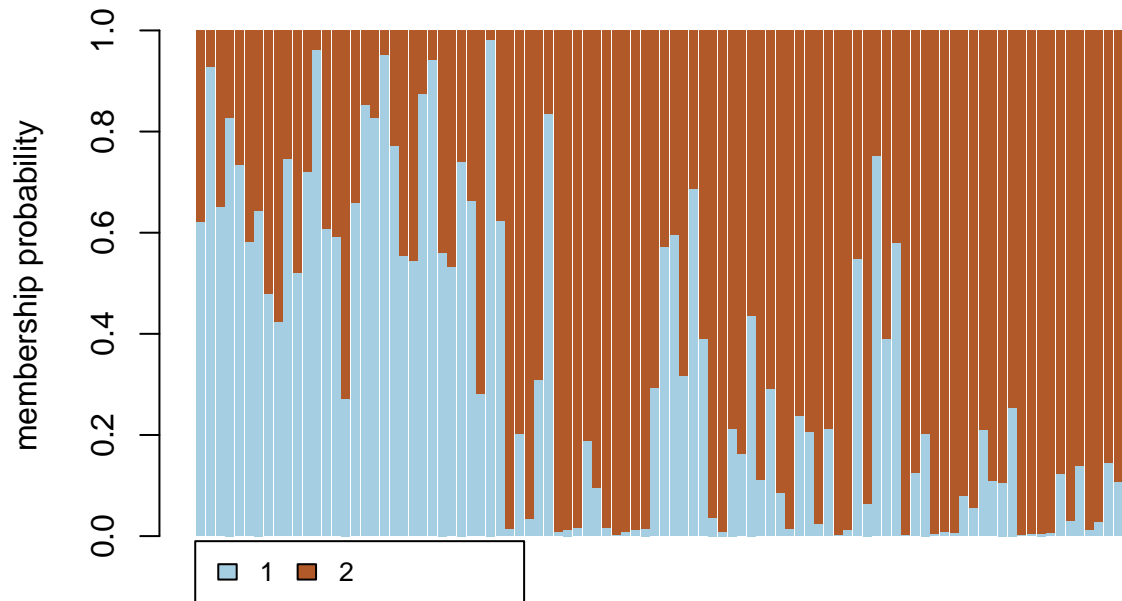
To start, set the “`n.pc`” option to about 90-95% of your individuals. You can set this lower after seeing how the variance explained is related to the number of PCs in the analysis.



The number of principal components included in the analysis can have a strong influence on identifying the number of clusters and assignment of individuals to those clusters. Let's find the optimal number of PCs to use in DAPC analysis, then re-do the DAPC analysis with that number of PCs.

## a-score optimisation – spline interpolation






---

**Q: How did results of population structure change when the number of PCs included in the analysis changed?**

---

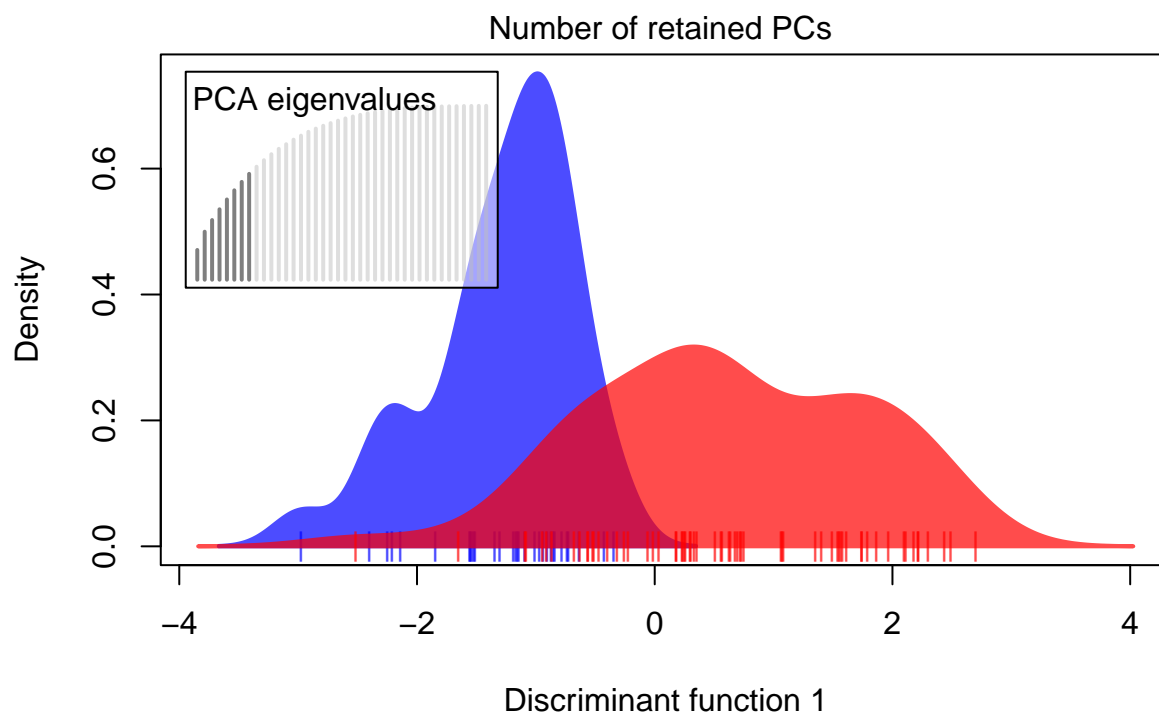
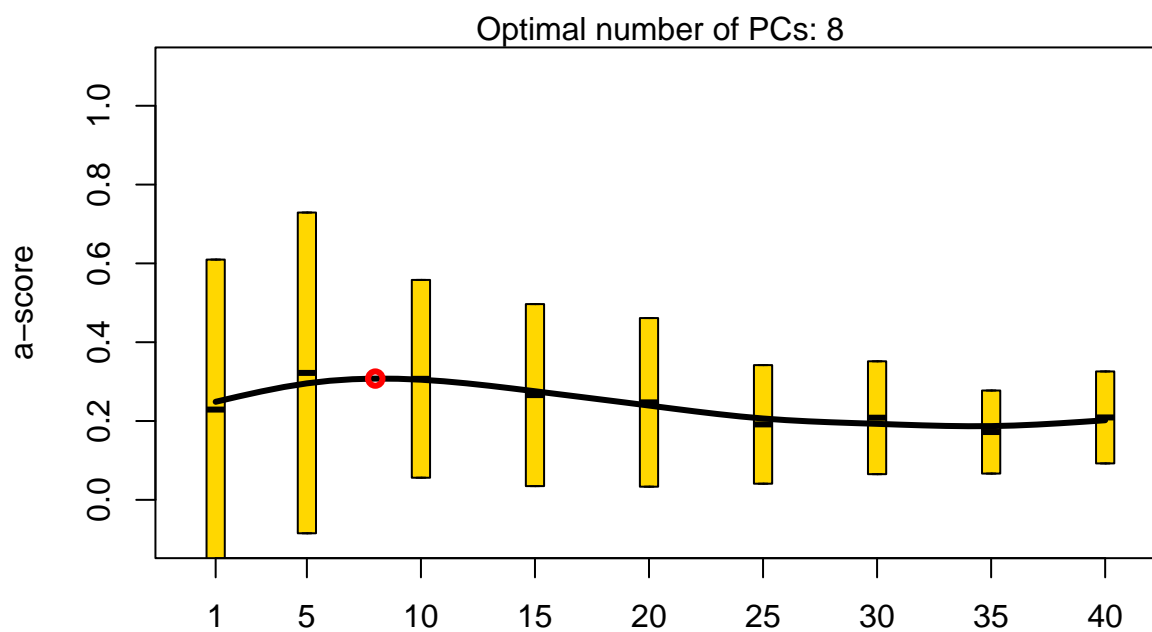
Now, let's not use our *a priori* expectations about population membership and let the data and “*find.clusters()*” determine the number of populations. In the code below, the “*find.clusters*” command will prompt you to select the optimal number of clusters ( $k$ ). This is usually the inflection point where the BIC value is the lowest. However, note that it is often beneficial to plot the results of multiple  $k$  values, given that one model may not be significantly better than the others.

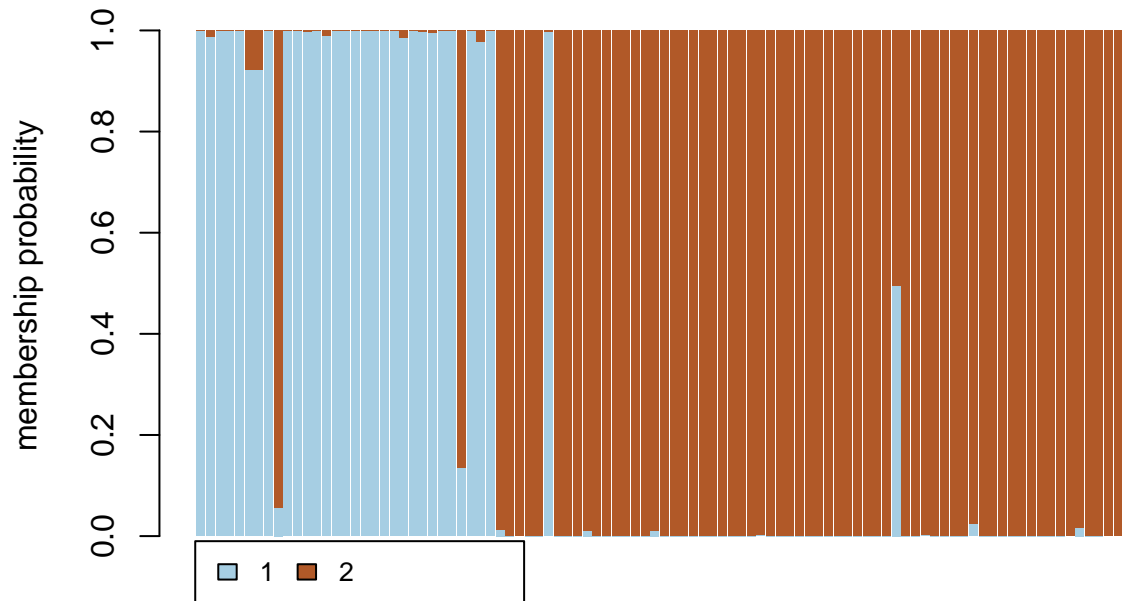
Again, we choose the number of PCs to be 85-95% of the total number of individuals.

Let's again find the optimal number of PCs to use in DAPC analysis, then re-do the DAPC analysis with that number of PCs.



## a-score optimisation – spline interpolation





In this case, 10 principal components are deemed optimal to use.

---

**Q:** How do the results differ from the “*find.clusters()*” result as compared to the DAPC of population assignments based on sampling location?

**Hint:** You can access individual group assignments in the “\$grp” slot of the DAPC objects

## Assessing population structure using the program STRUCTURE

Okay, now we are going to look at population structure in the program STRUCTURE (*Pritchard et al. (2000); Falush et al. (2003)*). This is a great program to visualize how an individual’s genome (represented by our genetic data as a subset of the entire genome) is apportioned to distinct populations. I.e., an individual’s genetic makeup could be represented by a mixture of two populations, which could be either due to drift and descent from an ancestral population, or through migration/admixture. In the case of the wolf data, we can see how distinct (or not) the individuals who are descended from transplanted individuals are from the population that the transplanted individuals were taken from (hopefully that makes sense!).

This program has two distinct ways to be run - either through the graphical user interface (GUI), or command-line. We’ll mainly look at how to run the program through the GUI, but I also provide files for running through the command-line. The command-line has the benefit of easily repeating analyses and running multiple distinct runs by just modifying options in a flat text file, rather than having to click through the various menus of the GUI version.

Another thing to consider is file format conversions. Many programs make use of a file format that is particular to that program (unfortunately!). This means that a file conversion needs to be done to put your data into the correct input format needed for that program. This either requires coding skills and your ability to write conversion scripts (in something like R or Python), or finding scripts or functions online that perform the conversions for you. For the STRUCTURE file that you are using, I converted it using an R function that I found online, to convert a *genind* object (*adeigenet* format) to the STRUCTURE format.

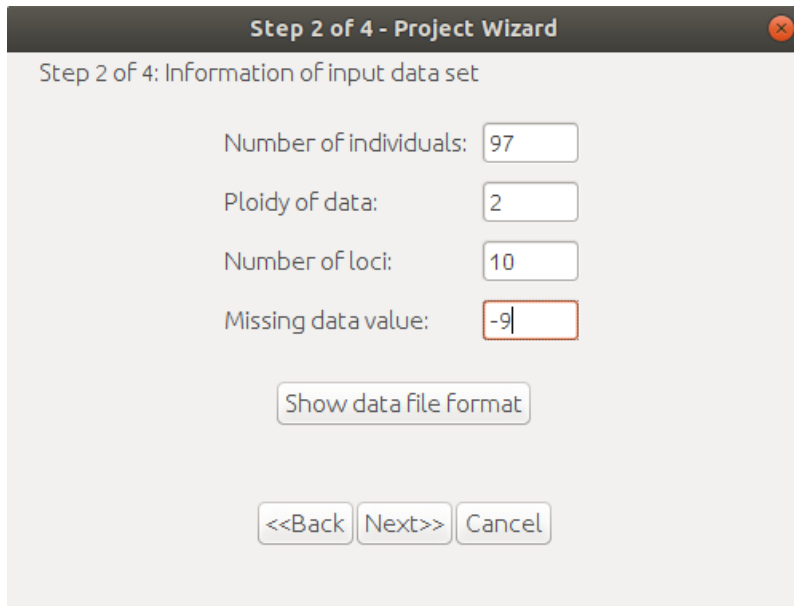
We are going to use STRUCTURE to tell us how many distinct populations are in the data (that fulfill Hardy-Weinberg assumptions), and then how each individual’s genetic data are assigned to those populations.

Determining the optimal number of populations ( $k$ ) in the data is somewhat of a philosophical issue (and we will not discuss it here!), but practically, this is often done through the Evanno et al. (2005) method. We will not discuss that method in detail, but know that it looks for the largest change in likelihood between competing models (that differ in how many populations/clusters individuals are assigned to). So to do this, the STRUCTURE analysis must be re-ran many times, assigning individuals to either 1, 2, 3... or  $n$  clusters, based on how many clusters you might think are in your data (and you always choose the highest number of clusters to be higher than how many clusters you think are actually in your data).

For the wolf data, our prior knowledge tell us that there may be just two populations in the data, so we will run STRUCTURE with  $k$  values of 1 - 5, each analysis with a single  $k$  value, where the program tries to assign an individual's genome into that number of clusters. For each  $k$  value, we will run five replicate analyses to make sure that our results are consistent. So, for five  $k$  values (1-5), we will run 25 separate analyses (and you quickly see why the command-line version is preferred!). We will first show you how to run a single analysis in the GUI version, then give you the resources to perform the same analyses in the command-line version.

Okay, once you have the appropriate version of STRUCTURE downloaded on your computer ("front end" = GUI; operating system specific <https://web.stanford.edu/group/pritchardlab/structure.html>), launch the program by double-clicking on the program icon. We will now import our dataset. Have a look at the "Yellowstone Montana wolf data set.str" file in a text editor. The format comes in two types: one version has both alleles at a locus on the same row (each individual is on a single row), and the other version displays each individual with two rows and one of two alleles for each locus is given on one of two rows; our dataset is in the second form, with marker/locus names given on the first line (just numbers 1 - 10).

Go to **File > New Project** and fill in the information for Step 1. For Step 2, fill in the size of our dataset. We don't have missing data in this dataset, but often "-9" is used for missing data:



Step 2 of 4 - Project Wizard

Step 2 of 4: Information of input data set

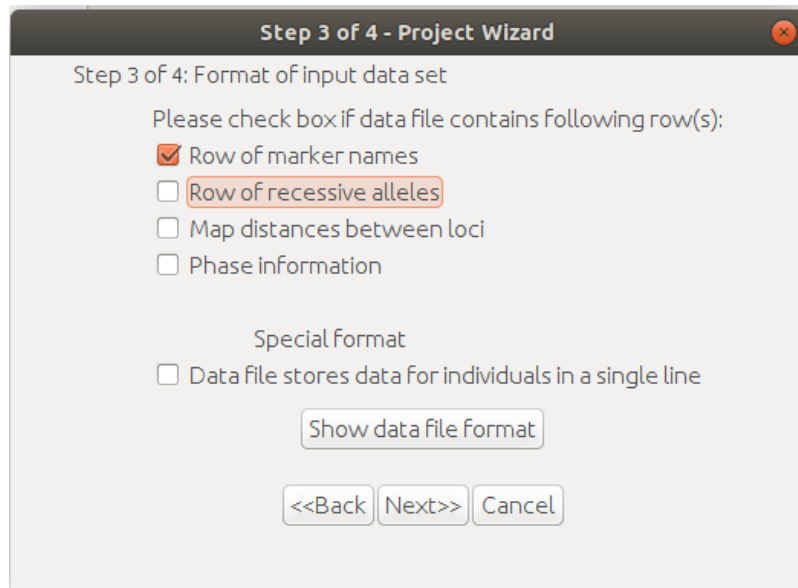
Number of individuals:

Ploidy of data:

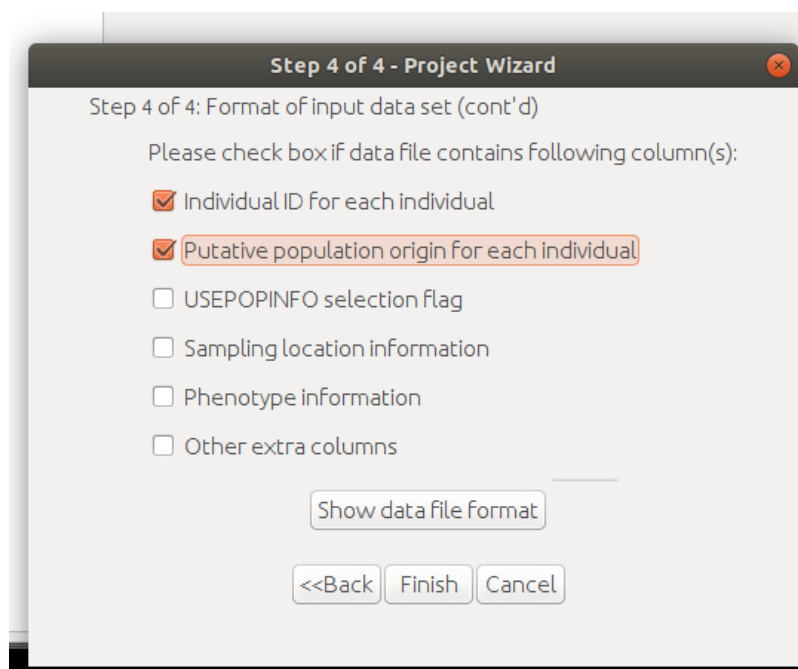
Number of loci:

Missing data value:

For Step 3, select the "Row of marker names" option:



And for Step 4, select the top two options (because we have *a priori* expectations of population assignment based on our prior knowledge):



Click **Finish**. If the data format is correct and you entered in the options correctly, you shouldn't get any errors and the data should display correctly.

Now go to **Parameter Set > New**, then the **Run Length** tab. Enter a Burnin Period of 50000 generations and a Number of MCMC Reps after Burnin = 75000. Note that, as for any Bayesian analysis, choosing the number of generations in the analysis is important. You don't want to be there forever and run the analysis unnecessarily long, but you also don't want to run it too short and not give the program a chance to converge. You will have to explore results and ensure that your analyses have converged, but for this analysis, these run lengths are good.

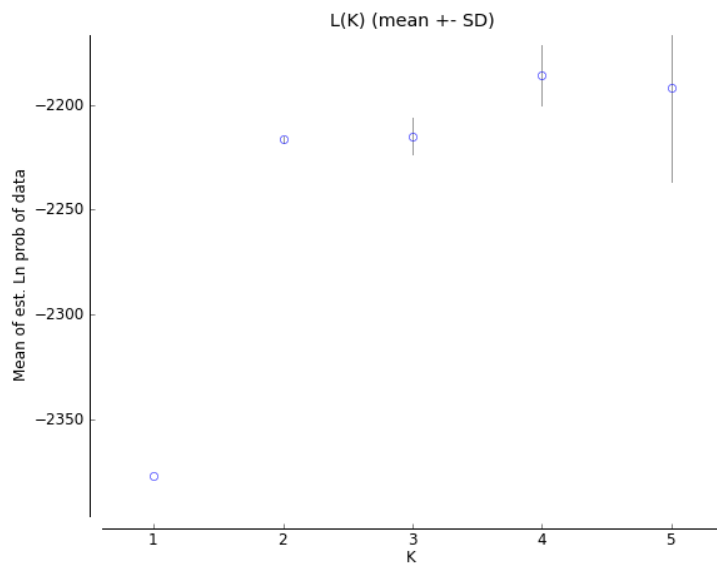
Now go to the **Ancestry Model** tab, and select the **Use Admixture Model** option. Next, in the **Allele Frequency Model** tab, select the **Allele Frequencies Correlated** option. Lastly, in the **Advanced** tab,

check the box for the “Compute probability of the data (for estimating K)” option. Click OK and then name the parameter set (e.g., “Params”, or whatever you wish).

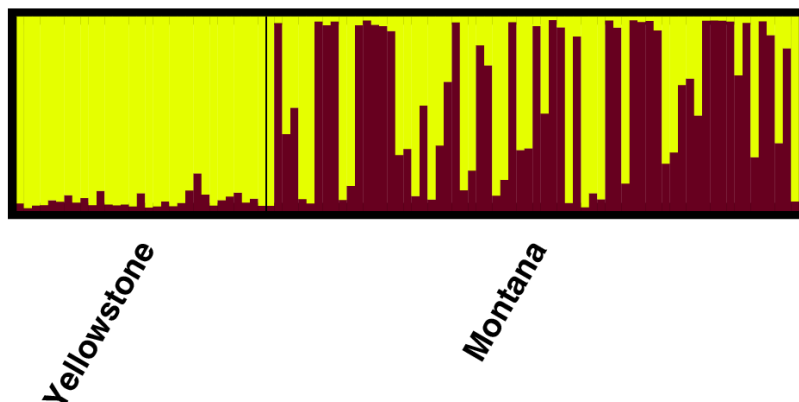
We are now ready to run the analysis! To initiate a run, go to **Parameter Set > Run**. Enter how many populations you assume (see discussion above regarding running a range of  $k$  values), then hit OK. Your analysis should now run! With a dataset this size (~100 individuals, 10 loci), each  $k$  replicate should take less than half a minute, with the higher  $k$  runs taking a little longer. But, for the purposes of this workshop, we’ve provided you with results files from running the wolf data with  $k = 1 - 5$ , in the folder “Structure Results”.

You can open up one of the results files (no extension, just ends in “\_f”, with the base output name you specified, or same base name as the input file) to see the information given. We will use the output files to summarize the variation in individual assignments to clusters ( $q$  values) and log-likelihood estimates of the model, across the replicate analyses. We will do this in the web-based program “STRUCTURE Harvester” (Earl and vonHoldt 2012; <https://taylor0.biology.ucla.edu/structureHarvester>). We provide you with these results as well, but all you need to do is zip all of your results files (ending with “\_f”) together into a single folder, then upload that to the STRUCTURE Harvester website, then it gives you a bunch of files that summarize your STRUCTURE results!

As one way to see that you’ve run your analyses long enough, look at the *meanLnProb.pdf* file and see the variation in the log-likelihood scores by  $k$  value. It is natural for the variance to increase with higher  $k$  values. The Evanno et al. (2005) method sees the largest change in log-likelihood between  $k = 1$  and  $k = 2$  (also seen in the *lnPPK.pdf* and *deltaK.pdf* files), and uses this to select  $k = 2$  as the optimal number of clusters/populations.



Now that we have established that  $k = 2$  is the optimal model for our data, we can plot the data to see how each individual’s genetic makeup is apportioned to these two populations. The GUI version of STRUCTURE produces the “bar plots” that you are probably familiar with, where each column is an individual and the amount of each color for that individual indicates how much of its genome (or genetic data) is derived from each population/cluster. The plots provided in the STRUCTURE GUI are okay for visualizing data during analysis, but for publication-quality figures, it’s nice to use the programs CLUMPP (<https://rosenberglab.stanford.edu/clumpp.html>) and Distruct (<https://rosenberglab.stanford.edu/distruct.html>). We will not go into detail on how to use these programs, but they are easy to use and well documented. STRUCTURE Harvester provides the .indfile and .popfile needed to run CLUMPP, which then produces the files to be run through Distruct that makes the bar plot. Here is the barplot produced by these programs, from  $k = 2$ :



**Q: What is your interpretation of this result?**

**Q: How does this result compare to the DAPC and PCA results?**

## Running STRUCTURE at the command-line

We have just seen how to run a single STRUCTURE analysis through the GUI/front-end version of the program. However, for running replicate analyses, across multiple  $k$  values, it's often faster and easier to use the command-line version. Pre-compiled command-line versions for each operating system are available at the STRUCTURE website previously linked. Running the command-line version is actually quite simple (don't be intimidated!). During your GUI run, two files should have been produced named "extraparams" and "mainparams"; we will use those in the command-line version to specify many of the same parameters (e.g., dataset size, run length, etc.).

At the command line, all we need to specify is the location of the STRUCTURE executable, the number of assumed populations (**-K** option), name for results file (**-o** option), and a random seed number (**-D** option, which is particularly helpful when troubleshooting). By default, the program will look for the "extraparams" and "mainparams" files in the same folder as the input data file ("Yellowstone Montana wolf data set.str"), whose location is specified in the "mainparams" file. Also, the options specified at the command line (such as "-o") will take precedent over whatever is written in the mainparams file. The STRUCTURE documentation .pdf covers all the options in the mainparams and extraparams files, but again, these came from the options you specified in the GUI version.

Here is an example command that could be run at the command line:

```
~/Desktop/Applications/Structure/command_line/structure -K 1 -o /home/jared/Desktop/ICCB_Workshop/Structure/Results/k1_run_1 -D 1356091
```

"~/Desktop/Applications/Structure/command\_line/structure" is the location of the STRUCTURE executable

"-K 1" tells the program to assign individuals to a single population

"-o /home/jared/Desktop/ICCB\_Workshop/Structure/Results/k1\_run\_1" indicates that the location and base name ("k1\_run\_1") of the results outfile; and

"-D" gives a random number seed

These paths are relative to my computer, thus you need to change the file paths to match the folder structure of your computer. With the correct paths for your computer, and files in the correct place, you can simply execute this command at the command line and STRUCTURE should run! See the file "Structure\_command\_line\_args.sh" to see a BASH script that will execute many such commands (e.g., replicate analyses with different random number seeds)!