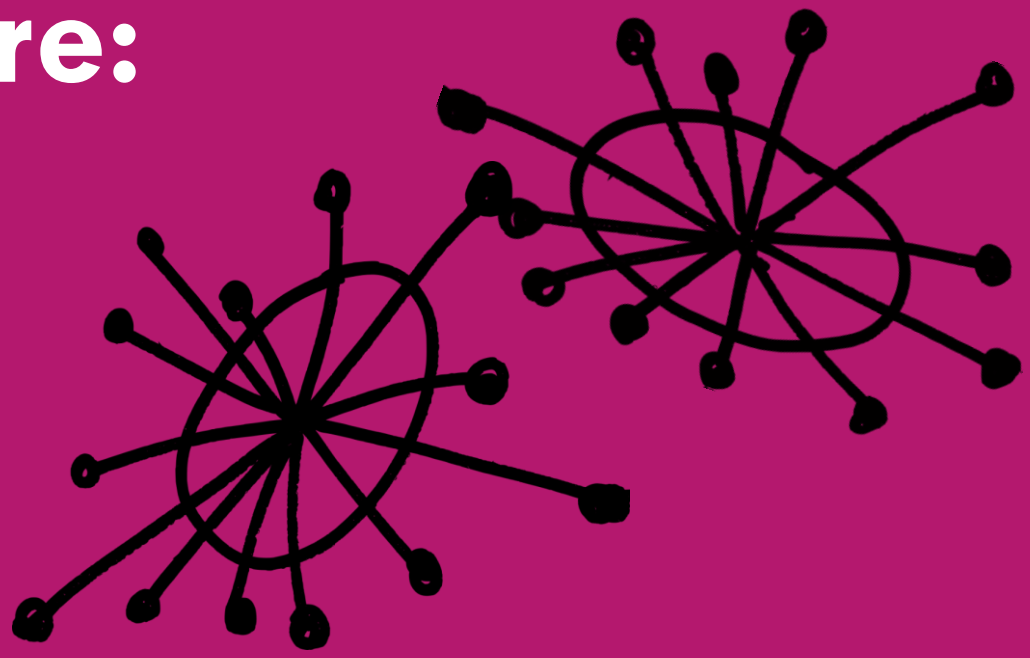
From SNPs to structure: population genetic analyses using R's *adegenet* package

Joshua Thia



Outline

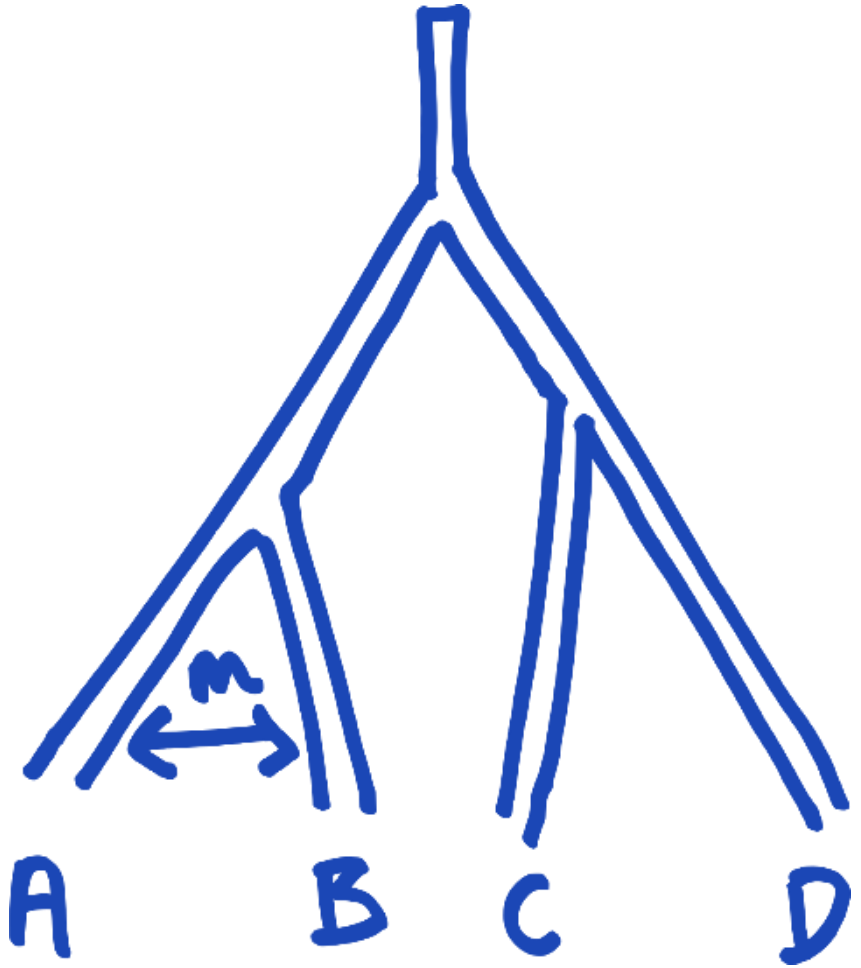
Some philosophy

A-bit-more-than-some theory

A lot of *R*

Population genetic data

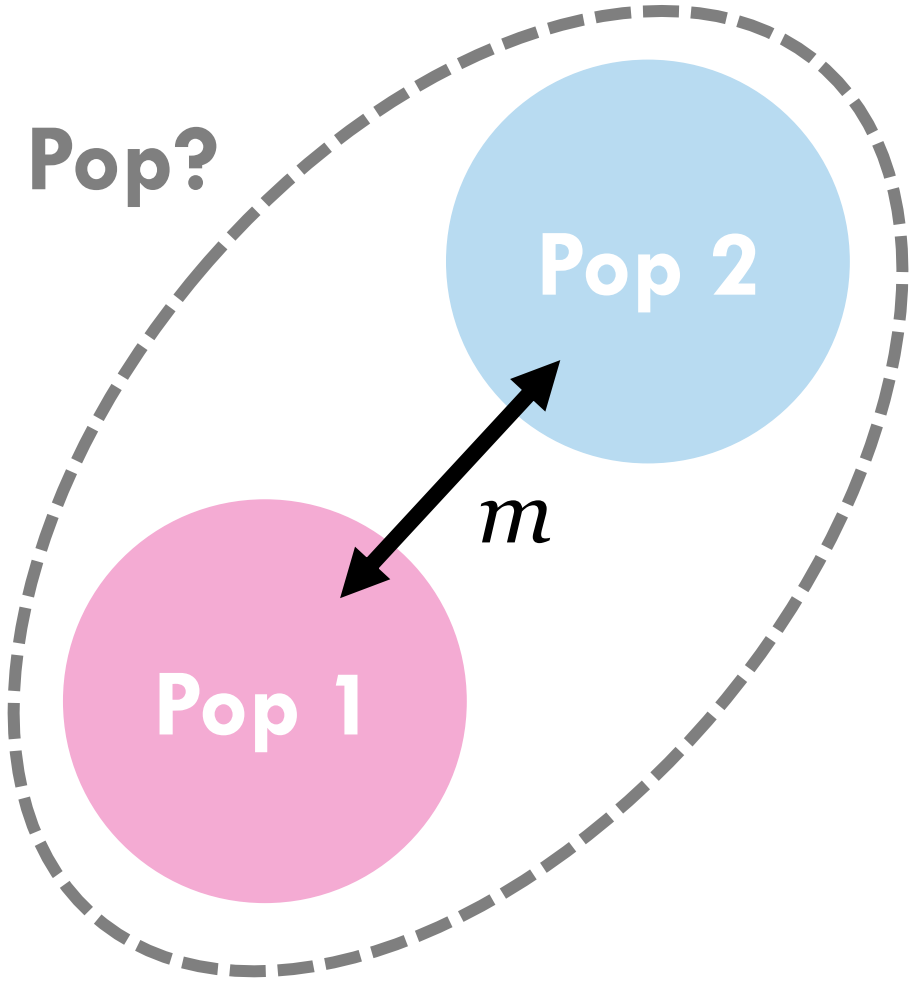
- Genetic variation is affected by eco-evolutionary processes
- We want to understand something about natural populations
- We indirectly infer processes from patterns in genetic data



Molecular ecology questions

How different are populations?

What even are the populations?



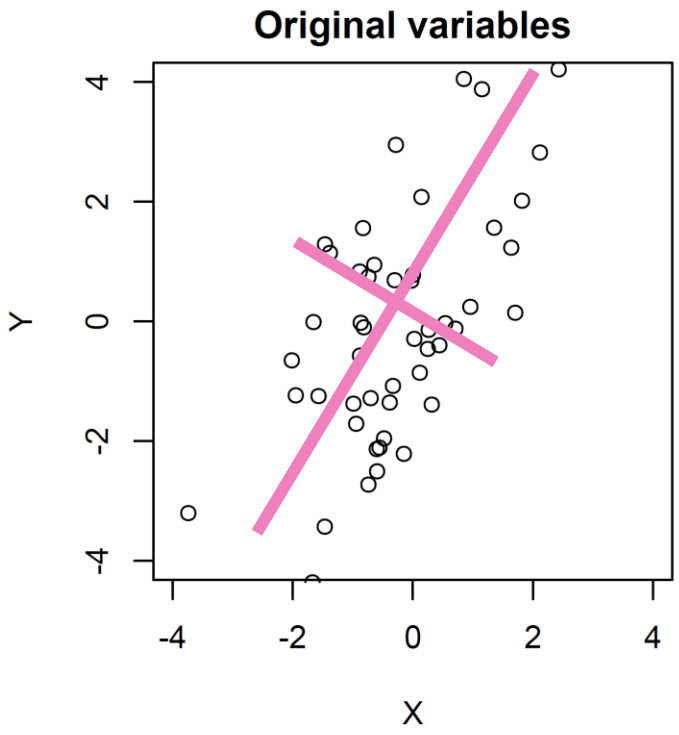
Detecting structure with *STRUCTURE*

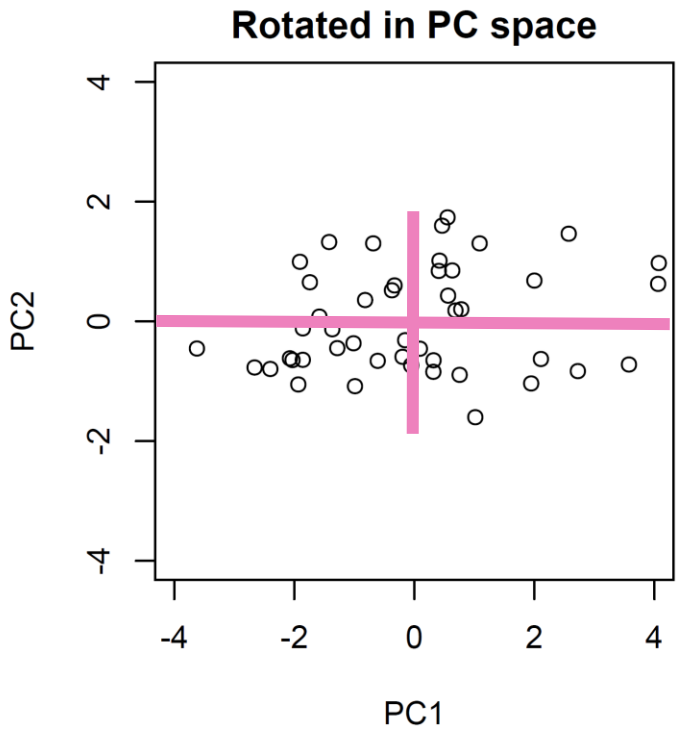
- Bayesian clustering
- Linkage equilibrium between loci within populations
- Requires parameterisation, originally formulated for “classic” multilocus pop gen data
- 1000s of genomic SNPs and many samples = computational nightmare
- Not in *R* ☹

web.stanford.edu/group/pritchardlab/home.html

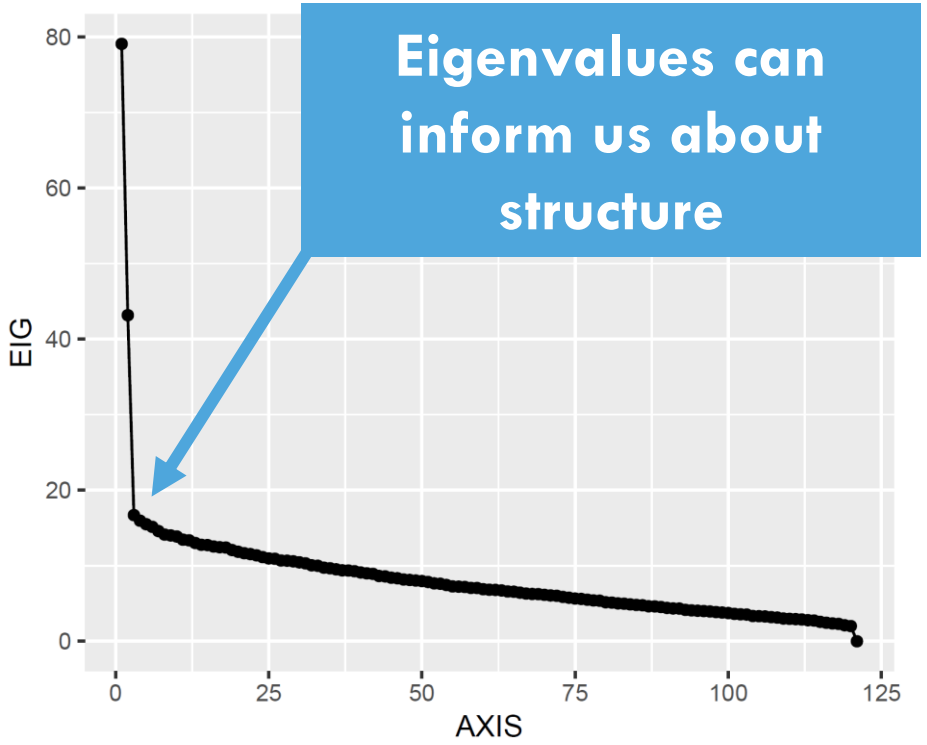
Principal components and structure

- Multilocus genotype data is multivariate
- PCA can be used to understand structure (Patterson et al. 2005)
- **<u>Goal:</u>** Obtain (orthogonal) linear axes that maximise variance across individuals using original variables

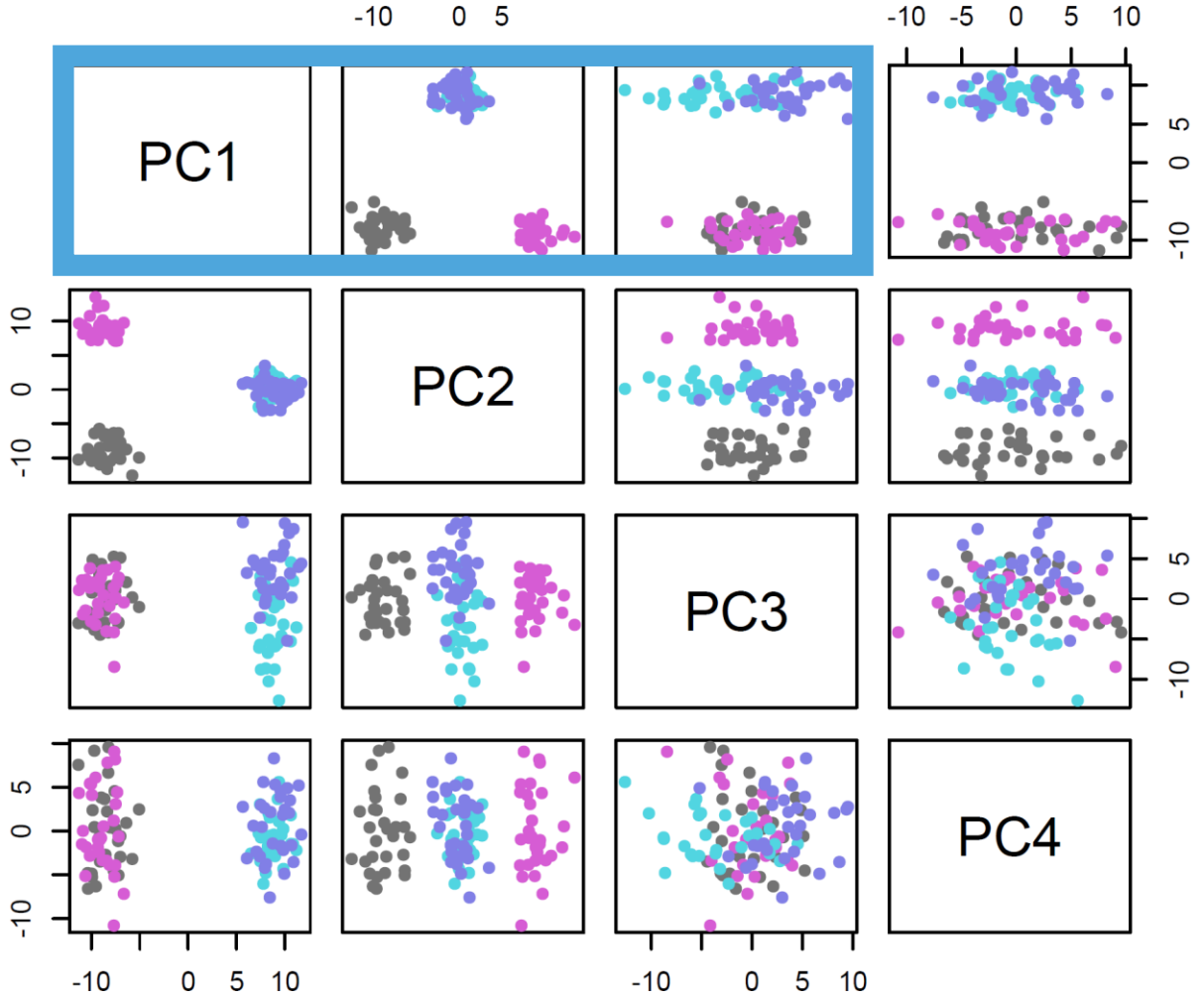




Principal components and structure



BUT **MULTIPLE** EIGENVECTORS ARE NEEDED TO CHARACTERISE THE MAIN AXES OF POPULATION STRUCTURE!



Principal components and structure

- PCA is an **undirected** analysis
- PCA allows **dimension** reduction and is **"demography free"**
- PCA can be an useful alternative for measuring **inter-individual genetic distance** (Shirk et al. 2017)
- PCA will capture, and allow us to visualise, structure but the explained variance is **maximised across individuals**

What about a directed approach?

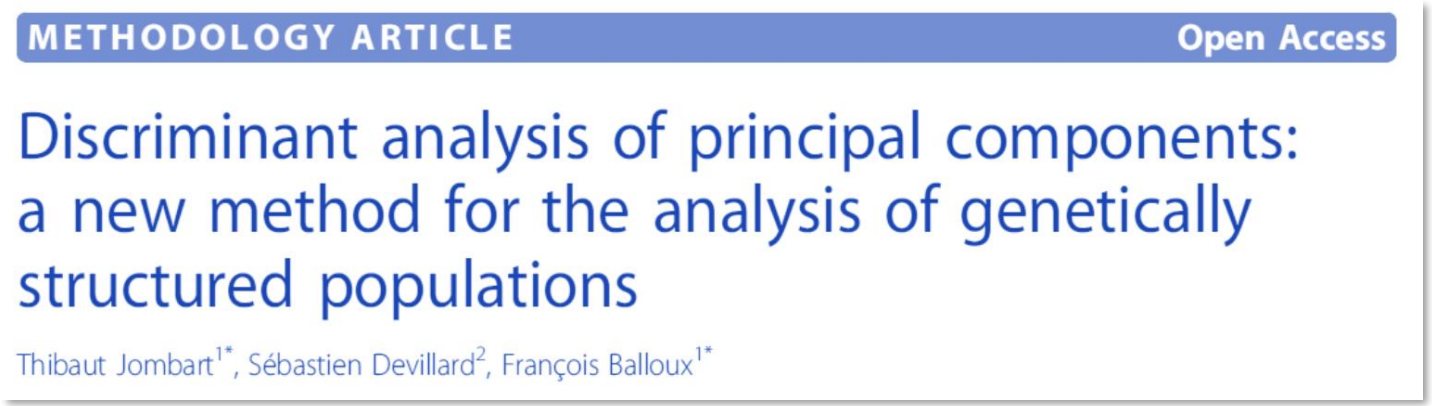
We want an analysis that is **hypothesis driven** and allows us to visualise axes of variation that **maximise differences between groups**.

Discriminant functions analysis (DFA)

- Also known as LDA (linear discriminant analysis)
- **Goal:** identify (orthogonal) linear combinations of original variables that maximise variation explained between groups
- Typically cannot use on genomic data because number samples << number loci
- PCA helps reduce dimensionality

Discriminant analysis of PCs

- **PCA → DFA = DAPC,** a discriminant analysis of principal components (Jombart et al. 2010)



- **It essentially amounts to an (M)ANOVA:**
  Population membership = Some combination of PC axes

adegenet and DAPC

- DAPC is implemented in the *adegenet* package

- Questions we can address:

    1. What are the relationships among populations?
    2. What are the populations?
    3. Which population(s) is an individual most likely to come from?

DAPC in practise

- Start with a wide table of genotypes, **individuals** (121) in rows, **loci** in columns (500), from one of four **populations**

- Genotypes are SNPs, coded as 0 = ref allele, and 1 = alt allele

- In this example, I have opted to use the `data.table` class instead of a `matrix`

```
> head(snpMat[, 1:5])
    POP    SAMPLE Chrom_1004_1 Chrom_1008_2 Chrom_1011_1
1: Pop1 Ind1.1040          0/0          0/0          1/1
2: Pop1 Ind1.1116          0/0          1/0          0/0
3: Pop1  Ind1.115          0/0          0/1          0/0
4: Pop1 Ind1.1153          0/0          1/0          0/0
5: Pop1 Ind1.1161          0/0          0/1          1/0
6: Pop1 Ind1.1184          0/0          1/1          1/0
> dim(snpMat[, !c('POP', 'SAMPLE')])
[1] 121 500
> class(snpMat)
[1] "data.table" "data.frame"
```

DAPC in practise

- *Adegenet* works with `genind` objects ("genotypes of individuals")
- Create this with the function `df2genind()`
- I like to think of `genind` objects simply as lists, which you use `@` to access indexes of the list from

```
> snpGenind <- df2genind(snpMat[, !c('POP', 'SAMPLE')]
+                        , sep='/'
+                        , ind.names=snpMat$SAMPLE)
> snpGenind
/// GENIND OBJECT /////////

 // 121 individuals; 500 loci; 1,000 alleles; size: 726 Kb

 // Basic content
   @tab:  121 x 1000 matrix of allele counts
   @loc.n.all: number of alleles per locus (range: 2-2)
   @loc.fac: locus factor for the 1000 columns of @tab
   @all.names: list of allele names for each locus
   @ploidy: ploidy of each individual  (range: 2-2)
   @type:  codom
   @call: df2genind(X = snpMat[, !c("POP", "SAMPLE")], sep = "/
", ind.names = snpMat$SAMPLE)

 // Optional content
   - empty -
```

DAPC in practise

- Individual genotypes are stored in the `@tab` of `genind` objects
- Genotypes are coded as dummy variables: each allele per locus is given its own column, with counts recorded

```
> snpGenind@tab[1:4, 1:4]
          Chrom_1004_1.0 Chrom_1004_1.1 Chrom_1008_2.0 Chrom_1008_2.1
Ind1.1040              2              0              2              0
Ind1.1116              2              0              1              1
Ind1.115               2              0              1              1
Ind1.1153              2              0              1              1
```

Ref Alt Ref Alt

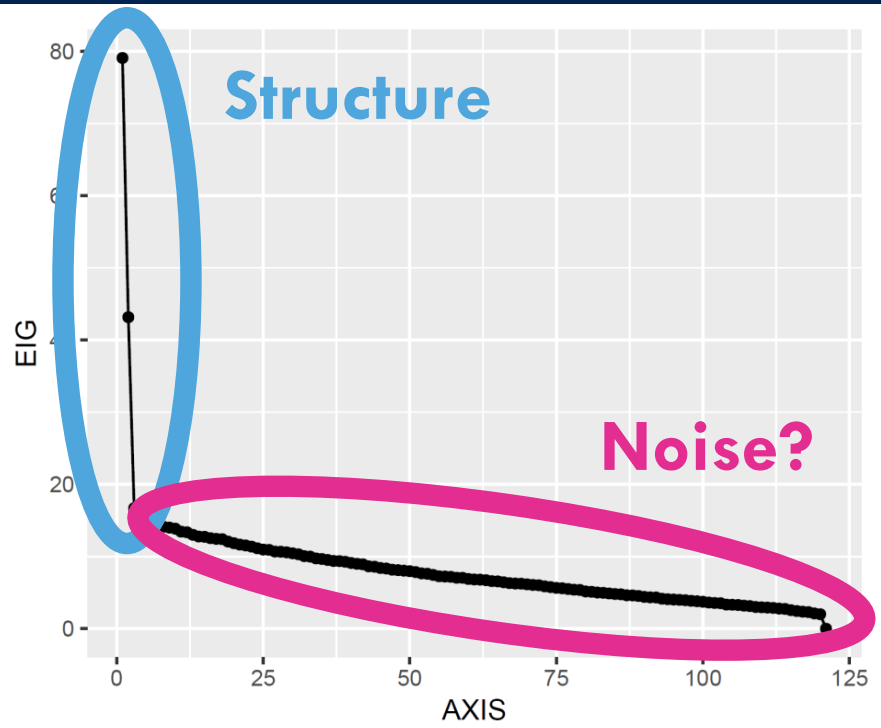
DAPC in practise

- Conduct PCA on allele counts to get a feel for data structure

- Remember: number of samples = 121 (<< loci), so only 121 PC axes

- Even though only 50% data is captured on first 32 PCs, main components of structure seem to lie on PCs 1–3

```
> # Conduct a PCA from the @tab table of allele counts
> pca <- prcomp(snpGenind@tab, center=TRUE, scale=TRUE)
>
> # Which PC explains up to 50% of the variance?
> var50 <- which(summary(pca)$importance[3,] <= 0.5)
> length(var50)
[1] 32
```



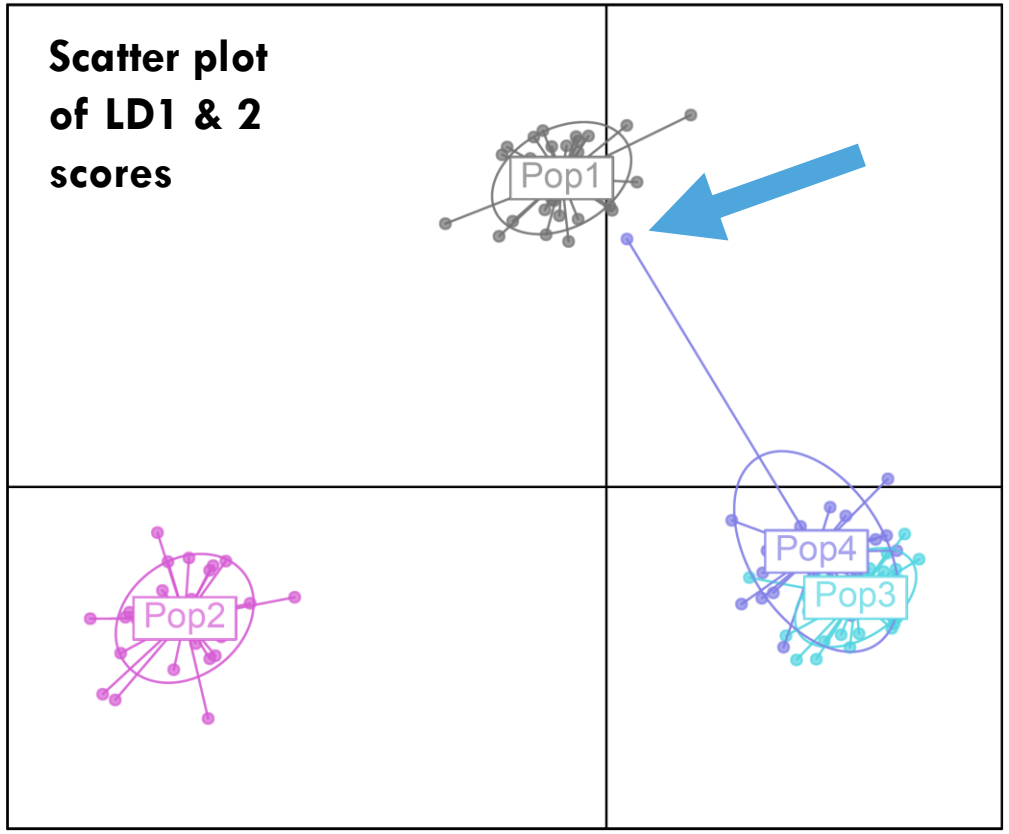
DAPC in practise

```
> # Conduct DAPC
> daPop <- dapc(snpGenind
+               , pop=snpMat$POP
+               , n.pca=length(var50)
+               , n.da=2)
>
> # PCA scores per individual
> daPop$tab[1:3, 1:3]
            PCA-pc.1  PCA-pc.2   PCA-pc.3
Ind1.1040 -3.162061 -3.108966 -0.1921109
Ind1.1116 -2.407914 -2.887011  0.4693019
Ind1.115  -1.479234 -3.382545 -0.0635438
>
> # Eigenvalues for each LD axis
> daPop$eig
[1] 1244.79587  562.61725   73.42052
>
> # Percent variance explained per LD axis
> daPop$eig/sum(daPop$eig) * 100
[1] 66.183199 29.913185  3.903616
```

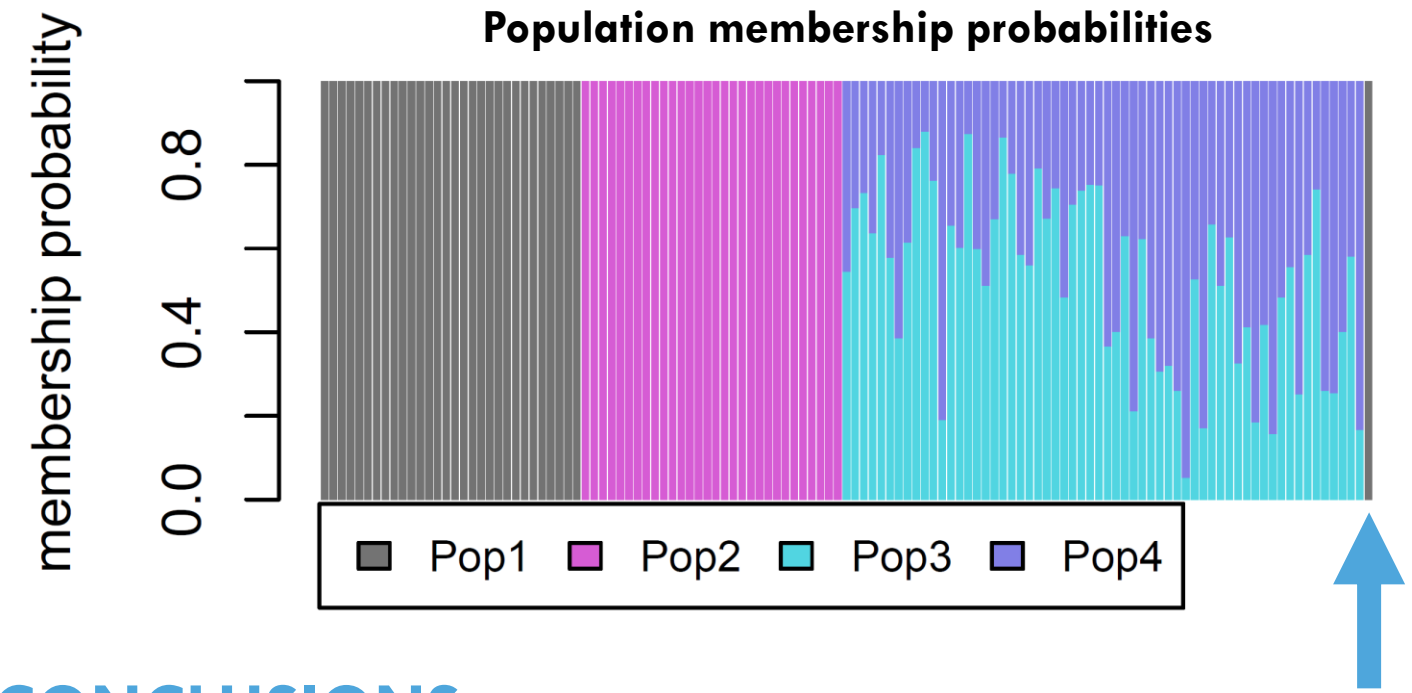
- Use the `dapc()` function to perform the DAPC
- In my `dapc()` call, I preassigned population identity (`pop=`), the number of PCs to use (`n.pca=`) and the number of discriminant axes to keep (`n.da=`)
- You can access the PC scores and the DFA eigenvalues (variance explained by each LD axis)

DAPC in practise

```
> scatter(daPop, col=popColsDT$COL, scree.da=FALSE)
```




```
> compoplot(daPop, col.pal=popColsDT$COL)
```

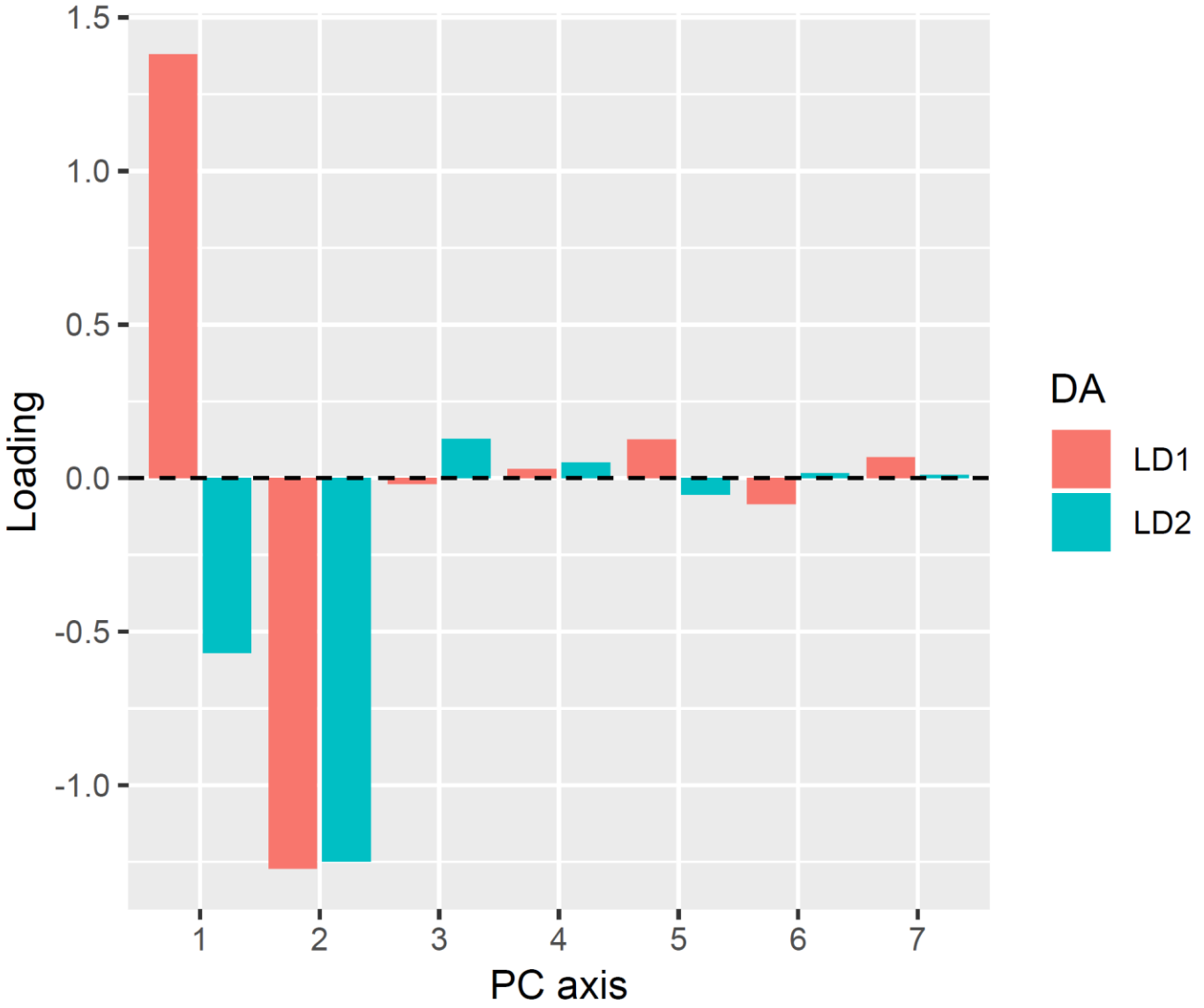


CONCLUSIONS:

1. Pops 3 & 4 are very similar, 1 & 2 are divergent
2. There is a migrant from Pop 1 into Pop 4

DAPC in practise

- Pulling out the loadings can tell you how much each PC axis contributes towards variation on each LD axis
- Illustrates that multiple PC axes can contribute toward single LD axes
- Illustrates that single PC axes can have different directions of effect

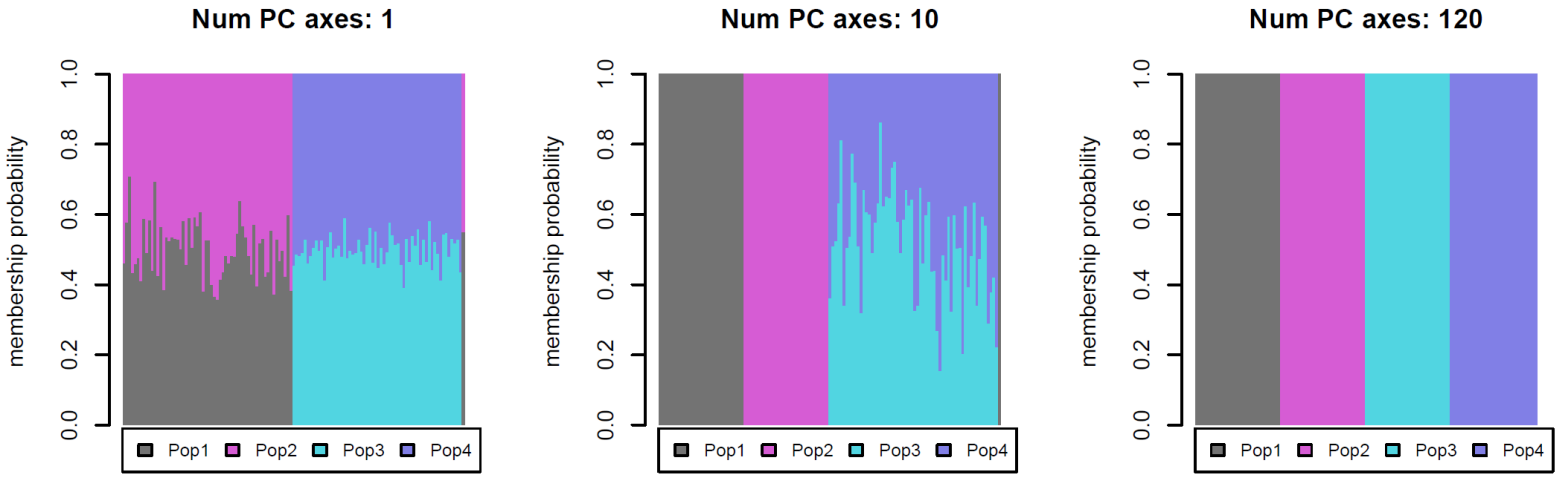


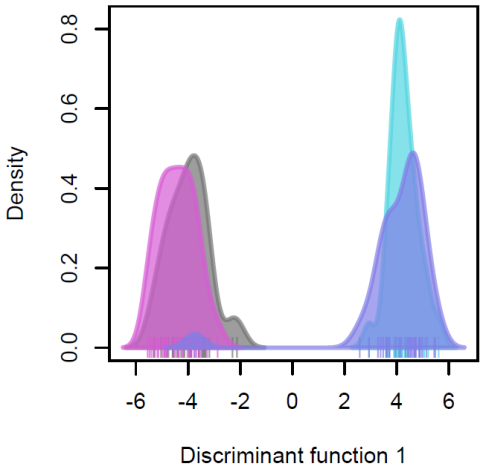
DAPC considerations

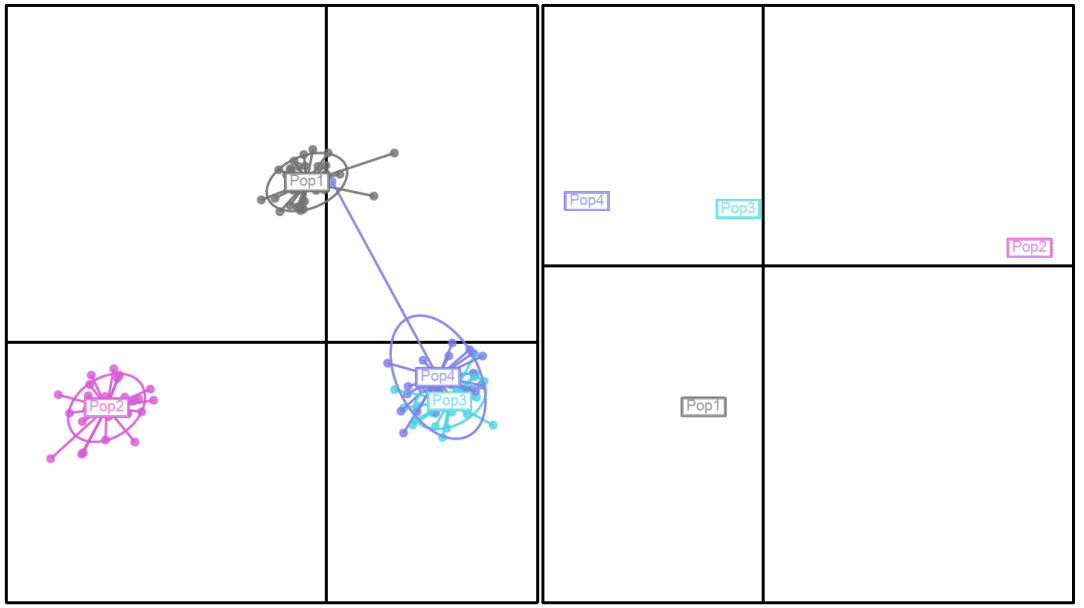
Observation	Consideration
Main components of structure lie on PCs 1–3, but are the rest noise?	How many PCs to use without over(/under) parameterising the DFA?
Genetic clusters might be < sampled “populations”	Redefine DFA model (in terms of clusters)?
There is a suspected migrant individual	Redefine the DFA model (in terms of cluster membership)?

DAPC: Number of PC axes

- We want enough information to discriminate populations, but we don't want to overparameterise
- Too little = cannot distinguish populations
- Too much = too much explanatory power (overfit)
- Overfit models cannot be used in a predictive capacity (just explain current dataset)







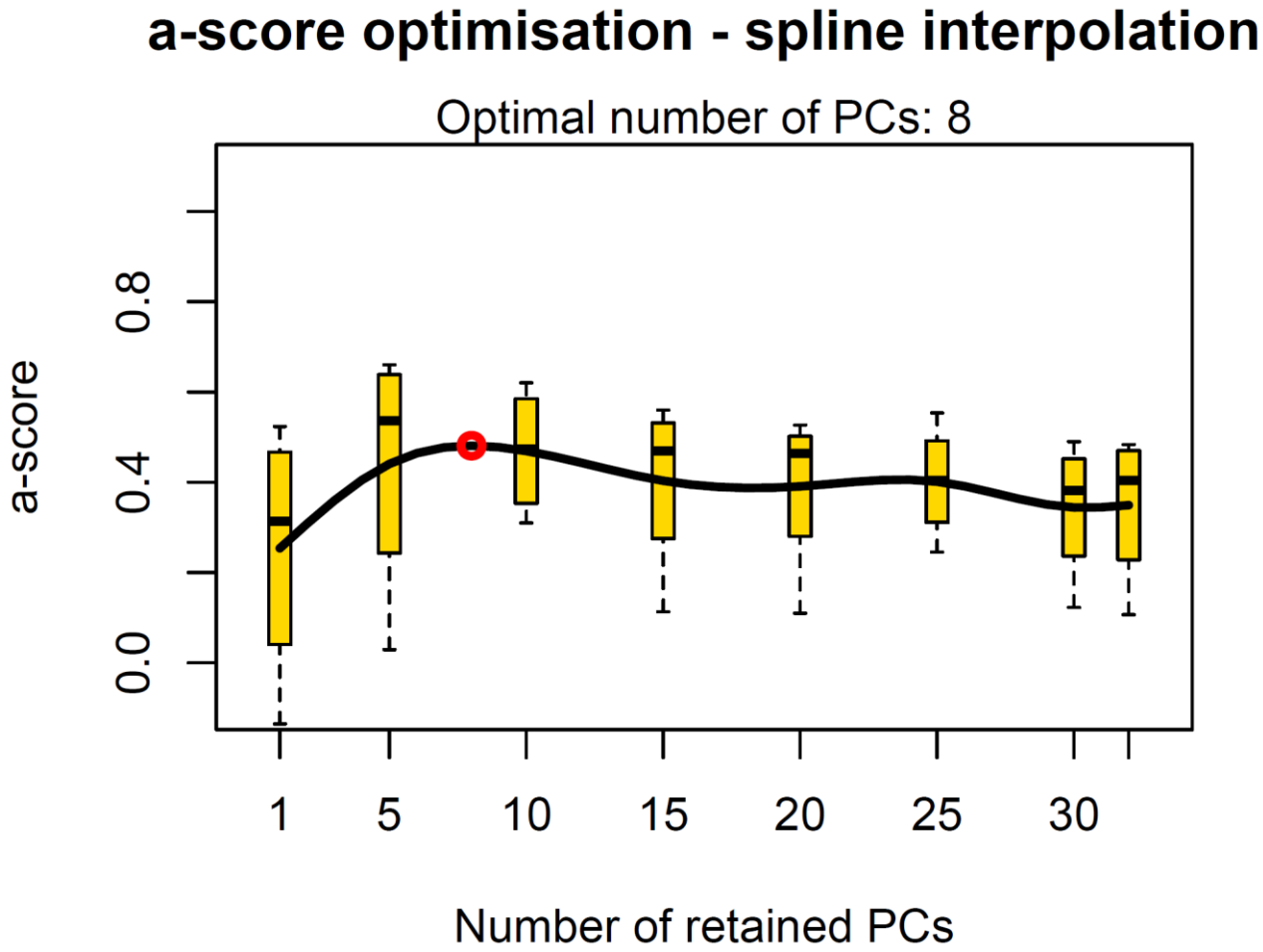
DAPC: Number of PC axes

- We can use the *a*-score to determine an optimal informative number of PCs to use

- The `optim.a.score()` function takes a `dapc` object produced from the `dapc()` function; it tests up to the maximum number of PCs used to construct the model

- **Rough interpretation:** The number of PCs to use that maximise the ability to assign individuals to their specified populations, penalised by the number of PCs

```
> class(daPop)
[1] "dapc"
> optim.a.score(daPop)
```

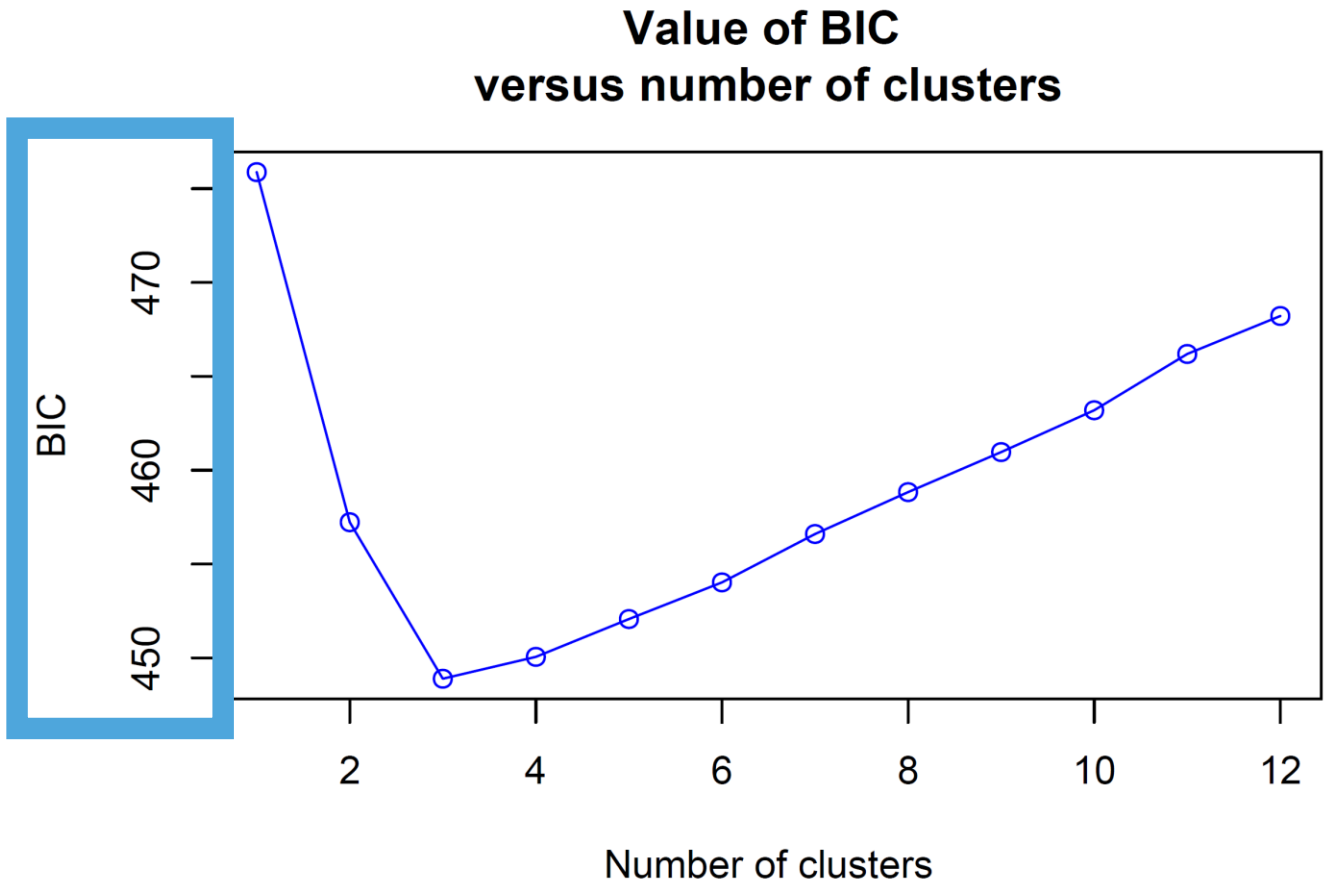


DAPC: Number of genetic clusters

- Genetic populations might ≠ physically sampled populations
- The `find.clusters()` function takes a `genind` object and reports a likelihood (BIC) for different numbers of clusters
- Lower BIC values indicate more likely clusterings

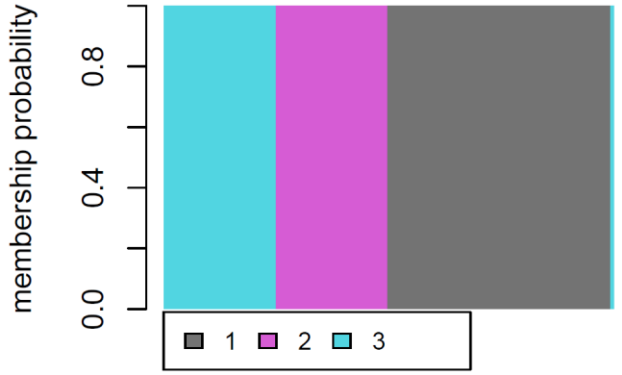
PAY ATTENTION TO SCALE: Curve can be misleading; consider difference in BIC among hypothesised clusters

```
> find.clusters(snpGenind, n.pca=length(var50))
```

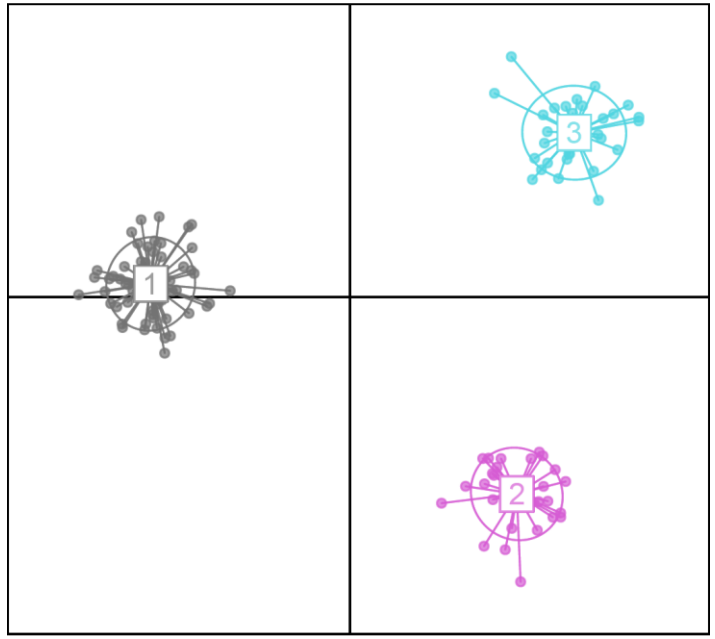


DAPC: Number of genetic clusters

The new model (based on genetic clusters) has grouped all individuals sampled from Pops 3 & 4 (except the migrant) into a single genetic cluster




```
> # Look for K=3 clusters, using PCs describing 50% variance (32)
> k3 <- find.clusters(snpGenind, n.pca=length(var50), n.clust=3, n.da=2)
>
> # Conduct the DAPC using 1 of K=3 clusters as pop
> daK3 <- dapc(snpGenind, pop=k3$grp, n.pca=7, n.da=2)
>
> # Plot the genotype composition and LD scatter
> compoplot(daK3, col.pal=popColsDT$COL[1:3])
> scatter(daK3, col=popColsDT$COL[1:3], scree.da=FALSE)
```



DAPC wrap up

☺

Fast

Demography-free

Membership probabilities

Between group variance

Predictive modelling

☹

Clustering can be arbitrary

There are always axes that can describe groups

May require optimisation

F_{ST} as an index of structure

F_{ST} is a measure of how allele frequencies differ among subpopulations

- Variance in allele frequency among populations relative to the mean (Wright 1969)

$$F_{ST} = \frac{\sigma_p^2}{\bar{p}(1-\bar{p})}$$

- Ratio of heterozygosity within versus across all subpopulations analysed (Nei 1987)

$$F_{ST} = 1 - \frac{H_S}{H_T}$$

- Variance among subpopulations and individuals in subpopulations (Weir & Cockerham 1984)

$$\theta = \frac{MSP - MSG}{MSP + MSG(n_C - 1)}$$

F_{ST} in practise

F_{ST} using a `genind` object

- You can pass `genind` objects to the `fstat()` function from the `hierfstat` package…

```
> hierfstat::fstat(snpGenind, pop=snpMat$POP)
              pop          Ind
Total 0.08352399  0.082559720
pop   0.00000000 -0.001052149
```

- This is incredibly slow on large datasets and bootstrapping for significance takes forever.

- There are much faster ways…

F_{ST} in practise

genlight objects and F_{ST} using the package *StAMPP*

- The package *StAMPP* can calculate F_{ST} (Weir & Cockerham's θ) from genomic data and ties into *adegenet* pipeline
- However, it uses *adegenet*'s `genlight` object (not `genind`)
- `genlight` objects are more efficient versions of `genind` objects, used for storing biallelic SNP data

F_{ST} in practise

genlight objects and F_{ST} using the package *StAMPP*

```
> snpMat[1:5, 1:4]
    POP    SAMPLE Chrom_1004_1 Chrom_1008_2
1: Pop1 Ind1.1040           0/0           0/0
2: Pop1 Ind1.1116           0/0           1/0
3: Pop1  Ind1.115           0/0           0/1
4: Pop1 Ind1.1153           0/0           1/0
5: Pop1 Ind1.1161           0/0           0/1
```

Genotype matrix

```
genoList <- lapply(snpMat$SAMPLE, function(samp){
  loci <- unlist(snpMat[SAMPLE==samp, !c('POP', 'SAMPLE')])
  return(
    unlist(lapply(strsplit(loci, '/', fixed=TRUE)
                  , function(allele){2-sum(as.integer(allele))
    }))
  )
})
```

Code to rearrange data into a list

```
> genoList[[1]][1:3]; genoList[[2]][1:3]
Chrom_1004_1 Chrom_1008_2 Chrom_1011_1
           2            2            0
Chrom_1004_1 Chrom_1008_2 Chrom_1011_1
           2            1            2
```

New list: indexed elements = samples with allele counts per locus

F_{ST} in practise

genlight objects and F_{ST} using the package *StAMPP*

```
> # Initiate a genlight object
> snpGenlight <- new('genlight', genoList
+                    , ind.names=snpMat$SAMPLE
+                    , pop=snpMat$POP
+                    , ploidy=2)
> snpGenlight
 /// GENLIGHT OBJECT /////////

 // 121 genotypes,  500 binary SNPs, size: 189.2 Kb
 0 (0 %) missing data

 // Basic content
   @gen: list of 121 SNPbin
   @ploidy: ploidy of each individual  (range: 2-2)

 // Optional content
   @ind.names:  121 individual labels
   @pop: population of each individual (group size range: 30-31)
   @other: a list containing: elements without names
```

F_{ST} in practise

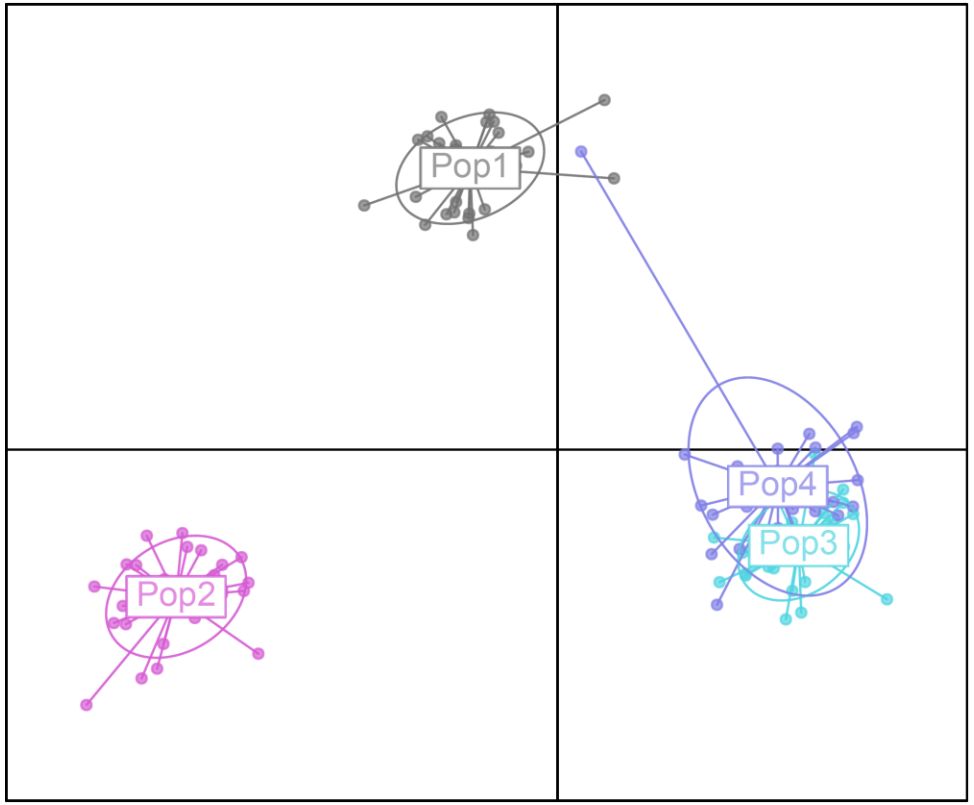
genlight objects and F_{ST} using the package *StAMPP*

```
> pairFst <- stamppFst(snpGenlight
+                      , nboots=1000
+                      , nclusters=10)
> # Get FST values
> pairFst$Fsts
           Pop1       Pop2        Pop3 Pop4
Pop1         NA         NA          NA   NA
Pop2 0.07402165         NA          NA   NA
Pop3 0.10279032 0.1076404          NA   NA
Pop4 0.09457817 0.1048576 0.01072514   NA
> # Get FST significance
> pairFst$Pvalues
     Pop1 Pop2 Pop3 Pop4
Pop1   NA   NA   NA   NA
Pop2    0   NA   NA   NA
Pop3    0    0   NA   NA
Pop4    0    0    0   NA
```

F_{ST} in practise

genlight **objects can also be used in DAPC**

```
> # Can do regular DAPC on genlight objects
> daGL <- dapc(snpGenlight, n.pca=7, n.da=2)
> scatter(daGL, col=popColsDT$COL, scree.da=FALSE)
```



Population colour objects

```
> # Make a colour reference table
> popColsDT <- data.table(POP=paste('Pop', 1:4, sep='')
+                         , COL=c('#737373', '#d65cd4', '#51d5e1', '#817fe6'))
>
> popColsDT
    POP     COL
1: Pop1 #737373
2: Pop2 #d65cd4
3: Pop3 #51d5e1
4: Pop4 #817fe6
>
> # Make a colour vector
> popColsvec <- popColsDT$COL[match(snpMat$POP, popColsDT$POP)]
>
> length(popColsvec); popColsvec[1:5]
[1] 121
[1] "#737373" "#737373" "#737373" "#737373" "#737373"
```

Relevant papers

1. Evan et al. (2005) Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. *Mol. Ecol.*
2. Jombart et al. (2010) Discriminant analysis of principal components A new method for the analysis of genetically structured populations. *BMC Genetics.*
3. Nei (1987) Molecular Evolutionary Genetics. Amsterdam: North-Holland.
4. Patterson et al. (2006) Population structure and eigenanalysis. *PLOS Genetics.*
5. Pritchard et al. (2000) Inference of Population Structure Using Multilocus Genotype Data. *Genetics.*
6. Raj et al. (2014) fastSTRUCTURE: Variational Inference of Population Structure in Large SNP Data Sets. *Genetics.*
7. Shirk et al. (2017) A comparison of individual-based genetic distance metrics for landscape genetics. *Mol. Ecol. Res.*
8. Weir & Cockerham (1984) Estimating *F*-statistics for the analysis of population structure. *Evolution.*
9. Wright (1969) Evolution and the Genetics of Populations, v2. The Theory of Gene Frequencies. Chicago, IL: Uni. Chicago Press.

Thibaut Jombart's github is a great source of information and tutorials on *adegenet*: https://github.com/thibautjombart/adegenet