Population Genetics of Polyploids

Ede-Wageningen, Netherlands, September 10-13, 2026

Teachers and organizers:

Violette Doublet Charles University Prague CZ, doublet.violette@gmail.com

Jörn Gerchen Charles University Prague CZ, gerchenj@natur.cuni.cz

Filip Kolář Charles University Prague CZ, fillip.kolar@natur.cuni.cz

Patrick Meirmans University of Amsterdam NL, p.g.meirmans@uva.nl

Alison Scott Max Planck Institute for Plant breeding Research Cologne DE, ascott@mpipz.mpg.de

Block A
Introduction to autopolyploid population genetics theory

A1: HWE and genetic diversity

Hardy-Weinberg equilibrium (HWE)

The Hardy-Weinberg principle is one of the cornerstones of population genetics and is something you have hopefully heard of. It deals with the expected genotype frequencies under random mating, in a single infinitely large population (without selection and other disturbing factors). However, it was originally formulated for diploids and works somewhat differently for polyploids.

Imagine a locus with two alleles, A and B, that have the frequencies p=0.8 and q=0.2, respectively.

1. *To fresh up your memory: for a diploid population what are the expected genotype frequencies under HW-equilibrium?*

Now imagine a locus with the same properties (biallelic, p=0.8 and q=0.2), but now in a population of autotetraploids (so assuming tetrasomic inheritance). Let's address the same problem as above, but in somewhat smaller steps.

2. *Which genotypes can be formed?*

The easiest way to derive the genotype frequencies for an autotetraploid is to first look at the gametes produced by this population.

3. *Which diploid gamete genotypes (allelic combinations) can be produced?*

Assume that the combination of alleles to form diploid gametes is completely random.

4. *Given the population allele frequencies above, what will be the frequencies of these gametes?*

Assume that the combination of gametes to form tetraploid zygotes is completely random.

5. *What will be the frequency of each genotype in the progeny? Check if the frequencies that you calculated sum up to one. HINT: make a cross table of all pairs of gametes.*

In diploids, HWE is reached in a single generation of random mating. Let's evaluate if this also holds for autopolyploids. Imagine that two previously separated populations of tetraploids have fused and the new population consists for 50% of genotype *AAAA* and for 50% of genotype *BBBB*.

6. *Under random mating, which gametes are produced by this population and at what frequencies?*

7. *Assume these gametes unite at random – so undergo a single generation of random mating. How can you immediately see that this population is NOT in Hardy Weinberg equilibrium?*

The inbreeding coefficient F_{IS}

To quantify deviation from HWE it is common to calculate the summary statistic F_{IS}, which is calculated by comparing the observed heterozygosity (H_O) with the expected heterozygosity (H_S) under HWE. The heterozygosity is calculated as the frequency of heterozygotes in the population. Note that, although expected heterozygosity is often abbreviated as H_E, we prefer to use H_S here, which stands for '*H* in the Subpopulation'. Later, this will allow us to distinguish it from H_T, which stands for '*H* in the Total population', i.e., a collection of multiple subpopulations. For now, we only deal with a single population, for which the inbreeding coefficient can be calculated as:

$$F_{IS} = (H_S - H_O) / H_S$$

The value of F_{IS} ranges from -1, indicating a complete lack of homozygotes, to 1, indicating a complete lack of heterozygotes. A value of 0 means that there are exactly as many heterozygotes as expected under HWE.

8. *For a diploid population with genotype frequencies AA=0.3, AB=0.2, and BB=0.5, what are the values of H_O, H_S and F_{IS}?*

When there are multiple alleles, the number of heterozygous genotypes increases quickly with the number of alleles. Therefore, it is easier to look at the homozygotes for calculating H_S. So, for a diploid, H_S can be calculated by summing over all the different alleles as:

$$H_S = 1 - \sum p_i^2$$

where p_i is the frequency of allele i.

9. *How could you then calculate the expected heterozygosity for a single multi-allelic locus in a sample of tetraploids?*

If you got the "correct" answer, you probably also realize that – for the same allele frequencies p – H_S is inherently higher for tetraploids than for diploids, which is obviously nonsense when you want to use it as an indicator of genetic diversity. This is an important realization, and we will come back to it later, but let us first consider the effect of inbreeding (selfing) on heterozygosity.

10. *When a diploid of genotype AB or a tetraploid of genotype ABCD self-fertilize, what are the respective offspring genotypes and their relative frequencies? (Hint: For tetraploids, first write down all the possible gametes that can be formed, and their frequencies)*

11. *By which ratio does the observed heterozygosity H_O decrease per generation of selfing for the diploid heterozygote (AB) in the previous question?*

You might wonder why we do not ask you (at this point) to calculate the same ratio for the tetraploid ABCD. The reason is that we first need to explain how to calculate genetic diversity, more specifically H_O and H_S, for tetraploids.

Genetic diversity

H_S is generally used as a measure of genetic diversity (then called the gene diversity, Nei 1987), which allows comparison of genetic diversity among populations or species.

12. *Take a locus with four alleles with frequencies 0.55, 0.2, 0.15 and 0.1. Using the above equations (see question 8 & 9, what would be the expected heterozygosity for diploids and for tetraploids?*

13. *So, are polyploids more diverse than diploids despite the same allele frequencies?*

You (supposedly) saw that calculating H_S for polyploids by only looking at the expected homozygote frequencies will make comparisons among ploidy levels impossible. To solve this issue, it is necessary to now switch off your biological common sense.

Generally, it is agreed upon to calculate the gene diversity for polyploids in exactly the same way as for diploids, namely by squaring and summing the allele frequencies (rather than using the power-term applicable to the ploidy level, as in question 9):

$$H_S = 1 - \sum p_i^2$$

This technically means that the gene diversity can no longer be called the "expected heterozygosity", but it is nevertheless common to still indicate it with H_S.

So to recap: no matter what the ploidy level is, the square term in the equation is **never** replaced with a higher (or lower) power.

14. *Take a locus with four alleles with frequencies 0.39, 0.28, 0.27 and 0.06. What would be the gene diversity for diploids and for tetraploids?*

15. *Write down some possible tetraploid genotypes for a locus with alleles A, B, C, and D (being exhaustive is not necessary). Are the heterozygotes all equally heterozygous?*

If we want to compare H_S (expected H) with H_O (observed H) in a tetraploid, for example for calculating F_{IS}, we have to use a trick (similar to the one we used for H_S) to calculate H_O. H_O for a tetraploid can be calculated as the so-called "gametic heterozygosity". To calculate this for a given tetraploid genotype, randomly combine its alleles into diploid gametes and assess the frequency of heterozygous gametes. Take, for example, genotype *AABB*. There is a probability of 0.5 that the first drawn allele is an *A*. To obtain a heterozygous gamete, the second allele drawn should be a *B* (which has a probability of 2/3). Alternatively, a heterozygous gamete can be formed by first drawing allele *B* and then *A*, which has the same probability (for this genotype).

16. *What is then the combined probability of drawing a heterozygote diploid gamete for genotype AABB? In other words, what is the gametic heterozygosity for this genotype?*

17. *Which would you (intuitively) think has a higher heterozygosity: AAAB or AABB?*

18. *To check your intuition, calculate the gametic heterozygosity for genotypes AAAA, AAAB, CCDD, AABC, and ABCD? It helps to enumerate all the possible ways to draw gametes from a genotype.*

Once we have the gametic heterozygosity for each genotype, we can average over all genotypes in the population to calculate H_O for a sample of tetraploids. The same concept of gametic heterozygosity can be applied to other ploidy levels. Note that for this we would – perhaps counterintuitively - still use conceptual diploid gametes, even for ploidy levels that do not actually produce diploid gametes (just like we never replace the square term in the equation for H_S for ploidies other than diploid).

19. *What is the gametic heterozygosity for the octoploid genotype AAAAAAAB?*

20. *What would be the reason why the gametic heterozygosity for an octoploid is calculated assuming diploid gametes and not tetraploid gametes?*

21. *So, now you should be ready to calculate by which ratio the observed heterozygosity H_O decreases per generation of selfing for a fully heterozygous tetraploid (ABCD). Compare your answer to the ratio you calculated for diploid AB in question 11.*

In summary, calculating gametic heterozygosity for polyploids (as if the species were diploid) allows comparing H_O and H_S for any ploidy level. In other words, it allows us to calculate a more meaningful estimate of F_{IS}. Doing this by hand is a bit too tedious for this practical, but luckily, there is software for this.

A2: Doing population genetic simulations in R

For this workshop, we will focus on the analysis of genetic data, and R can also be very useful for that. Simulations form an essential part of genetic data analysis, as they allow you to understand the behaviour of certain statistical methods under different scenarios. After all, you cannot check whether a method gives accurate results if you do not know what the true situation is to begin with. I therefore recommend testing any method that you have not tried before using simulated data, to get an idea of when it works and when not.

Though simulations sound very scary to many people, you actually do not have to be a mathematician nor a computer scientist to do them. A few lines of code are sufficient to do (simple) population genetic simulations. For this, you will make heavy use of the for-loops that were introduced above.

A simple genetic simulation

Let's start with a simple simulation in R of genetic drift in a small population of 10 haploid individuals. For this, you will use the sample() function to take a sample of a specified size from a population of 5 blue individuals and 5 red individuals. In this population, we assume colour is determined by a single Mendelian locus.

Open R-studio and create a new empty script. Now type in the following command to create a vector representing this population:

```
population = c(rep("blue", 5), rep("red", 5))
```

During six generations *g*, we will now replace these ten individuals with randomly selected offspring, so without any selection benefit for either of the colours. This amounts to random sampling, with replacement, of ten elements from the population vector.

```
for(g in 1:6){
    population = sample(population, 10, replace=TRUE)
}
```

22. *Are six generations enough to reach fixation in this population? And if you repeat the simulation several times?*

For such a small population, this is a useful approach, but for very large populations the population vector becomes so large that the sample function gets slow. From your first-year statistic course you may recall that when there are only two options to choose from (red and blue), sampling with replacement follows the binomial distribution. R has several functions regarding the binomial distribution, including a function to draw random numbers based on the binomial, which has the following syntax: rbinom(n, size, prob). Here, n is the number of random numbers that is returned, which is 1 for this purpose; size is the number of draws, for us that is the population size; prob is the probability of success for each draw, for us this is the frequency of the "blue" allele. The function returns an integer which is the number of successes after all size draws, where success means getting a blue allele. We can now recreate our previous simulation with the rbinom function, using N for the population size, and p for the allele frequency.

```
N = 10
p = 0.5
for(g in 1:6){
     p = rbinom(1, N, p)/N
}
```

23. Run this script, is the outcome the same as above?

Of course, it would be nice to see how the allele frequency fluctuates over time as a result of genetic drift. For this, we can store the allele frequency for each generation in a vector and then plot all values after the loop has finished.

```
generations = 1000
N = 25
p = 0.5
values = rep(NA, generations)

for(g in 1:generations){
     p = rbinom(1, N, p)/N
     values[g] = p
}
```

```
plot(1:generations, values, type="l", col="firebrick3", ylim=c(0,1),
     xlab="Generation", ylab="Allele frequency", font.lab=2)
```

24. Run this script with population sizes of different orders of magnitude. What happens?

Optional exercise for those who are fast: now use your knowledge and creativity to produce a plot with multiple simulations in one. Tip: use the function lines with the argument add=TRUE

Adding mutation

For this we will not use the allele frequency of the current generation (as we have been doing in the previous simulations) as probability for the random draw from the binomial distribution. Instead, we will use the expected allele frequency after mutation. For example, when a population has p=1 and u=0.001, the expected frequency after mutation becomes p_exp=0.999. However, we also need to include reverse mutation so when p=0, p_exp=0.001. When the allele frequency is exactly equal to 0.5, we don't expect mutation to have any effect, so p_exp=p. We can code this as follows:

```
> p_exp = p - u*p + u*(1-p)
```

and then add this to the loop we used above, so that we get:

```
generations = 5000
N = 100
p = 0.5
u = 0.001
values = rep(NA, generations)

for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p = rbinom(1, N, p_exp)/N
     values[g] = p
}
plot(1:generations, values, type="l", col="firebrick3", ylim=c(0,1),
     xlab="Generation", ylab="Allele frequency", font.lab=2)
```

25. Does the population still get fixed for this locus?

26. Run the model with increasingly large population sizes, what happens?

Randomly mating diploids

Our simulations above were technically simulating a clonally reproducing haploid. Below is the same simulation, but now for randomly mating diploids:

```
generations = 5000
N = 100
p = 0.5
u = 0.001
values = rep(NA, generations)

for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p = rbinom(1, 2*N, p_exp)/(2*N)
     values[g] = p
}
plot(1:generations, values, type="l", col="firebrick3", ylim=c(0,1),
     xlab="Generation", ylab="Allele frequency", font.lab=2)
```

27. Can you spot the change relative to the haploid script? Why is the difference so slight?

28. Can you now make the population tetraploid?

Multiple loci

When we want to simulate multiple independent loci, we can make *p* a vector and draw a separate random number for each locus, based on a vector of expected frequencies:

```
generations = 1000
loci = 100
N = 250
p = rep(0.5, loci)
u = 0.0001
for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p = rbinom(loci, 2*N, p_exp)/(2*N)
}
```

We did not store the allele frequencies this time, as we anyway cannot plot them all. A simple way to keep track of the diversity within the population across all loci is to calculate the gene diversity, H_S (expected heterozygosity), which is defined for a single biallelic locus as: $H_S = 1 - p^2 - (1-p)^2$. For multiple loci, one can simply take the average across loci. For our vector of frequencies this is:

```
> Hs = mean( 1- p^2 - (1-p)^2 )
```

When we use this in the simulation script to store the value of H_S every generation (note the changed y-axis title in the ylab argument):

```
generations = 1000
loci = 100
N = 250
p = rep(0.5, loci)
u = 0.0001
values = rep(NA, generations)

for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p = rbinom(loci, 2*N, p_exp)/(2*N)
     values[g] = mean( 1- p^2 - (1-p)^2 )
}
plot(1:generations, values, type="l", col="firebrick3", ylim=c(0,1),
     xlab="Generation", ylab="Gene diversity (Hs)", font.lab=2)
```

Multiple populations with migration

The last extension to our simulation is to create multiple populations connected by migration. This follows the classic Island model of Sewall Wright. Just like above, we will first do the simulation itself, without storing any results, and then see how we can calculate some interesting statistics for it. Rather than specifying them one-by-one, we will now initialise the allele frequencies as a two-dimensional array, with the columns representing the populations and the rows the loci.

```
> numPops = 10
> p = array(0.5, c(loci, numPops))
```

Under the island model of migration, all populations are equally connected to each other, so migration is not constrained by any spatial arrangement of the populations. This can be imagined as all migrants from all populations being gathered into a big migrant cloud and then redistributed over all populations (so every migrant has a probability of returning to its original source population). Since all populations have the same size, it follows that the allele frequency of the migrant pool is the same as the overall frequency of the allele across all populations. We can thus simulate migration by lowering the allele frequency in a population, proportionally to the migration rate, and then adding the overall population frequency, proportionally to the migration rate:

```
> p_exp = p - migration*p + migration*rowMeans(p)
```

Since *p* is an array with the loci as rows, the overall allele frequencies at the loci can be obtained by calculating for every row the average over the different columns (i.e., the populations). The function rowMeans can do exactly that. The output of rowMeans is in principle a vector, but is automatically expanded in order to be able to add it to the array *p*. In contrast, the output of the rbinom function is always a vector, even if you feed it an array with probabilities, so we have to convert its output back into an array, which is done with the dim function.

The updated script now first takes mutation into account, and then migration (the order has little effect):

```
generations = 1000
loci = 10
pops = 4
migration = 0.001
N = 250
p = array(0.5, c(loci, pops))
u = 0.0001
```

```
for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p_exp2 = p_exp - migration*p_exp + migration*rowMeans(p_exp)
     p = rbinom(loci*pops, 2*N, p_exp2)/(2*N)
     dim(p) = c(loci, pops)
}
```

We now have a script for simulating allele frequencies under an island model of migration that includes drift, mutation and migration for multiple loci. Amazingly, the actual simulation only requires six lines of code!

The gene diversity can now be calculated similarly as above, averaging both across loci and across populations:

```
> Hs = mean(1 - rowMeans(p^2 + (1-p)^2))
```

Another interesting statistic is H_T, which is the gene diversity based on the overall allele frequencies:

```
> Ht = mean(1 - (rowMeans(p)^2 + rowMeans(1-p)^2))
```

Note that the difference between H_S and H_T lies in the placement of some brackets. From H_S and H_T, we can calculate F_{ST}, including a correction for the number of populations:

```
> Fst = pops*(Ht-Hs)/(pops*Ht-Hs)
```

The complete Island model script

We can now combine everything into a single script that will run a simulation under the island model, and plot the three most important summary statistics as a function of time. Again, note that the actual simulation only takes a couple of lines; everything else is just setting up the input and the output.

If you have time left, play around a bit with the script below, change some settings and look what effect they have on the values of the summary statistics.

```
generations = 1000
loci = 10
pops = 4
migration = 0.001
N = 250
p = array(0.5, c(loci, pops))
u = 0.0001

allHs = rep(NA, generations)
allHt = rep(NA, generations)
allGst = rep(NA, generations)

for(g in 1:generations){
     p_exp = p - u*p + u*(1-p)
     p_exp2 = p_exp - migration*p_exp + migration*rowMeans(p_exp)
     p = rbinom(loci*pops, 2*N, p_exp2)/(2*N)
     dim(p) = c(loci, pops)

     allHs[g] = mean(1 - rowMeans(p^2 + (1-p)^2))
     allHt[g] = mean(1 - (rowMeans(p)^2 + rowMeans(1-p)^2))
     allGst[g] = pops*(allHt[g]-allHs[g])/(pops*allHt[g]-allHs[g])
}

cols = c("olivedrab3", "firebrick3", "dodgerblue3")
matplot(1:generations,cbind(allHs, allHt, allGst),col=cols, type="l",
     ylim=c(0,1), xlab="Generation", ylab="Statistic value", font.lab=2)
legend("topright", c("Hs", "Ht", "Gst"), col=cols, lwd=1, lty=1:3)
```

Simulating genetic diversity

Now that you know the basics of doing population genetic simulations in R, it is time to put your new skills to the test. Later in this workshop, you will use similar simulations to look at the effect of polyploidy on the strength of population structure and the workings of selection. Now we will first have a look at how polyploidy affects the genetic diversity of a population.

For these simulations, you will use the R-script '*Heterozygosity.R*'. The script first establishes a single population consisting of a specified number of individuals, all of the same ploidy level. However, like you did above, the individuals are not modelled explicitly; there is just an array containing the allele frequencies at a given number of loci. Genetic drift is simulated by drawing random numbers from a multinomial distribution. The expectation for these random draws are based on the current allele frequencies, with a bit of mutation sprinkled on top. Every random draw represents a single generation of random mating.

29. *Run the script with the default settings and study the resulting graph. Describe what is happening here.*

30. *Now start with maximum diversity (equal initial frequencies for all alleles at a locus), what changes?*

31. *Change the ploidy level to several different values and run the script for each (again start with maximum diversity). Describe what happens to the equilibrium level of genetic diversity (for which we here take the average value over last 1000 generations).*

32. *Now run the model for a tetraploid population and write down the equilibrium value of H_S. Now set the ploidy level back to diploid. Which other parameter do you need to change to get –approximately– the same level of diversity as for the tetraploids?*

A3: Population differentiation

The fixation coefficient F_{ST}

Up to now, we have mostly discussed a single population, but most population genetic analyses actually focus on multiple populations. Analyzing the differentiation among populations allows us to make inferences on a range of topics such as migration, historical processes, conservation, and adaptation. In a way, looking at genetic differentiation amounts to looking at genetic diversity, but then how it is distributed within and among populations. To quantify the degree of population differentiation, the summary statistic F_{ST} is used –a close relative of F_{IS} that we used above. F_{ST} is calculated as a comparison of the expected heterozygosity within populations (H_S) and the expected heterozygosity of the total population (H_T):

$$F_{ST} = (H_T - H_S) / H_T$$

The values of F_{ST} range from 0, indicating no differentiation, to 1, indicating fixation in all populations: all populations only have a single allele left, but this is not always the same allele in all populations.

The simplest model of population structure is the island model, which is widely used in population genetics. Under the island model there is a set of populations that all have the same number of individuals (N). Mating within populations is completely random and the species are hermaphroditic annuals. Population connectivity is modelled as equal migration among all populations, meaning that there is no spatial structure. As above, there is also some mutation. Under the island model, the equilibrium value of F_{ST} depends on the balance of drift, mutation, and migration.

Here, we will perform some simulations of a set of populations under the island model. The first simulations we will look at are in the script '*Genetic Differentiation Fst*'. This script is a bit more complicated than the previous, so take your time to go through it. Start by looking at the function called `sum.stats` that is defined at the top, which will calculate the summary statistics.

In contrast to the previous simulations we did for the heterozygosity, we now use a locus with only two alleles. This means that the populations can be represented by a simple array with every row representing a population and every column a locus. The cell value then represents the number of copies of allele A in the population. The allele frequency can thus be obtained by dividing this value by the total number of chromosome copies in the population (the number of individuals times the ploidy level).

33. ***Bonus for R-gurus:*** *Find the spot where the populations are initialised. Do all populations have exactly the same allele frequencies? If not, why is there variation?*

The core mechanics of the model are very similar to that of the previous model: all stochasticity –in migration, mutation and drift– derives from drawing random numbers. Last time, we used a multinomial distribution since we had multi-allelic loci, but this time a binomial distribution will suffice since we have biallelic data now.

34. ***Bonus for R-gurus:*** *The spots in the code where mutation and migration are implemented should be easy to find, but can you also pinpoint where drift is implemented?*

35. *Run the script with the default settings and study the resulting graph. Describe what is happening here.*

36. *Change the ploidy level to several different values and run the script for each. Describe what happens to the equilibrium level of* F_{ST}.

The script '*Compare F-stats.R*' will create a plot of the value of F_{ST} as a function of the migration rate for different ploidy levels (based on theoretical expectations, which we will not go into here). Run the script and study the output.

37. *At what migration rate do you see the largest difference in value between the ploidy levels?*

Alternatives to F_{ST}

Despite its wide use, there are some serious problems with F_{ST}: its value depends on the mutation rate (not specific to polyploids). This is annoying, as we prefer it to describe population connectivity. A problem that is important for this workshop is that F_{ST} also depends on the ploidy level. This complicates comparisons among ploidy levels. To overcome these problems, several alternative statistics have been proposed. The statistics F'_{ST} (Meirmans & Hedrick 2011) and D (Jost 2008) are supposed to solve the dependence on the mutation rate. The *rho*-statistic (Ronfort *et al* 1998) was developed to circumvent the problems related to polyploids, and is therefore of particular relevance for this workshop.

38. ***Bonus for R-gurus:*** *Open the script '*Genetic Differentiation Rho.R*', and have a look at it. What are the differences with the previous script?*

39. *Run the script '*Genetic Differentiation Rho.R*' for different ploidy levels. What happens with the value of rho?*

40. *For what ploidy level is the value of rho equal to that of* F_{ST}*?*

Run the script '*Compare F-stats.R*' again, but now modify it to plot *rho* instead of F_{ST}.

41. What difference do you see with the plot you made previously for F_{ST}*?*

42. Which statistic is preferable for comparing population differentiation across ploidy levels, based on these results?

A4: The missing dosage information

In diploids, the distinction between homozygotes and heterozygotes is very straightforward: individuals with one allele have two copies of that allele (they are homozygote); individuals with two alleles have one copy of each (they are heterozygote). However, for polyploid individuals, in many traditional approaches as well as for low-coverage short-read sequencing it is unlikely that you can obtain fully resolved genotypes (i.e., to determine the dosage). When the two alleles *A* and *B* are found in a tetraploid, this could be any of the partially heterozygous genotypes *AAAB*, *AABB*, or *ABBB*. For low-coverage sequencing, it is often impossible to distinguish between these based on the number of reads, so such individuals are sometimes coded in a partially dominant way, e.g. as *AB*. This missing dosage information introduces a bias in the calculation of the allele frequencies and the summary statistics for differentiation and diversity.

Assume a sample from a tetraploid population with the following genotype frequencies:

Genotype	Individuals	Partially dominant genotype
AAAA	10	
AAAB	20	
AABB	70	
ABBB	80	
BBBB	20	

43. What are the allele frequencies when the dosage for the genotype is known (as in the first column)?

44. Fill in the last column, i.e., list the genotypes when the dosages are not known. What are the allele frequencies when those values are used?

45. Why don´t we just forget about dosage, and analyze markers in a partially dominant fashion?

The next generation sequencing revolution has greatly facilitated genotyping and thus population genetics.

46. Does NGS data also suffer from problems in the determination of dosage? Why (not)?

For SNPs called from NGS genotyping, the strength of the problem of missing dosage depends on the sequencing coverage. Actually, when the coverage is very low the missing dosage problem also applies to diploids. Imagine a ridiculously low coverage of 2.

47. What is the probability of correctly inferring the genotype for a diploid individual that is heterozygous for a certain SNP with a 2x coverage?

With reasonable sequencing depth, dosage can be inferred from the number of times the different alleles are encountered at a locus. Since polyploids are more complex than diploids, a higher sequencing depth is needed for a good scoring of heterozygotes.

The script titled '*Overlap.R*' addresses how well different genotypes at a biallelic locus can be separated for tetraploids given a specified sequencing depth. Note that the `rbinom` function is used here, as that lends itself better to the creation of histograms.

48. Run the script with multiple sequencing depths. What is the lowest sequencing depth at which you think the results are still acceptable?

Block B
Evolutionary history of polyploids

Possibly the most interesting evolutionary questions that can be analyzed using population genetics involve the analysis of multiple ploidy levels in a single species. Unfortunately, this is also when the problem of missing dosage information and incorrect genotypes is especially troublesome and may cause bias.

B1: Principal components analysis (PCA) with mixed ploidy levels

Simulated data – bias and spread

Assume a single population where diploids and tetraploids co-occur and freely interbreed (somewhat unrealistic, we know). In other words, the diploids and tetraploids form a single gene pool and have exactly the same allele frequencies. A sample of 100 individuals is taken from each cytotype and analyzed using 100 SNP markers. However, the dosage information is missing for the tetraploids.

49. Why don´t you expect any separation between these two cytotypes when performing a PCA?

Such a PCA is simulated in the script '*SNP simulation.R*'. Note that in this script we do not simulate generations of random mating with mutation and drift. Instead, we directly draw random allele frequencies and use these to construct diploid and tetraploid genotypes. These genotypes are then stripped of their dosage information.

50. Did your above expectation turn out correct? What is happening here?

51. *Do you get better results when you increase the number of loci?*

Set the number of loci back to the default of 100 and change the cytotype of ploidy2 to hexadecaploid (2n=16x), and run the model.

52. *What happens to the spread of points of the hexadecaploids along the second PCA axis? Is this because they have less genetic diversity?*

In the above example, the dosage information was removed. Now, perform the PCA on the genotypes with complete dosage information by changing line 35 into `pca = prcomp(allFreqs)` and run the simulation again.

53. *What happens to the bias and the spread?*

Empirical data

For the following parts you will choose from one of five real datasets that we selected. Each dataset contains individually sequenced genome-wide single nucleotide polymorphism (SNP) data for diploids and autotetraploids with pure populations for each ploidy and one or more mixed-ploidy populations, where diploids, tetraploids and sometimes triploids or hexaploids co-occur.

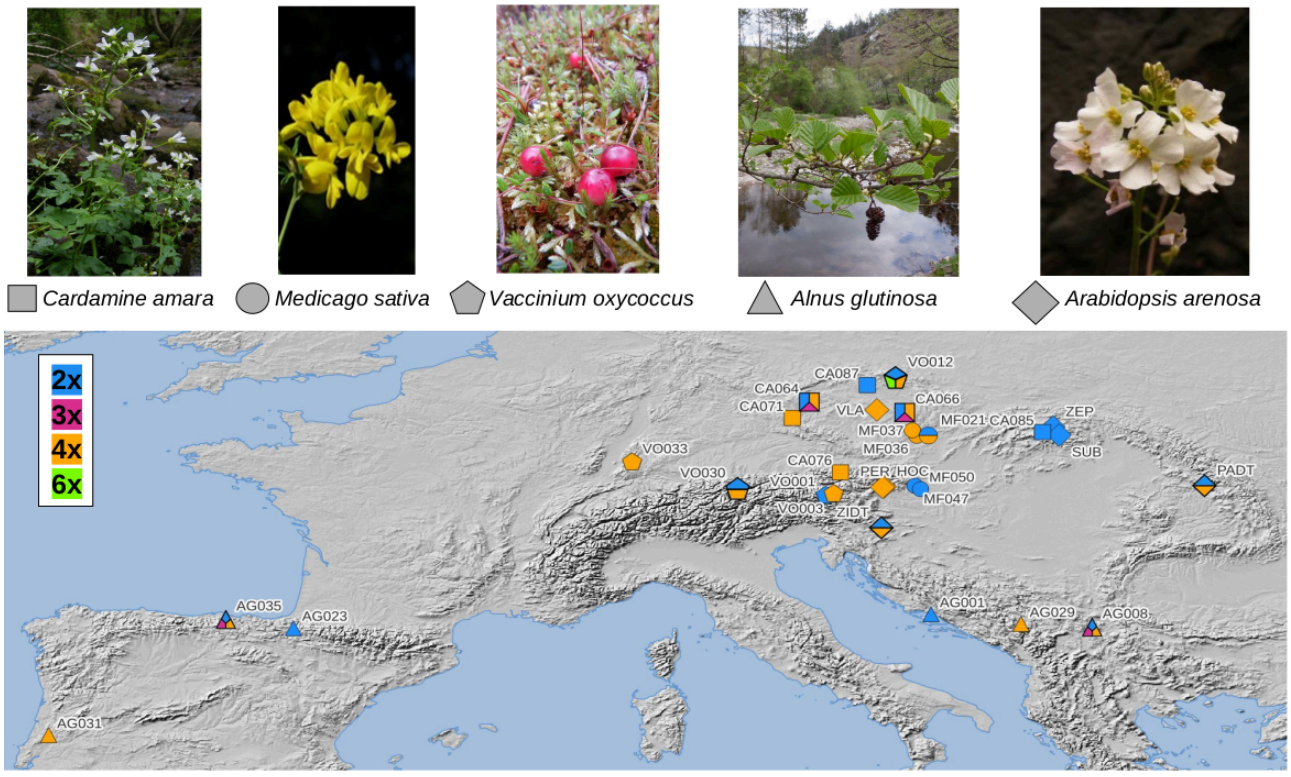


Fig. B1.1: The five species comprising and the locations of the included populations. Colors of symbols indicate population ploidy, symbols with multiple colors are mixed-ploidy populations where individuals with different ploidies co-occur in close proximity.

The 5 datasets are:

- *Arabidopsis arenosa (dataset=arenosa)*
- *Alnus glutinosa (dataset=alnus)*
- *Cardamine amara (dataset=cardamine)*
- *Medicago falcata (dataset=medicago)*
- *Vaccinium oxycoccus (dataset=vaccinium)*

You'll receive a folder for the dataset you choose, which will contain the following files:

- **dataset.fourfold.vcf.gz**

This is a genome-wide VCF file containing fourfold-degenerate (4dg) sites, which are putatively neutral. It contains biallelic SNPs, invariant sites and some degree of missing data.

- **dataset.fourfold.vcf.gz.tbi**

This is an index of the fourfold VCF file generated using the tabix command. Such an index allows software like bcftools to randomly access any place in the VCF file, defined by chromosome name and location, without reading through the rest of the file first.

- **dataset.pops.tsv**

This is a tab-separated file, which assigns a population ID and ploidy to each sample in the VCF file. You can open and edit it with a regular text editor. Ploidy of each individual has been a priori identified using flow cytometry.

- **dataset.pruned.vcf.gz**

This is a pruned version of the fourfold vcf file above. Pruning means subsetting a VCF file based on physical distance and genetic correlations between sites so that we end up with a representative subset of genome-wide SNPs, that are independent of each other (have low linkage disequilibrium), but capture the large-scale population structure. This is useful for computational reasons, but it can also be beneficial for population genomic analyses which assume independent segregation among loci. For all five datasets, we started with the dataset.fourfold.vcf.gz and generated the pruned dataset using the +prune module from bcftools. In case you want to know more or try it yourself, bcftools can also handle mixed-ploidy datasets using the following command, which selects up to 10 bi-allelic sites per 100kb window, with a maximum genetic correlation of 0.2:

```
bcftools +prune -m 0.2 -w "100kb" -n 10 -N "rand"
    dataset.fourfold.vcf.gz | bcftools view -m 2 -o
    dataset.pruned.vcf.gz
```

Principal Components Analysis

We will supply you with R scripts for most of the following parts, for this part you'll use the script PCA.r. We wrote these scripts so that they are self-contained and you only need to supply some parameters (mostly names of input and output files) to run the complete script. The scripts can be run both directly in Rstudio and via the command line using the Rscript command. You will find instructions on how to set parameters and how to run the script at the top of each script. We added comments throughout each script to make it easier for you to understand how these scripts work so that you can modify them or reuse parts of them for your own analyses.

Now for the PCA part, look into the PCA.r script and run it by using the pruned dataset (dataset.pruned.vcf.gz) as vcf input file (set the vcf_input variable at the beginning of the R script) and the dataset.pops.tsv file as population map (set the pop_map variable). This will generate two output files: a table with the raw data of your PCA (set with the pca_PCs_out variable) and a pdf file (set the pca_plot variable) with a plot of your PCA.

54. *Describe your PCA, in terms of clustering, dispersion, etc.*

55. *Which factor is more important - ploidy level or geography?*

56. *Revisit what you have learned about the potential effects of ploidy on patterns in a PCA (e.g., dispersion and spurious clustering by ploidy). Do you observe these effects here? Interpret the patterns you see.*

B2: Assessing relationships between populations and individuals

In this part we will aim to understand the relationships between our populations of interest beyond simple plotting with PCA: with trees, networks, and a range of summary statistics. This way, you will apply your newly attained theoretical knowledge from the previous blocks and become familiar with some useful tools.

First, you will generate estimates of genetic distance between individuals and populations of your dataset. Then you will estimate multiple population genetic summary statistics using an external script called `piawka`.

To get started, let's add a tree to look at in conjunction with our PCA from B1. To do this, we will use the NeisD.r script to estimate distances using Nei's D (1972) and use those estimates to plot heatmaps and generate neighbor-joining trees and neighbor networks. We can estimate Nei's D both on the level of populations, so that it will be based on all individuals in a population defined in the population table, and on the level of individuals, so that distance will be calculated between all pairwise combinations of individuals independent of population assignment.

57. *Based on your PCA, which pair of populations do you expect has the largest genetic distance between them? Which set do you think is closest?*

We will start by estimating Nei's D on the population level. Set the script_level variable in the NeisD.r script to pop and run the script according to the instructions on the top of the script to generate tree using the nj function. Have a look at the plot of your neighbor-joining tree. You may notice that mixed-ploidy populations in your dataset are represented by a single branch.

58. *Do you think it make sense to represent mixed-ploidy populations in this way if we are interested in the relationships between different ploidies? If not, what could be a better way to represent mixed-ploidy populations?*

Try to figure out a better way to infer population-level Nei's D, that also properly represents variation present in the mixed-ploidy population(s) and rerun your script. Then have a look at your NJ tree again.

59. *Based on our tree, which populations appear more closely related to each other - populations of the same ploidy or from the same environment?*

60. *Is a bifurcating tree the best way to represent these data? What might we be missing by using a tree?*

To avoid oversimplifying the relationships, let's have a look at the population-level network (generated using the neighborNet function in the script) instead of a tree.

61. *Which populations have additional connections in the NeighborNet? What factor could correspond to these connections?*

The previous analysis showed you relationships between populations, but often we are also interested in a more fine-grained individual-level analysis. You can run this type of analysis using the same script, by setting the script_level variable in the NeisD.r script to "ind" and rerun the script as before. Note that this may take much longer than running the script on the population level, because it has to estimate more pairwise combinations between individuals than it would have to do between populations. In particular generating the individual-based neighbor-net may take a long time. If this part takes too long, the script will also output the pairwise individual-based distance matrix in Phylip format (*.dst, defined by the distance_phylip variable) before estimating the neighbor-net. You can open this file using the external `SplitsTree` software, which can generate neighbornets much faster than the Phangorn R package which is used by the NeisD.r script.

62. *Does the network reveal anything you weren't expecting or couldn't see in our other visualizations?*

63. *Do you think there was a single origin of tetraploids, or more than one? Why?*

B3: Comparing within- and between-population statistics

Now we have a solid understanding of the relationships among our populations after looking at all of them at once, let us now characterize the diversity and differentiation among the populations, using some of the summary statistics from the previous sections. To calculate within- and between-population statistics we will use a handy set of scripts called `piawka`, that has been developed in the lab of Polina Novikova.

For the following part you will work with the complete dataset containing all (not pruned) fourfold degenerate sites (dataset.fourfold.vcf.gz). This dataset contains several million sites and when you work with such large datasets you can quickly run into limits of runtime and memory. These types of issues can become particularly pronounced when you work with R, because R is very bad at handling large objects in memory. Often when you try to load objects into memory in R that are above a certain size it will slow down considerably and it may run out of memory much sooner than you would think based on the size of your dataset.

In the previous parts we worked with the pruned VCF files, which are small and can be loaded into memory in R without problems, but with larger datasets we have to use different approaches. In the first case we use `piawka`, which is not based on R but on the command line utility `awk`, and it only loads chunks of the input data into memory while estimating summary statistics. For the following SFS part in session C you will see how a similar solution can be implemented in R.

Before running `piawka` we need the VCF file with fourfold degenerate sites (dataset.fourfold.vcf.gz) and an input table, which assigns individuals to populations. Since `piawka` infers ploidy directly from genotypes, it doesn't expect the ploidy column from the population table. You could either remove this third column by hand using a text editor or you could use the following command:

```
awk '{print $1"\t"$2}' dataset.pops.tsv > dataset.pops.piawka.tsv
```

64. *Before running piawka, think again about how your mixed-ploidy populations should be coded in your population table. Does it make more sense to leave mixed ploidy populations as a single population with different ploidies or should you separate them into multiple subpopulations based on ploidy?*

Now you can use `piawka` to estimate Pi, Tajima's D, F_{ST} and Rho using the following command:

```
piawka calc -s "pi,tajima,fst,rho" -v dataset.fourfold.vcf.gz -g
    dataset.pops.piawka.tsv > dataset.piawka.out.bed
```

This may take several minutes and in the end we will get a table with estimates per chromosome. Next, we can use the sum module of piawka to estimate genome-wide statistics using the following command:

```
piawka sum --ignore-chr dataset.piawka.out.tsv > dataset.piawka.out.genome-
    wide.bed
```

65. *Fill in the table of within-population statistics. In your output file, you'll find the names of populations in columns 5, the name of the statistic in column 7 and the value for your statistic in column 8.*

Population	Ploidy	Pi	Tajima's D

66. *Do you see any interesting patterns in your within-population statistics?*

67. *Fill in the table of between-population statistics.*

Population pair	Ploidy(-ies)	F_{ST}	rho

68. *Look at the between-population statistics. Which statistic is more meaningful for our population pairs - rho or F_{ST}? Specifically, which would you use to answer these two questions? Are diploid populations more differentiated from each other than tetraploid populations? What is the level of genetic differentiation between diploid and tetraploid populations? And is there a difference in the levels of genetic differentiation when contrasting ploidies from pure vs. mixed populations? Hint: think back to exercises on these statistics from the previous days.*

B4 - BONUS: Bayesian clustering analysis with the *Structure* algorithm

The program `structure` (current version 2.3.4) by Pritchard *et al* (2000) has been around for more than 25 years, but it is still widely used. It can accommodate datasets with variable ploidy and simulations have shown that it is more robust than other clustering methods for mixed-ploidy data (Stift et al. 2019). We will not go into the exact working of the program (and the algorithm it uses) in this workshop. In short, the program uses assignment methods to assign individuals to a specified number of populations (*k*). It then uses Bayesian methods, including a Monte Carlo Markov Chain, to find the optimal distribution of individuals over the *k* clusters. One of the most interesting aspects is that individuals are allowed to be admixed, meaning that they can be assigned to multiple clusters.

The original `structure` package came with a graphical interface, which allows users to set parameters, load input files and run replicate structure runs. However, this interface often feels clunky by todays standards of usability and it gets increasingly tedious to get it to run on modern operating systems. In addition, we can't use it if we want to integrate `structure` into automated bioinformatics workflows. So instead of using this interface here we use the command line version of `structure`, which can be run on a variety of computing environments. To generate input files, you will use the provided structure_input.r script, which will take a VCF file and generate files in `structure` format. Since `structure` was developed before genome-scale datasets became widely available, it has not been optimized to work on large datasets. However, running it on our pruned datasets is entirely feasible.

As before, run the structure_input.r script using the pruned dataset (dataset.pruned.vcf.gz) as VCF input file (set the vcf_input variable at the beginning of the R script) and the dataset.pops.tsv file as population map (set the pop_map variable). This will generate three output files required to run `structure`: your `structure` input file (set with the structure_out variable), a main parameter file (set with the mainparams_out variable) and an extra parameters file (set with the extraparams_out variable).

Your main parameter file will contain the most relevant options you may want to change when running `structure`. The file generated by the structure_input.r script contains a set of default values and for adjusting the parameters of your structure run you will have to change them using a text editor. The most relevant options you'll likely want to change are:

- The name of your structure input file, will be the same as defined by the structure_out variable:

```
#define INFILE dataset.struct
```

- The name of the output file of your `structure` run. Note that structure will append "_f" to this name for the actual output file name

```
#define OUTFILE dataset.struct.out
```

- The number of populations (k), which `structure` will assume (default 2)

```
#define MAXPOPS 2
```

- Burnin - The number of iterations of the Markov Chain `structure` will run before recording data (default 1000)

```
#define BURNIN 1000
```

- Number of replicates – the number of iterations of the Markov Chain `structure` will use for estimating cluster assignment probabilities (default 10000)

```
#define NUMREPS 10000
```

There are numerous other options in the main params and extra params file, which are explained in the structure manual, but we will leave them here as they are. To run `structure` you just run the `structure` command line program. If it is run in a folder, where the main parameter file is present and named "mainparams" and the extra parameter file is present and named "extraparams", we don't have to supply any additional parameters and `structure` will get all information (including the name of input and output files) from these two files. If your files are named differently or in a different place you can supply the names of the mainparams file with the -m option and the name of the extra parameters file with the -e option. Also you can overwrite the name of input and output files and the number of populations (*k*) set in the main parameters file by supplying the structure command with the -i, -o and -K parameters, respectively.

The output of your `structure` run will be a text file with various information and the table with the individual assignment probabilities is located somewhere in the middle of it. To make understanding these output files straightforward, we provided you with an additional R script named plot_structure.r, which plots these results as a barplots. To run it you have to provide your `structure` output file (option structure_results) and the same population map you used before (option pop_map) and it will generate your barplot as pdf file (option output_pdf). This plot will be based on a single structure run, however a typical study that uses `structure` would estimate mean assignment values across multiple replicate runs. While this goes beyond the scope of this course,

you can have a look at the `clumppling` software (Liu et al. 2026) if you want to learn how to do this.

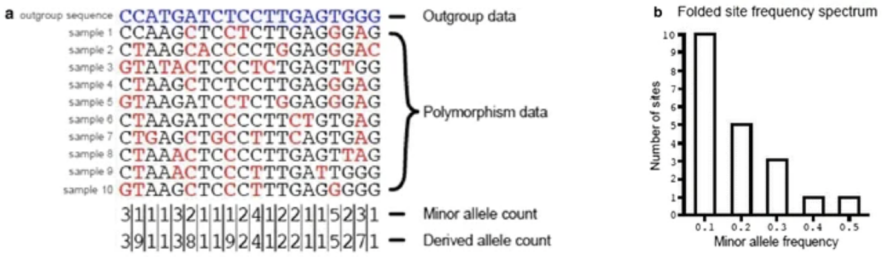
69. *Generate structure input files and do individual structure runs for multiple values of k (2-6) for your dataset. Plot your results and compare them to your previous results. Do your results reflect differences in ploidy or population structure?*

70. *In the previous exercises it mattered a lot whether you coded your mixed-ploidy populations as single populations or if you coded your ploidies as separate populations. How do you think this will affect your structure results? Try to rerun one of your structure runs with a different way of coding your populations so that you can compare both possibilities.*

Block C

Population genomic signatures of auto- vs. allo-polyploidy and mixed inheritance

The summary statistics we have used above are useful to assess our data, whether based on a few loci or for enormous genomic datasets. Another kind of summary we can generate for large genomic datasets is the **site frequency spectrum** (SFS; sometimes also called the allele frequency spectrum). The SFS is essentially a histogram of allele frequency within a population (or set of populations).



A "sample" in the alignment means one haplotype, where each base is a site. An allele count of 1 means that one of the sampled individuals has one allele at that site. In the folded site frequency spectrum, the *x* axis is the frequency of an allele in the population (allele counts), and the *y* axis is the number of sites in the dataset that fall in each frequency bin. This specific example thus tells us that low frequency alleles are in abundance. Figure excerpted from: Booker et al (2017).

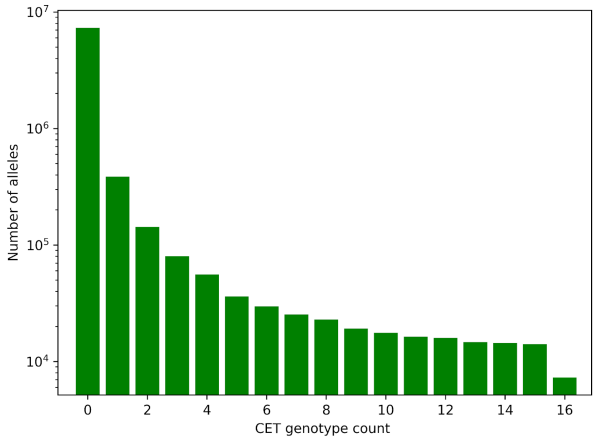


Fig C1.1 Example of a SFS of Minor Allele Frequencies.

71. *Look at the example SFS of Minor Allele Frequency. If this is a diploid population, how many individuals are in the dataset? What about if it was a tetraploid population?*

72. *Why is there generally such an abundance of rare alleles (e.g. singletons?) in SFS? Would you expect an excess of this category to be higher or lower in an autopolyploid population of the same size (N of individuals)?*

Table C2.1. Bi-allelic SNP data for 10 loci and 12 individuals.

sample	snp 1	snp 2	snp 3	snp 4	snp 5	snp 6	snp 7	snp 8	snp 9	snp 10
1	0	0	1	0	0	0	0	0	0	1
2	1	0	1	0	0	0	0	1	0	0
3	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	1	1	0	0
5	0	1	0	0	0	1	1	0	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	1	0	0
9	0	1	1	0	0	0	0	1	0	0
10	1	0	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	1	0	0	0	0	0	0	0

73. *To get familiar with the SFS, let's calculate one by hand. For a small population and a handful of SNPs, this is trivial – for larger datasets we will let the computer do the work. Calculate (or draw) your SFS for the bi-allelic SNP data in table C2.1.*

74. *Is dosage information important for estimating an SFS? Why or why not?*

Now you will generate your own SFS based on real data. You will use an R script, which will generate and plot SFS based on your whole dataset. This is a more complex script than the ones from the previous exercises, since it implements several solutions related to SFS calculation

- **Handling of missing data**

  Most datasets will have missing data, however if you think through the logic of SFS you can see that there is no obvious way to handle it. The SFS categories are the counts of alt alleles, but this assumes that this count is always based on the same number of haplotypes. However, this number will vary based on the number of samples with missing data. In other words, if we have three samples with missing data at one site, and zero samples with missing data at another site, their alt allele counts are not directly comparable. The solution implemented here is to subsample the number of haplotypes per site and per population, and exclude sites which have too much missing data after subsampling. In the script, you can control the degree of subsampling by setting the proportion of total haplotypes to keep (by setting the prop_subsample variable). A higher proportion will give you a larger (wider) and more informative SFS, but depending on the amount of missing data it will also dismiss more SNPs.

- **Dealing with large datasets**

  As discussed above, R is not good at handling large objects in memory, so here we implemented two solutions, to read your dataset into memory in easily digestible chunks. The first solution uses the vcfR package, which we also used in the previous scripts. You can tell vcfR to start reading chunks of VCF files defined by the numbers of sites to be read, and the number of sites that should be skipped before vcfR starts storing lines in memory. The downside of this approach is that it will have to read through all sites that come before the sites that are supposed to be stored in memory. This means that as we progress through the VCF file, each chunk will take longer, because R will have to skim through *all* preceding sites. For the second solution implemented here, R calls the external `bcftools` software, which uses a VCF index to start reading the VCF file at any point. Both approaches will give the same results – the first approach is slow, but requires no software aside from R, and while the second approach is faster, it requires `bcftools` to be installed and callable from R, which may not be the case on all configurations and operating systems. You can choose which approach you use by setting the read_vcf variable to either “bcftools” or “vcfR”.

- **Generating different types of SFS**

  There are two general types of SFS, folded and unfolded. In most cases we use a so-called folded SFS, where we count the *minor* allele (and correspondingly, the x-axis ends at 0.5). An unfolded SFS requires more knowledge about the system, because we count only the *derived* alleles – meaning we have to know which allelic state is ancestral, what can be inferred ideally using multiple outgroups. The provided script can do both (you can change this by setting the folded variable to either "unfolded" or "folded"). In addition, there is also the option to count alternative (derived) alleles in pairs of populations. While this goes beyond the scope of this course, the provided script can also generate and plot 2D SFS, which are two-dimensional counts of alleles for all pairwise combinations of populations.

The shape of the site frequency spectrum is determined by selection as well as by demographic processes like bottlenecks, population expansion, migration, etc. Therefore, it is often used for demographic modeling. We won't cover these kinds of analyses today, but the SFS has an additional utility specific to polyploid populations: it can tell us something about polyploid origins.

You probably know that polyploids can form either via genome doubling in a single lineage (autopolyploidy) or with hybridization among lineages (allopolyploidy). From a population genetics perspective, all we *really* care about is how alleles are passed on from one generation to the next. This depends on the mode of inheritance, which can be disomic or polysomic, or something in between (Stift *et al* 2008). To simplify, let's make some generalizations:

An **autotetraploid** has four homologous chromosomes, all of which are equally likely to recombine with each other during meiosis. This means all possible allelic combinations are produced in equal frequencies: **tetrasomic inheritance**.

Contrast this with an idealized, textbook **allotetraploid a la Stebbins**, which contains two sets of divergent (homeologous) chromosomes. During meiosis, recombination happens only between homologous pairs from the same parental subgenome, and alleles are passed down via **disomic inheritance**. Because there is never "shuffling" of the alleles from each subgenome, such allopolyploids have substantial fixed heterozygosity (i.e. cases when one subgenome is homozygous for A while the other is homozygous for B, resulting in genotype AABB for a tetraploid).

75. *Look at the example SFS from question 72. Is there evidence of substantial fixed heterozygosity? Why or why not?*

Now let's look in R at our data representing samples of *Arabidopsis* mapped to a reference of a diploid *A. lyrata*. Follow the script to generate SFS for populations CET and CED.

76. *Both populations comprise 6 individuals. Why is the x-axis different between the SFS?*

77. *Go ahead and generate an SFS for the rest of the tetraploid populations (pop names PUT and MYS). Look at the SFS for CET, PUT, and MYS. What can you infer about inheritance mode from the shape of the SFS?*

78. *As with any method, there are caveats, so let's consider a few.*
    a) What if different chromosomes have varying inheritance modes?
    b) What if the reference genome, to which we mapped our data, represents a diploid that is similar to one but very divergent from the other allotetraploid's subgenome?

79. *Obviously, the way we processed the population MYS by mapping short reads to a reference of a diploid A. lyrata has a caveat that does not allow further meaningful population genomic analyses. How would you change the analysis to analyze this population better (in an ideal scenario, even if such resources do not yet exist)?*

Understanding the evolutionary histories behind our polyploid (or mixed ploidy) study organisms is critical for downstream studies. As this exercise-block showed you, these histories can be complex and are best untangled by using multiple approaches for a holistic view.

Block D

Signatures of selection in autopolyploids

Preparation: read Monnahan & Brandvain (2020). The effect of autopolyploidy on population genetic signals of hard sweeps. Biology Letters. https://doi.org/10.1098/rsbl.2019.0796

Can autopolyploids efficiently adapt to environmental changes? If yes, do they do it more (or less) efficiently than diploids? How do polyploids adapt to the specific cellular change which created them – whole genome duplication? The main goal of the following exercises is to learn how we can use genomic data to address these questions. Specifically, you will learn how to conduct scans for signals of positive selection within population genomic data of diploids and autopolyploids, based on empirical model datasets (you know them already know from the previous day). However, we will first discuss theoretical expectations derived from simulations in the vain of the *Monnahan & Brandvain (2020)* paper (which you certainly enjoyed reading before the course). Then, we will learn how to detect signatures of positive selection and specifically explore the effect of ploidy on the manifestation of these signals. Finally, we will compare our empirical results to theoretical expectations and discuss potential experimental designs to test them. We hope that by the end of this day, you will feel well-experienced and interested in searching for signatures of adaptation in your favorite polyploid organism!

D1: Natural selection in autotetraploids

Theoretical expectations

In the first exercise you will explore the effects of natural selection at a single locus in autotetraploids. Step-by-step we will explain the theoretical expectations of how selection changes allele frequencies at a biallelic locus in diploids and then challenge you to derive the same steps for autotetraploids. We will assume that the produced offspring follows the Hardy-Weinberg frequencies (which you derived on day 1), which is a valid assumption if the selection does not affect the random union of gametes during reproduction. As a reminder, the expected genotype frequencies for diploids under HWE are:

$$[AA] = p^2;\ [AB] = 2pq;\ [BB] = q^2 \quad \text{and} \quad [AA] + [AB] + [BB] = 1.$$

80. What again were the expected genotype frequencies for tetraploids? Try to answer this without peaking at the answers of day 1.

You can use these expectations to derive allele frequency changes when allele A is under positive selection. In this case, allele B must be under negative selection, since it will decrease in frequency when the frequency of allele A increases. The strength of selection and thus of the allele frequency change is determined by the selection coefficient *s*, which is the difference in relative fitness between the most- and the least-fit genotype. You can then introduce a fitness term *w* for each genotype (w_{AA} for AA, w_{AB} for AB and w_{BB} for BB).

If we assume that the effect of allele A is completely dominant, genotypes AA and AB are equally fit. We can thus determine that the relative fitness of genotypes AA (w_{AA}) and AB (w_{AB}) is 1, while the relative fitness of genotype BB (w_{BB}) is 1-*s*.

81. Under complete dominance of allele A, write down the fitness of each of the five possible tetraploid genotypes.

The change in genotype frequencies is determined by multiplying these fitness terms with the expected HW frequencies for each genotype. However, since allele frequencies have to sum to 1 again after selection, you also have to divide these frequencies by the mean fitness of the population w_m, which is the sum of the relative fitnesses of all genotypes. For diploids, this is calculated as:

$$w_m = p^2 w_{AA} + 2pq w_{AB} + q^2 w_{BB}$$

82. Write down the corresponding equation for the average fitness for tetraploids.

Genotype frequencies in diploids after one round of selection will be:

$$[AA]_{t+1} = p^2 w_{AA} / w_m$$

$$[AB]_{t+1} = 2pq w_{AB} / w_m$$

$$[BB]_{t+1} = q^2 w_{BB} / w_m$$

If we substitute the fitness values as specified above ($w_{AA} = 1$, $w_{AB} = 1$, $w_{BB} = 1\text{-}s$), the mean fitness will be:

$$w_m = p^2 + 2pq + q^2(1 - s)$$

and the resulting genotype frequency changes can be calculated by

$$[AA]_{t+1} = p^2 / w_m$$

$$[AB]_{t+1} = 2pq / w_m$$

$$[BB]_{t+1} = q^2(1 - s) / w_m$$

83. Give the corresponding genotype frequencies changes for tetraploids, for complete dominance of the A allele.

Alternatively, if the effect of allele A is completely recessive, in diploids only genotype AA will be under positive selection, while both genotypes AB and BB are under negative selection. The relative fitness for genotype AA (w_{AA}) will thus be 1, while the relative fitness of genotypes AB (w_{AB}) and BB (w_{BB}) will both be 1-s.

Now the mean fitness in diploids will be:

$$w_m = p^2 + 2pq(1 - s) + q^2(1 - s)$$

The resulting genotype frequencies will be:

$$[AA]_{t+1} = p^2 / w_m$$

$$[AB]_{t+1} = 2pq(1\text{-}s) / w_m$$

$$[BB]_{t+1} = q^2(1\text{-}s) / w_m$$

In both cases, the frequency of allele A after one round of selection is $p_{t+1} = [AA]_{t+1} + [AB]_{t+1} / 2$ and the frequency of allele B will be $q_{t+1} = 1 - p_{t+1}$.

84. What is the frequency of allele A after one round of selection in tetraploids?

85. Calculate the diploid allele frequencies after one round of selection for p=0.4, q=0.6 and s=0.1 for a completely dominant and for a completely recessive allele.

86. *Now do the same for autotetraploids, given the same values as above, and using the equations you derived above.*

Stochastic simulations using R

The efficacy of selection does not only depend on the selection coefficient and dominance, but also on other factors such as the mutation rate, population size, etc. In the next step you will use the R-script '*Selection.R'*, which can simulate single-locus allele frequency changes using stochastic simulations. You can set parameters at the top of the script and then simply run the code to do the simulations and plot allele frequency trajectories.

87. *Explore how the effects of selection changes when you change the ploidy level or the dominance of the selected allele. Leave all the other parameters unchanged. What is the effect of increased ploidy? What is the effect of recessiveness?*

In the previous task you used a very large population size (N=1000000). With such a large effective population size stochastic simulations will behave like theoretical predictions, because the effect of genetic drift is minimal. As a result, there is hardly any variation in allele frequency trajectories. Moreover, with the given parameters (high initial allele frequency and large effective population size) there is zero probability that the positively selected allele will go extinct.

Natural populations may, however, not have such large effective population sizes. Additionally, new alleles generated by mutation will start as a single copy and even if they are under strong positive selection there is a substantial probability that they will go extinct. You can simulate this scenario by reducing the number of individuals *N* and the starting frequency *p*, which will cause allele A to get lost regularly. If you run these simulations repeatedly and count how often an allele gets lost or reaches fixation you can test the combined effect of ploidy and dominance on fixation probability.

88. *Change the population size to 1000 and the starting frequency to* `1/(ploidy*N)`*, which corresponds to the frequency of an allele with a single copy in the population. Modify the code to run simulations repeatedly and implement a way to count how often a dominant allele under positive selection will go extinct or go to fixation. What are the fixation probabilities for different ploidy levels? Repeat the procedure for a recessive allele and test which parameters you could change to increase its fixation probability.*

Mutation-selection balance

Thus far, you simulated the trajectory of a single mutation with a beneficial effect that exists at the beginning of the simulation. In the following part you will introduce new deleterious mutations at the same locus with a mutation rate u. This allows you to explore how the combined effects of selection and mutation change with differences in ploidy.

89. *Set the initial allele frequency p to 1, the population size to 1000000, and the selection coefficient s to 0.01. To introduce new mutations, set the mutation rate u to 0.0001. The parameter settings now reflect a scenario of slightly deleterious mutations entering the population. How do the allele frequency trajectories differ between diploids and tetraploids? What is the effect of dominance?*

Effect of bottlenecks

In the previous parts you assumed a constant population size. In the following part you will explore what the effect of a dramatic reduction of the population size (bottleneck) on fixation probabilities and mutation-drift balance will be and how it differs between ploidies.

90. *Change the R code to introduce a genetic bottleneck, where the population size decreases from 1000 to 20 at generation 4800 (hint: use an `if`-statement). Test what effect this has on fixation probabilities in diploids and tetraploids for dominant and recessive alleles using the parameters from task 88 and how it will affect mutation selection equilibrium using the parameters from task 89.*

91. Describe what the effect of a bottleneck will be for different ploidy levels when mutation-selection equilibria for slightly deleterious recessive alleles occur simultaneously at thousands of genes across the whole genome.

D2: The effect of autopolyploidy on population genetic signals of hard sweeps

We will first take a closer look at Monnahan's paper to highlight some differences in signatures of selection between diploids and tetraploids. Before we do so, let us clarify some of the 'sweep' terminology, as this is a rather crucial aspect of the paper. A **selective sweep** refers to a process by which a new advantageous mutation eliminates or reduces variation in linked neutral sites as it increases in frequency in the population (Fig D1.1; Nielsen 2005). Generally, selective sweeps can be grouped into 'hard sweeps' and 'soft sweeps' (this is explained in detail in Box D1.1).

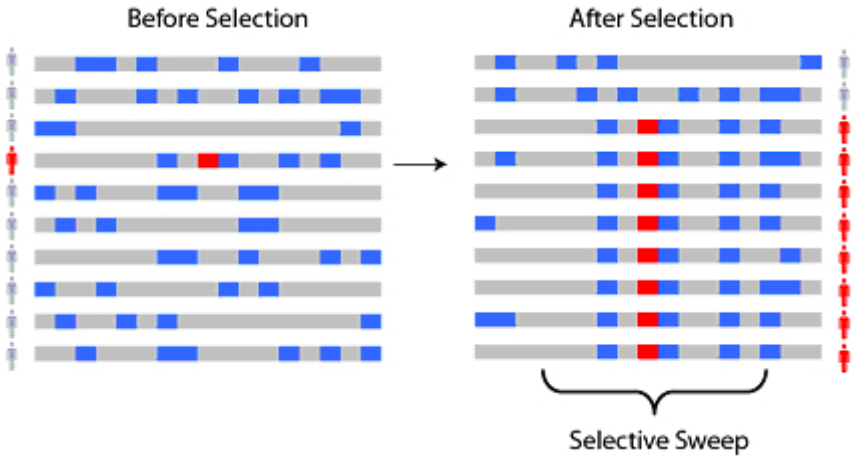
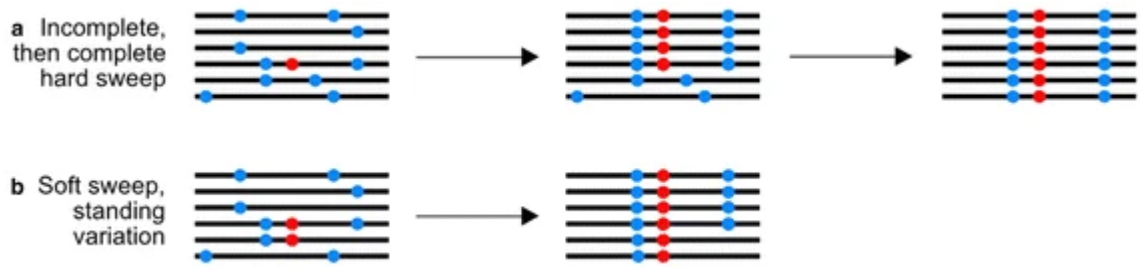


Fig D1.1. Selective sweep. Before selection (left), a beneficial mutation arises (marked in red). After selection (right), not only the frequency of the red mutation has increased, but also that of variants linked to the mutation. At the population level, this has led to a strong reduction of vaiation, both for selected site and the linked neutral sites.

Box D1.1 – Hard *vs*. Soft sweep



Hard sweeps (also: classic sweeps) are the most well-studied model of sweeps (and also the ones which are assumed in Monnahan & Brandvain (2020). In a hard sweep, a new advantageous mutation rapidly increases in frequency to eventual fixation. In a soft sweep, a neutral allele segregating in a population becomes favored (due, for example, to a change in the environment). The segregating allele may be associated with multiple haplotypes. As it rises in frequency, so do the multiple haplotypes. (Although it won't be considered here, a similar process, also termed a soft sweep, can occur if an advantageous mutation arises by multiple, distinct mutation events.)

Using the information from Monnahan's paper, try to think about following questions:

92. *Should there be any difference between diploids and tetraploids in the efficacy of positive selection acting on dominant vs. recessive alleles? What result (figure) brings the evidence to support your answer?*

93. *Natural population of autotetraploid* Arabidopsis arenosa *has nucleotide diversity* $\pi = 0.012$ *, and mutation rate of Arabidopsis is assumed to be ~7 × 10-9 base substitutions per site per generation, based on mutation accumulation experiments (Monroe et al. 2022). Let's assume that genetic diversity,* θ*, could be approximated by nucleotide diversity,* π*. Given Monnahan's definition of* θ ($\theta = 2Nk\mu$)*, try to estimate the effective population size of* A. arenosa *tetraploids.*

94. Theoretical population genetic scenarios often assume polyploid formation is associated with a strong bottleneck. Is the above result consistent with such a prediction? Why or why not?

*95. Let us for now assume that most selection events which we will be able to see in tetraploids are acting on additive or dominant loci. Would the loss of diversity in SNPs around the selected site (seen as a peak of reduced diversity and increased differentiation) be more apparent in tetraploids or in diploids? Note that recombination rate in Arabidopsis was estimated to be Rho = 3.7*10-8. Hint: Check Fig. 2 (in Monnahan's paper).*

96. What practical implications does such variability in signal strength have for the choice of optimal window size when calculating diversity and differentiation across genomes in a study including both diploids and polyploids? Consider the pros and cons.

D3: Detecting signals of selection in diploid and autotetraploid populations

To detect signatures of local adaptation in genomes, we may use a set of methods called selection scans. Selection scan approaches are typically based on a specific design, comparing sets of populations from contrasting environments/history (here called ancestral and derived). Specifically, to mixed-ploidy and autopolyploid data sets, one may wish to scan for selection between diploids and tetraploids (contrast 2x-4x), for example to ask which genes are involved in adaptation to whole-genome duplication. We will work with similar datasets used previously, starting with *Arabidopsis arenosa* with an option to explore also further so far unexplored datasets from the other plant species, *Cardamine amara* and *Medicago falcata.*

We decide to use the selection scan for excess differentiation that we will first estimate with F_{ST}, due to their broad applicability and intuitive interpretability. The goal of this part will be to detect signatures of positive selection in the empirical examples – acting between ploidies (2x-4x). We will identify ploidy-specific patterns of diversity and interpret them in line of P. Monnahan's simulations.

Open the script '*Selection_scans.R*'.

The analysis should be executed step-by-step (suggestion: set up line wrapping in your R Studio). Here, we will scan all variants in the genome to identify those which are the most differentiated (as identified by outlier F_{ST}) between our diploid and tetraploid populations. We pre-calculated the F_{ST} value for each variant in the genome for you, as this calculation is time consuming.

For practical reasons, we will use smaller datasets that contain only information about the 4-fold degenerate sites. These sites are neutral to natural selection; however their diversity patterns are sensitive to nearby selected sites (blue dots in the D2 figures!) and thus include the sweep effect.

Let's start running the script.

97. What do the columns 'start' and 'end' mean?

98. Considering your F_{ST} histogram, why is there the distribution peak around 0?

99. Where in the distribution should we look for candidate genes? What does that mean in terms of differentiation within and between ploidies?

Go back to R and we will generate Manhattan plots and identify candidates.

100. Comment about the distribution of the outliers (black dots) along the genome. What do you see with regards to their clustering along the genome?

Once we understand the genome-wide landscape of differentiation in our diploids and tetraploids and identified candidate SNPs, we can have a look at more gene – functional – level. We now want to discover which genes are located in the most differentiated regions and what are the possible functions of such genes. To identify candidate genes, you have to overlap the candidate variants with gene locations. Our data were mapped onto available reference genome assemblies for each species, for which the locations of all genes are available such as in our working examples.

101. How many candidate variants are located within each gene, on average? Min? Max? Use your own scripting skills to find out :).

102. As you saw in Monnahan's paper, the selection signal is defined by a peak of increased F_{ST} (and decreased diversity), not just by a few variants with high F_{ST}. Thus, let's further select only genes with high outlier SNP number and density. How many candidate genes do you have in the end? What is the mean/median number of candidate variants per candidate gene? Use functions `nrow()`, `mean()`, `median()` *to find out.*

103. In general, what do the F_{ST} plots show? Discuss it with your colleagues.

Let's have a look at the function of the genes. *A. arenosa* genome is well annotated (gene, start and end) but functions are poorly annotated. Luckily, *A. thaliana* is both the most studied plant species and a close relative of *A. arenosa*. By using Orthofinder, we made a list of correspondences between *A. arenosa* and *A. thaliana*. Run the script to extract *A. thaliana* orthologs of your candidate genes.

104. Do you see anything particularly interesting in terms of biological processes in which these genes are involved?

In addition to using F_{ST} to identify genes potentially under selection, we can use other summary statistics as well. Today we will use *pi* (nucleotide diversity within a population) and *dxy* (diversity between populations) to look for additional candidates. While *pi* reflects the number of differences for that site/window between a pair of chromosomes randomly drawn from the same population, *dxy* measures the same but between populations. Note that while F_{ST} only requires biallelic variant sites, both *pi* and *dxy* calculations also require inclusion of invariant sites (so it is suitable for our genome-wide SNP data).

Return to R and let's continue with our script. Load the pi input file and take a peek at it with `head()`. Recall that we calculated F_{ST} for each SNP.

105. What is the window being used for pi/dxy calculation?

106. What processes would you expect to result in high or low values of pi within a genome?

107. Now take a look at Dxy, which is reported for diploid vs tetraploid populations. How can we interpret high vs low Dxy? Consider some processes that could result in high or low Dxy.

108. Given what you know about pi and Dxy, what combination of those metrics are we looking for to identify candidate genes for adaptation to whole genome duplication? E.g. high pi in diploids + low Dxy, low pi in tetraploids + low Dxy, etc… Discuss with your colleagues and explain your answer.

Continue with the script to generate candidate gene lists.

109. How many candidates do you have?

110. We now have sets of candidate genes from scans using two different summary statistics. If we wanted to come up with a conservative list of candidates, what could we do? (....now do it in R)

111. Now that we have refine the candidates you can have a look at the literature and elaborate about the functions required for adaptation to whole genome duplication. Looking at your shortlist of candidate genes, go to tair.org and check out the function of a few of them. What processes are our candidate genes involved in? Can you think of how they might be beneficial to polyploids?

You might think about what to do next to validate the candidates. Common practice would be using the gene function annotation and experimentally checking for the gene effect (gene knock out, etc.). As a bioinformatician, you might rather stay behind your computer and wonder: What about testing whether selection footprints occur in the same genes/functions in other species that have also encountered WGD? Choose another species from the available data (or your own data!) and run the same analysis. Ultimately, explore the overlap of the candidate loci (indicating genetic convergence) between species

112. Re-run the scripts for one or two more species and compare their candidates. Do you see a pattern in the candidate genes found? Any convergence observed?

In conclusion, selection scans may provide interesting insights into the genomics of adaptive evolution in both diploids and autopolyploids. However, we stress that similarly to the diploid case, and likely even more, selection scans should serve as hypothesis-generating tools to stimulate

further research, not an approach to confidently identify the adaptive genes. Understanding an adaptive role of specific SNPs will require functional or *in silico* validation of candidate selected alleles, i.e. modelling the impact of candidate allele on a protein structure or introduction of adaptive allele into a common background and testing for its effect in a common garden experiment.

Thank you for your effort and interest and we hope that this motivated you to become interested in adaptation of polyploids!

References

Booker TR, Jackson BC & Keightley PD (2017) Detecting positive selection in the genome. BMC Biology 15: 1-10

Jost L (2008) G_{ST} and its relatives do not measure differentiation. Molecular Ecology 17: 4015-4026.

Liu, X., Rosenberg, N. A., & Ramachandran, S. (2026) Clumppling 2.0: A Clustering Alignment Program for Population Structure Analyses. Human population genetics and genomics, 6(2), 0004.

Meirmans PGM & Hedrick PW (2011) Assessing population structure: F_{ST} and related measures. Molecular Ecology Resources 11: 5-18.

Monnahan P & Brandvain Y (2020) The effect of autopolyploidy on population genetic signals of hard sweeps. Biology Letters 16: 20190796.

Monroe JG, Srikant T, Carbonell-Bejerano *et al* (2022) Mutation bias reflects natural selection in *Arabidopsis thaliana*. Nature 602: 101-105.

Nei M (1972) Genetic distance between populations. The American Naturalist 106: 283-292.

Nei M (1987) Molecular Evolutionary Genetics. Columbia University Press, New York.

Nielsen R (2005) Molecular signatures of natural selection. Ann. Review of Genetics 39: 197-218.

Pritchard JK, Stephens M, Donnelly P (2000) Inference of population structure using multilocus genotype data. Genetics 155: 945

Ronfort J, Jenczewski E, Bataillon T, Rousset F (1998) Analysis of population structure in autotetraploid species. Genetics 150: 921-930.

Stift M, Berenos C, Kuperus P & van Tienderen PH (2008) Segregation models for disomic, tetrasomic and intermediate inheritance in tetraploids: a general procedure applied to Rorippa (Yellow Cress) microsatellite data. Genetics 179: 2113-2123

Stift, M., Kolář, F., & Meirmans, P. G. (2019) Structure is more robust than other clustering methods in simulated mixed-ploidy populations. Heredity, 123(4), 429-441.