# Population Genomics Lab 01

Dr Josephine Paris, Dr Emiliano Trucchi and Dr Joan Ferrer Obiol (*in absentia*)

In this first part of the workshop, we're going to make use of a couple of different datasets to demonstrate some fundamental steps in investigating your population genomic data. Starting from a raw Variant Call Format (VCF) file, we will explore:

1. Filtering variants
2. Population structure
3. Diversity statistics
4. Preliminary selection scans

***Datasets***:

1. ***DS1*** comprises whole-genome sequencing (WGS) data from experimentally introduced populations of the Trinidadian guppy (*Poecilia reticulata*)
2. ***DS2*** comprises whole-genome sequencing (WGS) data combined from experimentally introduced populations, and natural populations of the Trinidadian guppy (*Poecilia reticulata*)
3. ***DS3*** comprises *phased* whole-genome sequencing (WGS) data from the experimentally introduced populations. One VCF file per population.

The files for these datasets can be found in:

```
~/workshop_materials/pop_gen/lab_01
```

The datasets are subsampled from the full VCF files from the following publications:

- Whiting, JR, Paris JR, van der Zee MJ, Parsons PJ, Weigel D and Fraser BA (2021). "Drainage-Structuring of Ancestral Variation and a Common Functional Pathway Shape Limited Genomic Convergence in Natural High- and Low-Predation Guppies." *PLoS Genetics* 17 (5): e1009566
- van der Zee MJ, Whiting JR, Paris JR, Bassar RD, Travis J, Weigel D, Reznick DN and Fraser BA (2022). "Rapid Genomic Convergent Evolution in Experimental Populations of Trinidadian Guppies (*Poecilia reticulata*)." *Evolution Letters* 6 (2): 149–61

# Part I

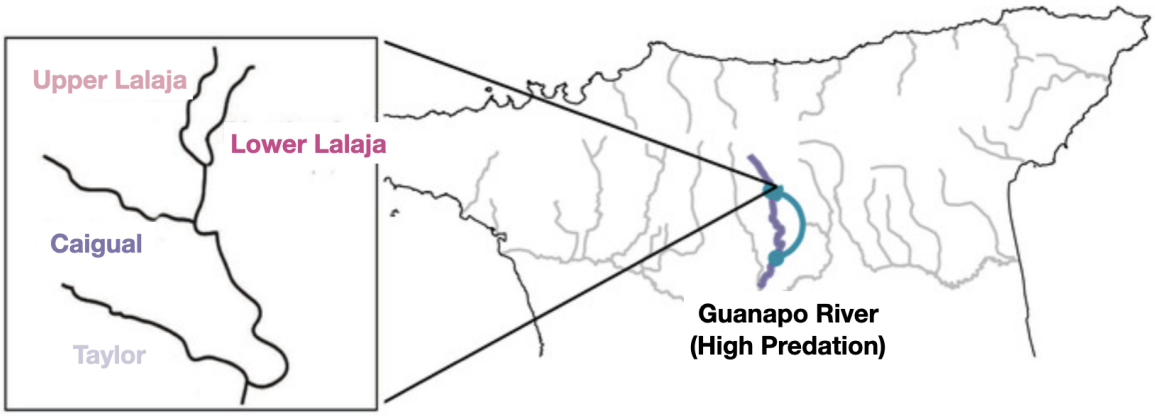
We are going to start by using the first dataset (**DS1**) to explore filtering variants.

***Preamble on DS1:***

Guppies in the Northern Range Mountains of Trinidad are a well-known model for studying phenotypic evolution in wild and experimental populations. The mountains of Northern Trinidad are drained by a set of parallel rivers punctuated by waterfalls preventing upstream colonisation by larger fish species, including guppy predators, resulting in replicated gradients of environments from high-predation (HP) to low-predation (LP) environments within rivers.

In 2008 and 2009, a set of **four** experimental low-predation populations were initiated from the same high-predation source population (Guanapo High Predation), which is downstream from the four populations in the same drainage. Natural barriers restrict migration into the experimental populations. The experimental LP populations are called:

- Upper Lalaja (UL)
- Lower Lalaja (LL)
- Taylor (T)
- Caigual (C)




We will start with a raw VCF file, generated at the end of using the GATK pipeline. We are going to filter this VCF file to create a set of variants.

Let's begin by exploring the dataset a bit. The first file for ***DS1*** (guppy_intros.vcf) is inside the directory `~/workshop_materials/pop_gen/filtering` Move into this directory

1. First let's count how many total variants are in the dataset.

***HINT***: Recall that a VCF file comprises header line information (which starts with a #) followed by one line per variant. Use some basic UNIX to count how many variants we have.

2. Now let's look at what samples we have in the dataset. The easiest way to do this is using bcftools

```
bcftools query -l guppy_intros.vcf
```

bcftools is the sister program to vcftools (you were introduced to this cool software earlier in the workshop). bcftools contains some fantastic ways to explore your VCF file. For some handy commands using bcftools, check out this useful cheatsheet hosted by Ernesto Lowy on GitHub.

3. Now let's count how many total samples are in the dataset.

```
bcftools query -l guppy_intros.vcf | wc -l
```

4. How many samples do we have from the founding population (Guanapo High Predation)?

***HINT***: use UNIX pipes from the above command to do this quickly!

5. How many samples do we have for each of the experimentally introduced low-predation populations?
6. We're going to begin by filtering the dataset. Remember that each row of a VCF file describes one genetic variant in the population, and the columns describe the allele calls for each individual. See the Wikipedia VCF page here, if you need to remind yourself about the VCF file format.
7. We're going to start by looking at and filtering variants. Take a look at the first few variants in the VCF file (using `grep -v "^#"` as we're not interested in the header lines).
8. What do you think is happening at these variant positions:

chr1 20362
chr1 122010
chr1 152524

***HINT***: use `grep "^chr1<tab>20362"` to find these variants quickly (where <tab> = Ctrl+V + TAB)
OR `grep "^chr1[[:space:]]20362"`
OR `grep -P "^chr1\t20362"`

These are indels (insertion-deletions) and for (many) population genomics analyses we only want to use single nucleotide polymorphisms (SNPs). We can exclude indels (i.e. keep only SNPs) using vcftools.

Here's a link to the manual page of vcftools to see all the options.

9. See if you can figure out how to exclude indels from our VCF file and write this output to a new VCF file. Don't forget to include the `--recode` and `--recode-INFO-all`

options when writing the new VCF file. Write to a VCF new file: `guppy_intros.SNPs`

You should remove 18437 indels from the VCF file, resulting in the message:

“After filtering, kept 55262 out of a possible 73699 Sites”

We also only want to include bi-allelic SNPs, i.e. keeping only sites with two alleles, the reference genome allele being one, and the derived or variant allele being the other.

To learn more about multiallelic sites and their frequency in natural populations, see this article by Heng Li.

10. Take a look at the manual page of vcftools on how to keep only bi-allelic SNPs. Don’t forget to use your SNP-only VCF file as input. Again, don’t forget to include the `--recode` and `--recode-INFO-all` options when writing the new VCF file. Write to a VCF new file: `guppy_intros.SNPs.bi`

You should remove 432 non-biallelic SNPs from the VCF file, resulting in the message:

“After filtering, kept 54830 out of a possible 55262 Sites”

Great! Now we have a filtered VCF file containing only bi-allelic SNPs. But what about the quality of these SNPs? There are several ways to process your SNPs for quality. We’re going to take a look at one approach below. To do this, we’re going to use vcftools again.

11. Let’s first look at the mean quality for each site:

```
vcftools --vcf guppy_intros.SNPs.bi.recode.vcf \
--site-quality --out guppy_intros_QC
```

***NB*** remember that the \ parameter allows you to continue your code neatly without it wrapping horribly ugly like on a single line

12. Take a look at the first few lines of the generated output file (`guppy_intros_QC.lqual`). The third column shows the QUAL value for each of our SNPs. Recall that QUAL is the Phred-scaled probability that the site has no variant and is computed as:

QUAL = -10*log10 (posterior probability of a homozygous-reference genotype (GT=0/0). That is, QUAL = GP (GT=0/0), where GP = posterior genotype probability in Phred scale. QUAL = 20 means there is a 99% probability that there is a variant at this site

13. Ok, so what’s our minimum QUAL value? Let’s use some awk to find out.

```
awk 'NR==2 || $3 < min {min=$3} END {print min}' guppy_intros_QC.lqual
```

This should print 10.01

***What is this code doing?*** We are telling awk to set the number of records (NR) to 2 (this skips the header line in the file). The || is a logical where we set column 3 (the column containing our qual information) by using the $ which specifies the field ($3) for the min value. END is an awk statement which is executed after the input (in this case calculating the minimum value) is completed. At the end we ask it to print the minimum value.

***NB***: You could, of course, perform these calculations in R, but we think it's good to show you a diversity of approaches. awk is another basic UNIX program distributed in most OS' (including Windows!). Unfortunately, in our UNIX workshop, we didn't have enough time to cover it in too much detail. It's useful as you can use it directly in the terminal. Check out these handy awk one-liners compiled by Eric Perment. For some useful genomics commands, check out this page by Dmytro Kryvokhyzha. It's such a useful tool to have up your sleeve, so it's well worth spending some time to learn awk!

14. What's our maximum QUAL value?

```
awk 'NR==2 || $3 > max {max=$3} END {print max}' guppy_intros_QC.lqual
```

This should print 533792

15. What's our average QUAL value?

```
awk '{sum+=$3} END {print sum/(NR-1)}' guppy_intros_QC.lqual
```

This should print 3583.7 - Not bad!

16. These numbers are informative, but it's much nicer to visualise these data. This gives us a better indication of the distribution of the quality scores. Let's plot the data.

In RStudio open the plotting file QC_vcf_plotting.R and follow the instructions within the Rscript to look at the density of the quality scores. You should see that in general, the quality scores are very good! Normally a QUAL value of >=30 is considered a high-quality SNP call. How many of our variants are equal or above this threshold?

```
awk '{if ($3>=30) {print}}' guppy_intros_QC.lqual | wc -l
```

This should print 51050

17. QUAL is an important filter to use to be sure that our variants have been called with our chosen genotype caller (for example GATK) with high confidence. However, other metadata contained in the VCF file can also be used to filter for variants we would

normally not want to include in our dataset. Let's start by looking at the mean depth per SNP:

```
vcftools --vcf guppy_intros.SNPs.bi.recode.vcf \
--site-mean-depth --out guppy_intros_QC
```

18. Again, let's look at our minimum, maximum and average depth per SNP (using the same awk code as above but adapted for our new output).

- Minimum:

```
awk 'NR==2 || $3 < min {min=$3} END {print min}' guppy_intros_QC.ldepth.mean
```

- Maximum:

```
awk 'NR==2 || $3 < max {max=$3} END {print max}' guppy_intros_QC.ldepth.mean
```

- Average:

```
awk '{sum+=$3} END {print sum/(NR-1)}' guppy_intros_QC.ldepth.mean
```

19. Again, let's plot these data to get a better idea of the distribution. Open the file QC_vcf_plotting.R again. And this time look at the mean depth per variant.

You can see that we have quite a few variants with excessively high depth. These are likely paralogs or repeats and we don't want to include them in our filtered dataset. Similarly, we want to filter out variants with a low depth. We can see our peak is at approximately 9x per site. Let's filter out the excessively high coverage variants and remove the variants with low coverage too.

20. Use vcftools to filter out our unwanted variants:

```
vcftools --vcf guppy_intros.SNPs.bi.recode.vcf \
--minGQ 30 --minDP 6 --max-meanDP 20 \
--recode --recode-INFO-all --out guppy_intros.filtered --max-missing 0.5
```

- `minGQ` is used to filter variants based on the QUAL
- `minDP` is used to filter (site-wise) on the minimum depth of a variant to include it
- `maxmeanDP` is used to filter (across all sites in all individuals) on the maximum depth of a variant
- `max-missing 0.5` sites where 50% of samples have missing data
- The `minDP` option converts sites below our chosen value (6) into missing data. The addition of the max-missing flag will remove sites entirely where there is 50% of the information missing across all individuals.

"After filtering, kept 5401 out of a possible 54830 Sites"

You will see that we have reduced our dataset quite a bit! But don't panic. These data are toy data only (we randomly subsampled a dataset for the purpose of this exercise ;) ). The original VCF contained 16,158,183 sites and was 41 Gb in size. Applying these same filters on the full dataset gave us 6,510,265 high-quality SNPs for analysis.

21. Finally, we're going to look at missingness in the data. Let's start by looking at missingness at an individual sample level. We may want to remove individuals which show a high amount of missing data as this indicates they did not sequence well. Keeping such individuals may bias our downstream analyses, or reduce the total amount of data we can use when we apply missingness filters across the entire dataset (across all individuals). We can first do this by assessing individual levels of missing data.

Recall that missingness is encoded in a VCF file as ./. vcftools will look for these positions across each of the individuals and report the fraction of missingness. For this last part of our filtering tutorial, we will use the VCF file called guppy_intros.missingness.vcf.gz. This VCF is compressed so we have to use the `--gzvcf` option to read the VCF file in vcftools.

```
vcftools --gzvcf guppy_intros.missingness.vcf.gz \
--missing-indv --out guppy_intros_QC_missingness
```

22. And the same again, but this time looking at missingness at a per site level:

```
vcftools --gzvcf guppy_intros.missingness.vcf.gz \
--missing-site --out guppy_intros_QC_missingness
```

23. Plot the outputs using the code in QC_vcf_plotting.R script. Based on these outputs, which individuals would you consider removing due to a high amount of missing data? How about the missingness per site?

***Conclusion for Part I***: Filtering variants is a bit of an art, and this exercise is by no-means an exhaustive analysis of all the filtering options. But hopefully we've demonstrated to you the main filters to explore, and how to do it. In reality, you will want to spend some time playing around with these filters (and others depending on your dataset in hand and your questions). Our overall advice is: 1) Explore how many variants are removed from each filtering option; 2) Visualise your data; 3) Make informed decisions the best you can; 4) Be careful, but don't spend weeks doing this - you have important biology to do!

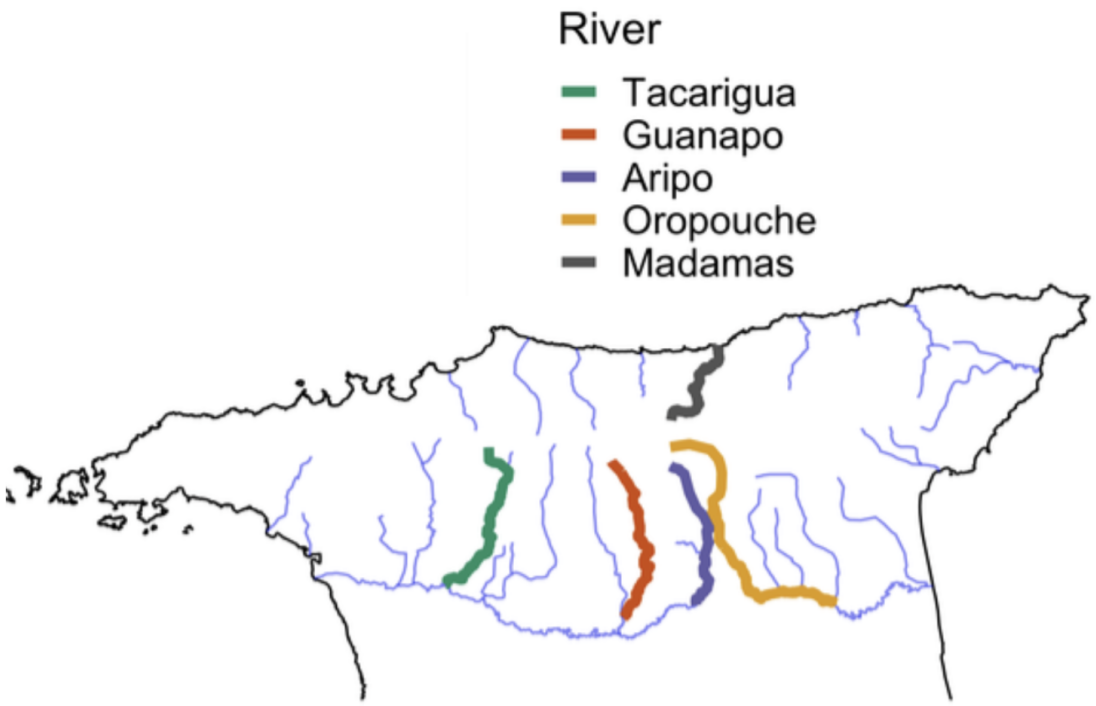
Take a short break here if you like :)

# Part IIa

Now we've seen how to filter a VCF file for informative variants for population genomics analysis. We're now going to move on to the second dataset ***DS2***, which contains pre-filtered variants from the experimentally introduced populations, which have been merged with some naturally-occurring High Predation/Low Predation populations of guppies. We're going to use this dataset to explore population structure and differentiation between the populations. Methods for understanding population demography will be explored in the next lab (Population Genomics Lab02 - after dinner).

***Preamble on DS2:***

This dataset comprises WGS of five replicated high-predation (HP) - low-predation (LP) guppy population pairs sampled across the main drainages of Northern Trinidad: Caroni drainage (Tacarigua, Guanapo, Aripo rivers), Northern drainage (Madamas river) and Oropouche drainage (Oropouche river). Each of these rivers contains a low predation and a high predation population of guppies.




We've added the experimentally introduced populations from ***Part 1*** to this dataset. As a reminder, this contains the four introduced populations: Upper Lalaja, Lower Lalaja, Caigual and Taylor. Move into the directory, where you will find the VCF file `guppy_natural_intros.vcf.gz` for this exercise.

```
~/workshop_materials/pop_gen/lab_01/pop_structure_diversity
```

We're going to start by using this dataset to investigate population structure using a Principal Components Analysis (PCA).

1. Let's begin again by exploring the dataset a little. How many variants are there? How many samples? How many samples per population?

***HINT***: Use the same code you used above to get these numbers!
***HINT***: Note that this VCF file is compressed!

2. Let's perform a Principal Components Analysis (PCA). In population genomics, you will often use a PCA as a first way of visualising the genetic relationships among and between populations. PCA is great because it's intuitive, it performs a simple transformation of the data and is model-free. However, before we can do this, we need to filter our VCF so that it only contains independent markers, i.e. markers that are not segregating together due to linkage disequilibrium or physical linkage, as these would create artefacts in the PCA. To filter out these markers, we are going to use a tool called plink v1.9. The VCF file we need to filter is called `guppy_natural_intros.vcf.gz`

3. Plink is a program originally written for the analysis of human data (bahh!) so we need to first add IDs that Plink recognises to the ID column to our VCF file using bcftools. Look at the first few entries of the VCF file first, specifically at the first three columns and recall the VCF file format. The first column is the chromosome name, the second column is the position and the third column is the ID that we are now interested in modifying.

```
zgrep -v "^#" guppy_natural_intros.vcf.gz | head | cut -f 1-3
```

```
chr1    29988   .
chr1    32730   .
chr1    48679   .
chr1    69174   .
chr1    74879   .
chr1    85377   .
chr1    87911   .
chr1    91508   .
chr1    92553   .
chr1    93758   .
```

4. We want to add the chromosome and position information to the ID column. Let's do this using bcftools. To make these steps easier, and save time typing the name of our VCF file, we will first assign the name of our VCF file as a variable. To assign a variable in the terminal, you enter the variable name followed by an equals (=) sign, followed by the name of the file (in this case the prefix). To call this variable you add a $ sign to the beginning of the variable name and surround the variable name with curly braces { }

```
# Assign variable
VCF=guppy_natural_intros

# Call variable
echo ${VCF}.vcf.gz

# Look at the first few lines
zgrep -v "^#" ${VCF}.vcf.gz | head -n 3
```

5. Ok now let's run the bcftools annotate command to add variant IDs into the ID column of the VCF file. Take a look at the manual page so that you understand what the options are doing.

```
# Add variant IDs
bcftools annotate -Oz --set-id +'%CHROM\_%POS' ${VCF}.vcf.gz \
-o  ${VCF}_IDs.vcf.gz
```

6. Check out the first few lines of our VCF file with the newly added ID column:

```
zgrep -v "^#" ${VCF}_IDs.vcf.gz |  head | cut -f 1-3
```

```
chr1    29988  chr1_29988
chr1    32730  chr1_32730
chr1    48679  chr1_48679
chr1    69174  chr1_69174
chr1    74879  chr1_74879
chr1    85377  chr1_85377
chr1    87911  chr1_87911
chr1    91508  chr1_91508
chr1    92553  chr1_92553
chr1    93758  chr1_93758
```

Cool! We've successfully added an ID column.
You'll see why we needed to do this in the next step below ;)

7. And now let's prune for linkage

```
# Prune for linkage
plink --vcf ${VCF}_IDs.vcf.gz --out ${VCF}_pruned \
--indep-pairwise 50 5 0.2 --allow-extra-chr --double-id
```

This command is removing SNPs in linkage using the `--indep-pairwise` option. The numbers after the option control how plink removes these SNPs. The first number is the window size (in the number of variants it investigates). The second number is the frame-shift for the window size (how many variants the window will move between analysis). The third number is the r2 cut-off value (designating an upper limit for how correlated/in-linkage SNPs in a window are allowed to be). 50 5 and 0.2 are commonly used values.

***NB*** the `--allow-extra-chr` argument allows for chromosome numbers and names not included in the human genome. The double-id argument allows for IDs with a "_" separator as these are present in our sample names (as we saw above in step 1).

This will output various information to STDOUT about how many variants were pruned from each chromosome. We will also get two output files.

One is called `guppy_natural_intros_pruned.prune.in` and is a list of the variants plink found **were not** in linkage together (these are the ones we want to keep!)

The other file is called `guppy_natural_intros_pruned.prune.out` and is a list of the variants that plink calculated were in linkage together (these are the ones we don't want!). See the plink Linkage Disequilibrium page for more information. If you look at the head of these files you will see that these files include the IDs we generated above.

8. Now we can make our pruned dataset!

```
# Make pruned
plink --vcf ${VCF}_IDs.vcf.gz \
--extract ${VCF}_pruned.prune.in \
--make-bed --out ${VCF}_NoLD \
--allow-extra-chr --double-id
```

You'll see here that we are extracting the pruned SNPs (not in linkage disequilibrium) based on the IDs we generated above. We ask plink to give us a bed file as output. This is plink's binary format. Check out this page on the plink website about the various plink file formats.

***NB*** the `--allow-extra-chr` argument allows for chromosome numbers and names not included in the human genome. The `--double-id` argument allows for IDs with a "_" separator as these are present in our sample names (as we saw above in step 1).

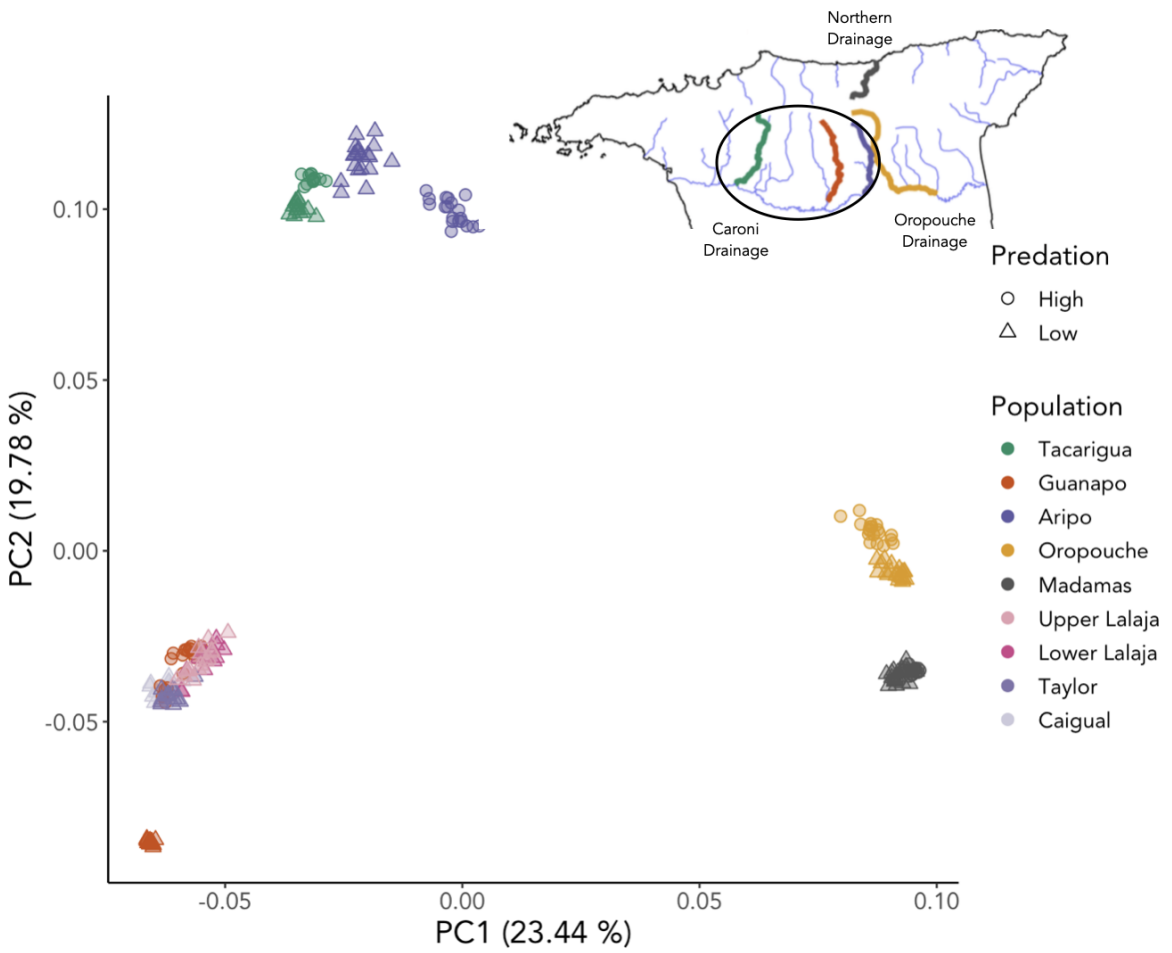
9. Now that we have our linkage-pruned dataset, we can ask plink to make the files we need for a PCA

```
 # Run PCA and make PCA files
plink --bfile ${VCF}_NoLD \
--pca --out ${VCF}_NoLD_PCA --allow-extra-chr
```

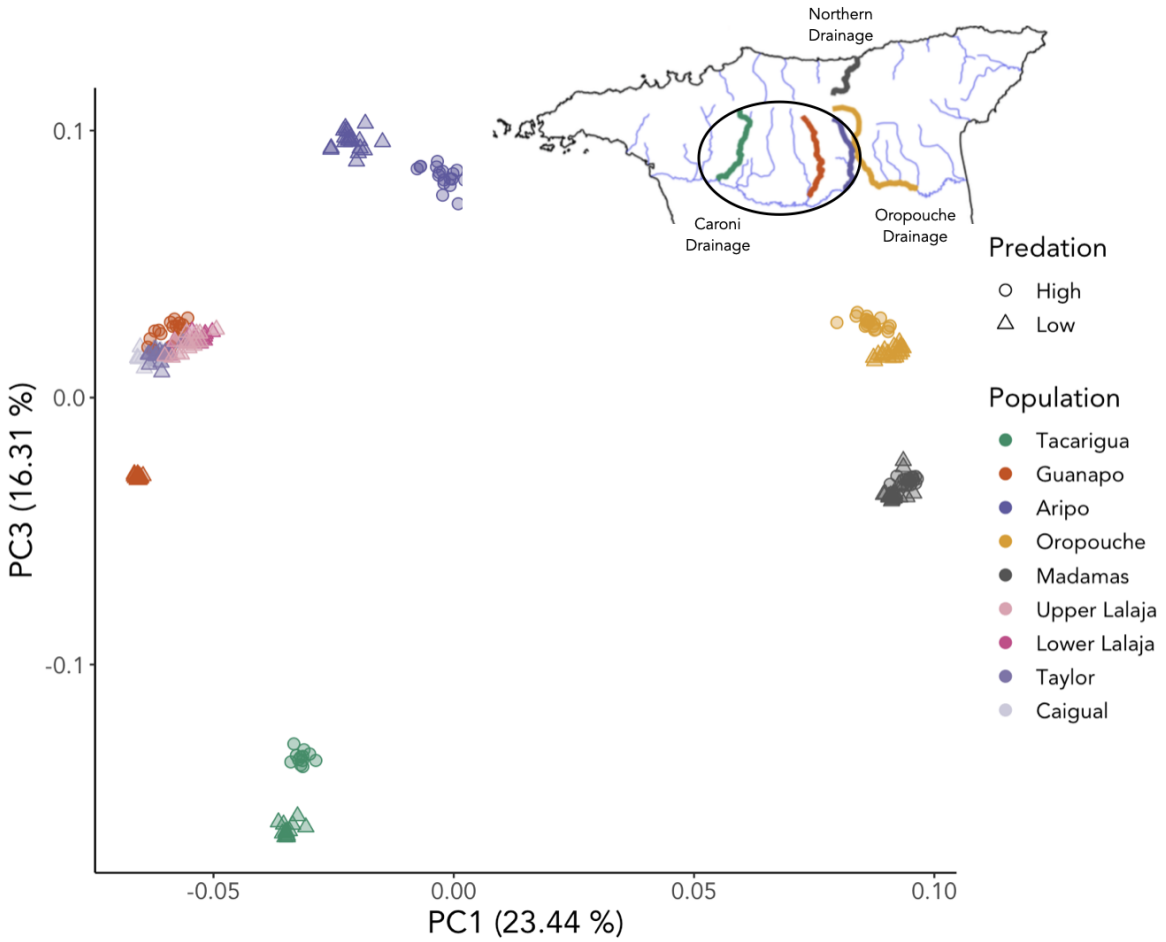
This command will create four files. We need two of these for plotting the PCA. One with the suffix `eigenvec` (eigenvectors) and another with the suffix `eigenval` (eigenvalues). These two files explain the covariance of how the variants from each individual sample are associated to one another. In short, the eigenvectors describe the direction and spread of our data (the Principal Components), and the eigenvalues describe the relative importance of each of these directions. If you want to know more, check out this great explanation.

10. We can now plot these data and see what our population structure looks like! Open the script plot_PCA_allpops.R in Rstudio and go through the plot lines one by one. Make sure you understand what each line of code is doing. Look at the plots directly in RStudio or save the plots to a PDF.

We'll start by looking at the plot for PC1 vs PC2. We've added the map of the rivers to help aid interpretation.




Let's have a think about what this PCA is telling us. We will start by thinking about the main PC axes separating the guppies from each drainage and river. PC1 (~23%) of the variance separates the Oropouche (Oropouche river) and Northern drainages (Madamas river) away from the Caroni drainage (Guanapo river, Aripo river and Tacarigua river). PC2 (~20% of the variance) separates the Guanapo river (along with the experimentally introduced populations) from the Aripo river and Tacarigua river. Looking at the geography of these rivers, wouldn't we expect the Aripo river and Tacarigua river guppies to also be separate from one another? … Let's look at the plot for PC1 vs PC3 …




Ah! Now we can see that PC3 (~16% of the variance) is the axis which separates the Aripo river and Tacarigua river from one another. This is only ~3% less variance than PC2. So the PCA nicely explains separation based on drainages and rivers. From this, we can also begin to hypothesise that the Oropouche drainage and Northern drainage are less diverged than the other drainages.

We can also say from this PCA that the experimentally introduced populations cluster closely with their source population (Guanapo high-predation (HP)). This is of course intuitive, given that these experimental low-predation (LP) populations were introduced in 2008 and 2009, and therefore there has been considerably less time for these populations to diverge from the source, and from one another. We will explore the experimentally introduced populations in more detail later.

For now, let's focus on the natural HP-LP pairs. Now let's think about the separation within each river between the high-predation (HP - circles) and low-predation (LP - triangles) populations.

11. Based on the PCA plots, which HP-LP pair are most divergent from one another?
12. Which HP-LP pair are the least divergent from one another?

Ok, let's explore this divergence another way! To do this, we will use $F_{ST}$, a commonly used statistic which describes the differentiation between populations. There are many ways to calculate $F_{ST}$ but the basic principle is the same. In its most basic form, $F_{ST}$ describes the proportion of the total genetic variance contained in a subpopulation (the s subscript) relative to the total genetic variance (the t subscript). Values can range from 0 to 1. High $F_{ST}$ implies a considerable degree of differentiation among populations.

13. Let's calculate $F_{ST}$ between each of the natural high-predation - low-predation pairs. We can do this easily using vcftools. The popmap files (.pop) contain a list (one individual per line) of the samples from each population. They can be found in the directory:
    `~/workshop_materials/pop_gen/lab_01/pop_structure_diversity/metadata`

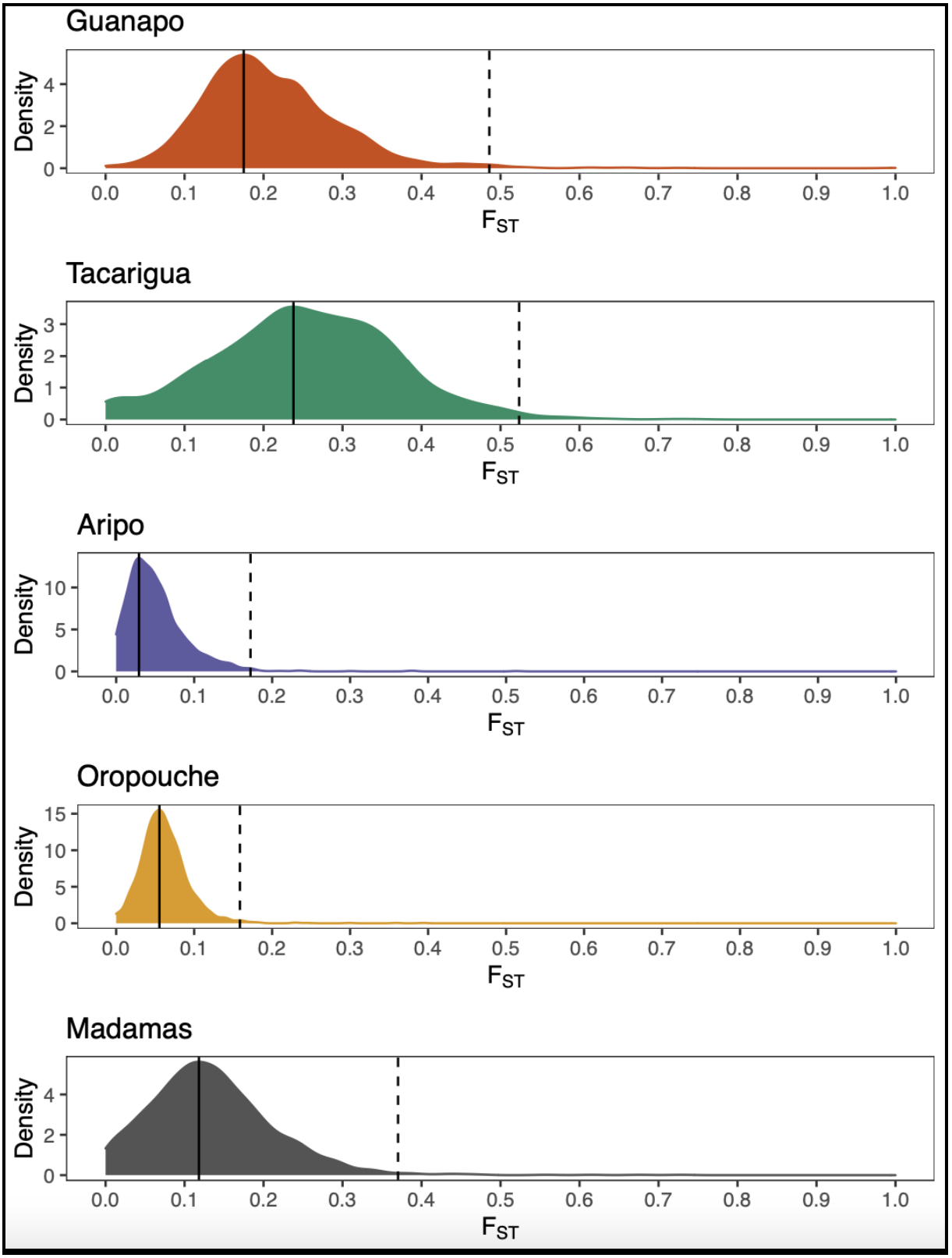
14. Let's start by calculating the pairwise $F_{ST}$ between Guanapo high-predation (HP) and Guanapo low-predation (LP).

```
vcftools --gzvcf guppy_natural_intros.vcf.gz \
--weir-fst-pop metadata/guanapo_high.pop \
--weir-fst-pop metadata/guanapo_low.pop \
--fst-window-size 500000 --out guanapo_fst_500kb
```

To save you some time, we've generated the data for the other 4 population pairs. You can find these data inside the folder:
`~/workshop_materials/pop_gen/lab_01/pop_structure_diversity/fst_data`

You can move (or copy) these into your current working directory
`~/workshop_materials/pop_gen/lab_01/pop_structure_diversity`

15. Now let's plot the pairwise $F_{ST}$ comparisons. Open the RScript plot_fst_distributions.R in RStudio. Run each of the populations through the R code. At the end, you should have a PDF file with the pairwise $F_{ST}$ densities between each population. Open this file and take a look!




16. Based on pairwise-$F_{ST}$, which river now shows the highest differentiation between high-predation (HP) and low-predation (LP)?
17. How do the plots of pairwise-$F_{ST}$ differ from what we saw in PCA space? Can you think of some reason(s) for the discrepancies?

There is actually an important difference between principal components analysis (PCA) and pairwise $F_{ST}$. Pairwise $F_{ST}$ values do not depend on what other samples are included in the analysis while principal components do. Although both representations of data reflect pairwise coalescence times, principal components depend on pairwise coalescence times for a particular pair of samples relative to other pairs of samples. We will discuss the coalescence in relation to demography in the second lab this evening.

In conclusion, the two ways of looking at data are both useful. Using pairwise $F_{ST}$ values allows a more direct tie to the underlying demographic processes. Moreover, the order of PCs will depend on the relative magnitudes of migration rate, genetic drift, geographical isolation and admixture between populations. For more information, take a look at this paper by Gil McVean.

And just because it's nice to see all of this in context...

Aripo river waterfall is on the left and Guanapo river waterfall is on the right. Do you think this geography explains the divergence we observed between the high-predation and low-predation populations sampled from these rivers?!

Take a break here if you like :)

## Part IIb

Now let's return to the experimentally introduced populations. In the PCA above we can't really see the PCA space for the introduction populations (Upper Lalaja, Lower Lalaja, Taylor and Caigual) as it's obscured by the higher divergence of the natural populations. To realise the PCA space for these populations, we need to make a new linkage-pruned dataset for a VCF file with only these individuals. Let's do that!

18. Return to the VCF file with the IDs already added in (`guppy_natural_intros_IDs.vcf.gz`). We're going to keep only those individuals from the experimental populations. Again, we're going to use a list of individuals we want to keep and we will use vcftools to keep only these individuals. Let's have a go!

```
vcftools --gzvcf guppy_natural_intros_IDs.vcf.gz \
--keep metadata/intro_pops.pop \
--recode --recode-INFO-all \
--out guppy_intros_IDs.vcf
```

At the end, this should print:
"After filtering, kept 94 out of 233 Individuals.
After filtering, kept 105417 out of a possible 105417 Sites"

Okay, all very good and dandy. But the removal of individuals from the VCF file will have led to some of the sites becoming monomorphic (otherwise known as "invariant") for either the reference or the alternative allele. Because PCA uses variant (polymorphic) sites, we should remove these non-informative sites. Let's do that!

We are going to run the same code as above, but this time let's partner vcftools and bcftools! Here, we will use bcftools view. We will pipe the output of vcftools directly to bcftools to keep only our individuals of interest and we will use bcftools to remove monomorphic sites ***at the same time***! We are showing you this to demonstrate the power of combining multiple tools using pipes. If you start working like this, it will save you lots of time in the future!

Ok so this time, we will ask vcftools to send the output to STDOUT and then use pipes to send this STDOUT directly to bcftools.

```
vcftools --gzvcf guppy_natural_intros_IDs.vcf.gz \
--keep metadata/intro_pops.pop \
--recode --recode-INFO-all --stdout | \
bcftools view -Oz -e 'COUNT(GT="AA")=N_SAMPLES || COUNT(GT="RR")=N_SAMPLES' \
-o guppy_intros_IDs.vcf.gz
```

***A little bit of explanation***: the -e option excludes sites for which an expression is true. So we build two expressions to exclude sites where the genotype column (GT) is homozygous for the alternative allele (AA) in all samples, or the reference allele (RR) in all samples. The -o command is for output. The -O is for output type and the -z specifies that we want the output format to be compressed.

This should print:

"After filtering, kept 94 out of 233 Individuals"

This gives us the VCF file we will work with for this part of the exercise.

19. Just to be certain, let's double-check that it's worked. Again, count how many individuals are now in the dataset:

```
bcftools query -l guppy_intros_IDs.vcf.gz | wc -l
```

This should be 94

20. And how many variants:

```
zgrep -v "^#" guppy_intros_IDs.vcf.gz | wc -l
```

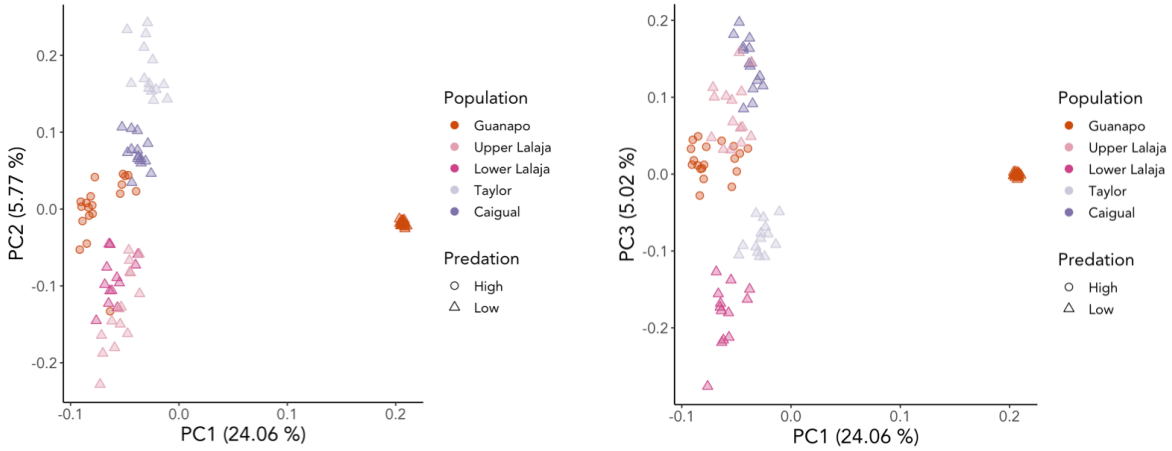
This should be 105417

21. Using this new VCF file, repeat the steps above to make linkage-pruned plink files, and then the files for PCA.

***HINT***: Set `VCF=guppy_intros` to save time! Then you can copy the commands above :)
***HINT***: Start from the code which is headed: # Prune for linkage

22. Plot the resulting eigenvectors and eigenvalues using the script named plot_PCA_intros.R

Cool! Now we can see the experimental introduction populations on their own and make some interpretations of what the PCA space is telling us.




Here, we can see that the natural low-predation population (Guanapo Low) explains the majority of the variance along PC1 (~24%), whereas the experimentally introduced populations cluster closely with the source. There is only ~6% variation explaining PC2 and ~5% variation explaining PC3.

23. What do you think the clustering of the Guanapo Low individuals tells us about this population in terms of diversity? What do you think happened to this population in it's demography?
24. Looking at PC1 vs PC2 and PC1 vs PC3, which selective processes do you think are driving the differentiation between the introduced populations and their source? We will explore selection in these populations in Part III

***Conclusion for part II***: Today, we've shown you how to run a PCA on your data. But we've also shown how PCA and $F_{ST}$ distributions can differ. In reality, you will run more than one analysis in order to explore your datasets' population structure. There's some more commonly-known methods of Structure and Admixture. But there's also some really cool new ways to do this which we haven't covered today. One is fineStructure, which uses the full haplotype information from phased variant data. There's also a complimentary program, fineRADstructure, which follows the same algorithms but is designed for RAD data (and also doesn't require a reference genome). Recently, another method called f4-struct is powerful in that it analyses population structure accounting for drift using f4-statistics. As for population genetics statistics, vcftools works just fine (sometimes!) but we recommend either PopGenome or pixy for pi and Dxy. These are just a few ideas for you :)

# Part III

We're now going to touch on a little selection! We're going to perform some genome scans for convergent evolution among the four experimentally introduced low-predation populations. Move into the directory:
`~/workshop_materials/pop_gen/lab_01/selection_scans`

You will see five VCF files, one for each of the introduction populations, plus one for the Guanapo high-predation source. These VCF files however are slightly different from the ones you have been using previously as they are *phased*. What does this mean?

Phasing is an essential and frequently used process in population genetic analyses. Mike covered the ideas behind phasing in week 1. Briefly, for each variant found in the VCF file, we can report which chromosome pair each allele comes from. Allele phasing information can be incredibly useful for us as it means rather than using the information from each variant in isolation, we can use the full phased haplotype or haplotype blocks. With traditional short-read Illumina WGS data (as we are using here), phasing is performed algorithmically, using software such as Beagle and/or Shapeit. Other ways of phasing include sequencing trios, or with the current technology used in long-read sequencing, you can get the full haplotype straight off the sequencer! Read PacBio's blog on this here for more information.

1. Phased VCF files contain this phase information. Take a look at the first few lines of one of the VCF files. Do you see how the genotypes are encoded slightly differently?
2. We're going to use this phase information to perform a selection scan later. For now, let's start with a more fundamental method for detecting selection: nucleotide diversity ($\pi$), which is the average number of nucleotide differences per site between two DNA sequences. Let's calculate $\pi$ for each of the populations.
3. First, we need to create the popmap files (as used previously in Part II). This time, we will make them ourselves. Create a directory called `popmaps`. Then, using bcftools query make a popmap file for each of the populations, e.g.:

```
## Get popmaps
bcftools query -l guanaposource_phased.vcf.gz > popmaps/guanaposource.pop
bcftools query -l upperlalaja_phased.vcf.gz > popmaps/upperlalaja.pop
```

```
bcftools query -l lowerlalaja_phased.vcf.gz > popmaps/lowerlalaja.pop
bcftools query -l caigual_phased.vcf.gz > popmaps/caigual.pop
bcftools query -l taylor_phased.vcf.gz > popmaps/taylor.pop
```

4. Now we will calculate nucleotide diversity (π) in windows of 50000 base pairs (50kb) for each introduced population and the source population. We will use vcftools again. But rather than running this one time for each population like above, we can put this in a for-loop to speed things up!

```
for pop in caigual lowerlalaja upperlalaja taylor guanaposource;
do;
vcftools --gzvcf ${pop}_phased.vcf.gz \
--keep popmaps/${pop}.pop \
--window-pi 50000 \
--out ${pop}_pi_50kb;
done
```

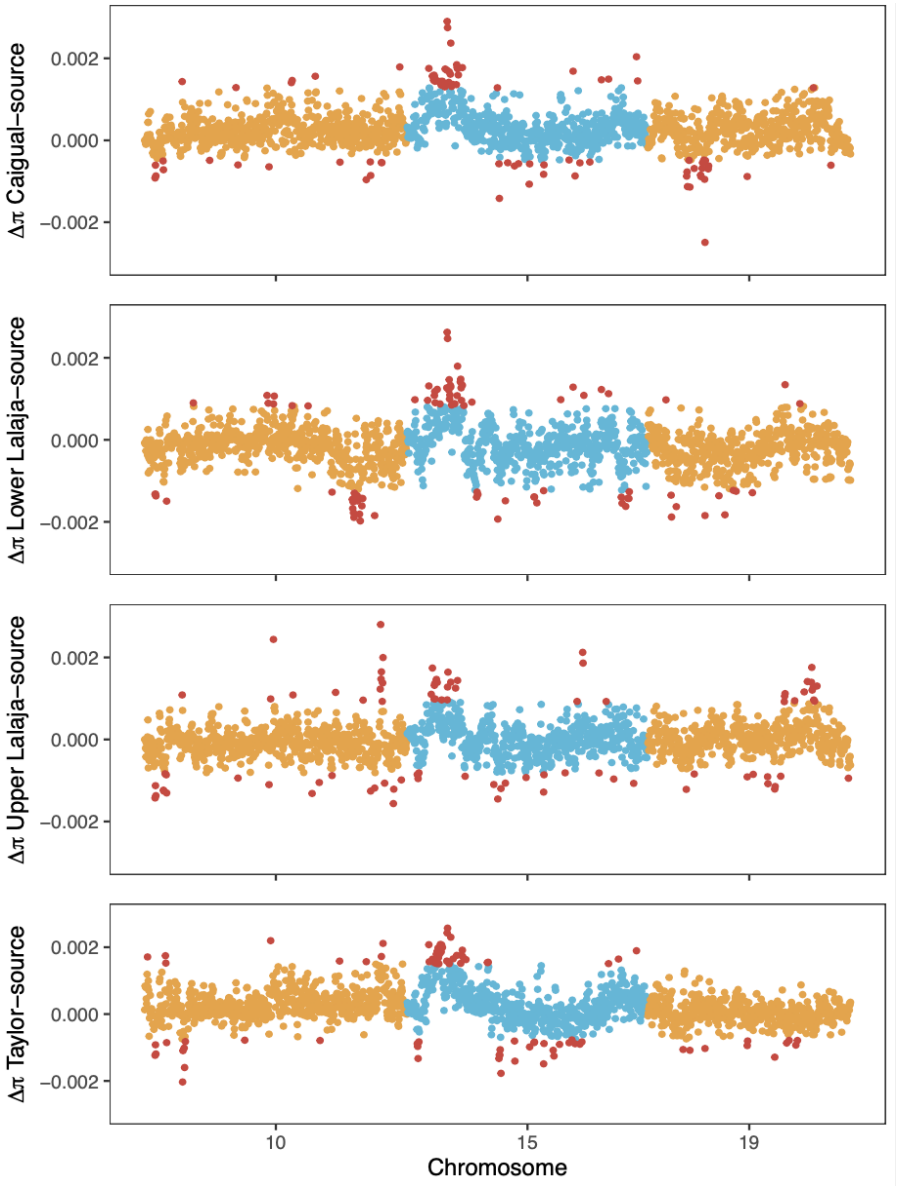
How neat is that!

5. Now that we have π for each of the populations, we are going to calculate Delta pi (Δπ) between each of the introduced populations and the Guanapo source population. This is calculated by subtracting π of the introduced population from π of the source population:

$$\Delta\pi = \text{source } \pi - \text{introduction } \pi$$

We are interested in regions of the genome where the introduced populations have lower π compared to the source because this is indicative of selection in the form of a selective sweep from standing genetic variation (diversity present as π in the source population).

6. Open the RScript guppy_deltapi_genome_scans.R. Go through the script line-by-line and run the code. You will see in the script where we calculate Δπ. Let's take a look!




You'll see that we have calculated and plotted Δπ for three chromosomes (10,15,19). Looking across these chromosomes, do you see any consistent peaks indicative of convergent selection to low-predation environments?

The chromosome 15 peak is showing reduced nucleotide diversity (π) in all four of the introduced populations relative to the Guanapo source population. But, these differences in Δπ are performed on a per-variant level. What happens if we include the whole haplotype?

We are going to use a statistic called XP-EHH from the selscan package. XP-EHH is part of a family of haplotype statistics which are frequently used to detect positive selection in the form of soft sweeps. Check out this paper by Sabeti and colleagues for more information.

7. We're going to calculate XP-EHH for the introduced guppy populations. We first need to prepare the files for selscan. selscan runs on a per-chromosome basis, and requires a map file for each of these chromosomes. The map files are a tab-separated file for each chromosome which look like this:

```
chr10   chr10-12427     12427   12427
chr10   chr10-12434     12434   12434
chr10   chr10-12443     12443   12443
chr10   chr10-12449     12449   12449
chr10   chr10-12458     12458   12458
chr10   chr10-12466     12466   12466
chr10   chr10-12843     12843   12843
chr10   chr10-12967     12967   12967
chr10   chr10-12971     12971   12971
chr10   chr10-12984     12984   12984
```

The first column is the name of the chromosome, the second column is the locus ID (in this case name of the chromosome followed by the variant position, separated by a -), the third column is the genetic position and the fourth column is the physical position. In cases where you have a genetic map for your organism, the fourth column will contain the physical position (in cM). For simplicity, we will use the physical position in this exercise.

8. Let's make the map files for chromosomes 10, 15 and 19. Again, we will do this in a loop, this time iterating over these three chromosome names (chr10, chr15 and chr19), by making them variables, and then using a bit of UNIX (awk again!) to format the files correctly. All the VCF files have the same variants so we can just use the Guanapo source population VCF file for this.

```
## Get selscan mapfiles
for chr in chr10 chr15 chr19;
do
zcat guanaposource_phased.vcf.gz | grep -v "^#" | \
grep $chr | cut -f1-2 | \
awk '{print $1 "\t" $1"-"$2 "\t" $2 "\t" $2}' > $chr\_selscan.map;
done
```

9. Now let's make a VCF file for each of these chromosomes using vcftools. Again, we will do this in a loop. Read over these lines to make sure you understand what each part of the for loop is doing!

```
## Get selscan VCF files per chromosome
for chr in chr10 chr15 chr19;
do
vcftools --gzvcf guanaposource_phased.vcf.gz --chr ${chr} \
--remove-filtered-all --recode \
--stdout | gzip -c > guanaposource_phased_${chr}.vcf.gz;
vcftools --gzvcf upperlalaja_phased.vcf.gz --chr ${chr} \
--remove-filtered-all --recode \
--stdout | gzip -c > upperlalaja_phased_${chr}.vcf.gz;
vcftools --gzvcf lowerlalaja_phased.vcf.gz --chr ${chr} \
--remove-filtered-all --recode \
--stdout | gzip -c > lowerlalaja_phased_${chr}.vcf.gz;
vcftools --gzvcf caigual_phased.vcf.gz --chr ${chr} \
```

```
--remove-filtered-all --recode \
--stdout | gzip -c > caigual_phased_${chr}.vcf.gz;
vcftools --gzvcf taylor_phased.vcf.gz --chr ${chr} \
--remove-filtered-all --recode --stdout | gzip -c >
taylor_phased_${chr}.vcf.gz; done
```

10. Now we can run selscan on each of the VCF files. Again, we can do this in a loop:

***NB***: This will take approximately 15 minutes. If you're running out of time you can stop the command. We have generated the data already (see below at step 11).

```
for pop in caigual lowerlalaja upperlalaja taylor;
do
selscan --xpehh --vcf ${pop}_phased_chr15.vcf.gz \
--vcf-ref guanaposource_phased_chr15.vcf.gz \
--map chr15_selscan.map \
--out chr15_selscan_${pop}-source.win50 --threads 4;
done
```

Once this is running, it will print the following:

"selscan v2.0.0
Opening caigual_phased_chr15.vcf.gz...
Loading 30 haplotypes and 267918 loci...
Opening guanaposource_phased_chr15.vcf.gz...
Loading 38 haplotypes and 267918 loci...
Opening chr15_selscan.map...
Loading map data for 267918 loci
Starting XP-EHH calculations."

***HINT***: As an aside, you could even do a nested for loop to run selscan on all populations and all chromosomes:

```
for pop in caigual lowerlalaja upperlalaja taylor;
do
for chr in chr10 chr15 chr19;
do
selscan --xpehh --vcf ${pop}_phased_${chr}.vcf.gz \
--vcf-ref guanaposource_phased_${chr}.vcf.gz \
--map ${chr}_selscan.map \
--out ${chr}_selscan_${pop}-source.win50 --threads 4;
done;
done
```

See how powerful variables and for loops can be! If you begin to use a computer cluster, understanding these will be useful to help you speed up your code ;)

11. We have already pre-generated the data for chromosome 15. You can find it in the folder `data`
12. Haplotype homozygosity statistics are affected by the frequency of the SNPs within each core haplotype. Therefore, once you have calculated XP-EHH, you then need to normalise for this variation in SNP density in the genome. We can do this using the norm program from selscan. We will normalise in 50kb windows, using a critical percentage of 0.05. The critical percentage is set in order to mark SNPs in the extreme tails (two-tailed) as extreme SNPs. This is because selective sweeps tend to produce clusters of extreme scores across the sweep region, while under a neutral model, extreme scores are scattered more uniformly. Again, let's run this in a loop:

```
for pop in caigual lowerlalaja upperlalaja taylor;
do
norm --xpehh --crit-percent 0.05 --winsize 50000 \
--files chr15_selscan_${pop}-source.win50.xpehh.out;
done
```

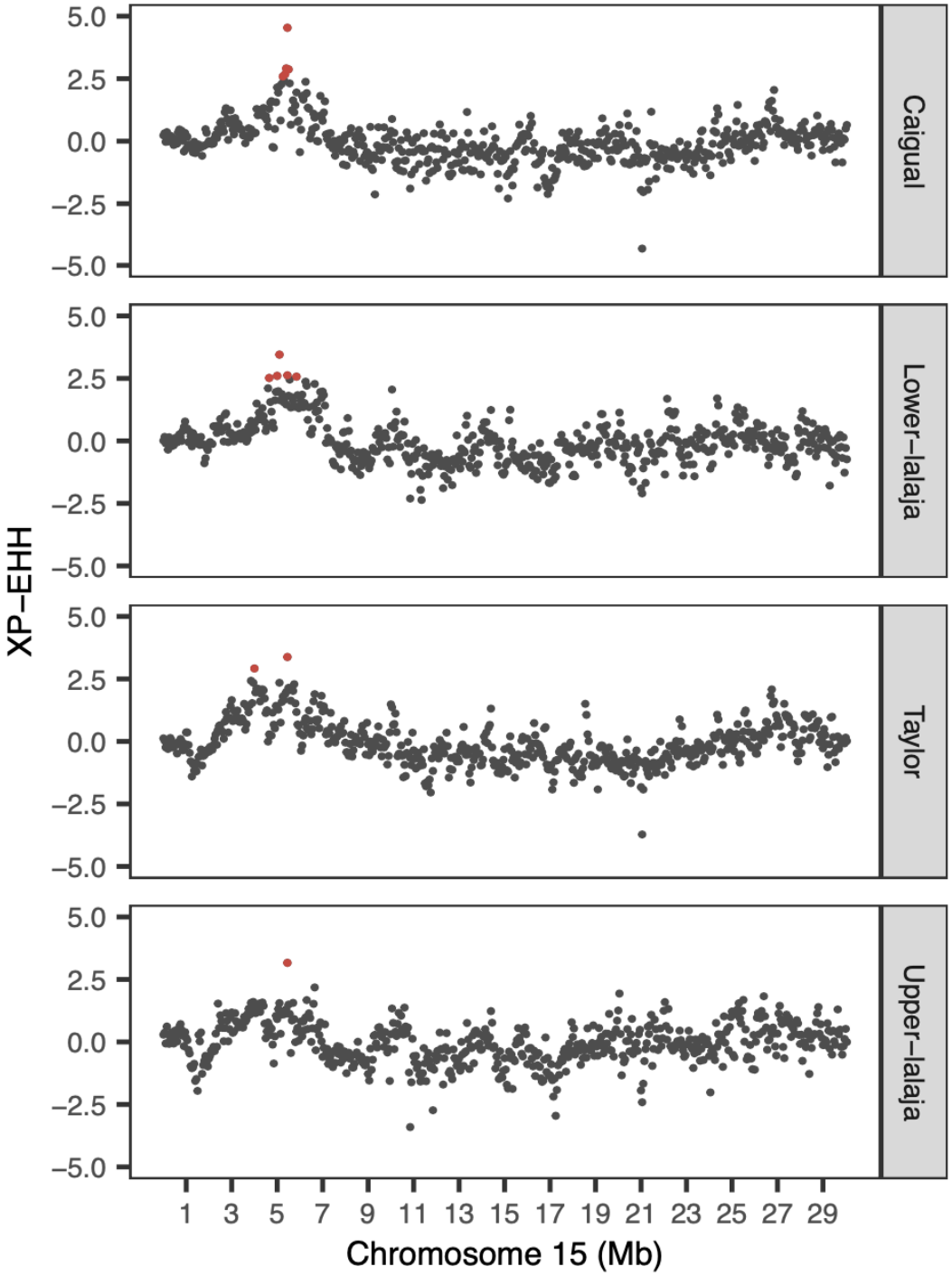
13. Have a look at the output files. We are interested in the final column with the header "normxpehh". However, these outputs give us a normalised haplotype homozygosity score per variant, and we're interested in seeing if XP-EHH scores are elevated along the chromosome in windows. To do explore this, we will "windowise" our data. This is similar to the window function we used above in vcftools. The script you need to window our data is called xpehh_window_estimation.R. It contains a windowing function (written in R) to perform this for us for each of our pairwise comparisons. This script will also identify outlier windows in chromosome 15 (above a normalised XP-EHH score of 2.5). Take a look at the script if you like (it's a bit complex!), but if you can also just run it directly from the command line:

```
Rscript xpehh_window_estimation.R
```

This will output the following:

- XP-EHH windows for each introduction population, named with the suffix: `win50.xpehh_windows.txt`
- A file with the XP-EHH scores in windows across chromosome 15 from all populations together called `guppy_intros_xpehh_all.txt`
- XP-EHH outlier windows for each introduction population, named with the suffix: `_outliers_XP2.5.txt`
- A file with all the XP-EHH outlier scores in windows across chromosome 15 from all populations together called guppy_intros_xpehh_all_outliers.txt

14. Finally, let's plot our outputs! Open the Rscript xpehh_window_plots.R in RStudio. Run the code and generate the plot.




Now we can see that XP-EHH also identifies a significant peak on chromosome 15 in all four of the introduced populations!!

***Conclusion to Part III*:** There are many ways to scan genomes for selection. The tools you will use depend on your questions and knowledge of the system you are investigating. For directional selection, the statistics FST, Dxy and Δπ can be used. There's also a variety of more complex methods for identifying signatures of selection, such as XP-EHH we have used today. For balancing selection, we're fans of Ballet and Ballermix :) Controlling for false positives, and other confounding issues, e.g. demography (which we will cover in the next

workshop), linkage disequilibrium, recombination landscape has always been a sticking point, see Ahren and colleagues' review. When you find an interesting region, dig deeper, plot the region and investigate it in many ways! In the next workshop, we will look at how you can explore the demography of your populations. See you after dinner!