**Genome size differences are underpinned by structural variation in various copy-number classes as revealed by k-mers**

**Methods (Draft)**

**Input data**

* The required input data are short read sequences from different *Euphrasia* species. Raw sequences can be trimmed or untrimmed (May work better with trimmed?).
* The pipeline is designed to compare *Euphrasia* kmer dumps within and between species. The working directory for the script should contain a set of sub-directories, where each sub-directory corresponds to a single *Euphrasia* population. The sub-directory names should indicate the ID of each population. Each directory should contain a text file called *file\_names*, where the path to the raw sequence data files in FASTQ format (gziped or not) are listed.

For instance, in the sub-directory “arctica\_E026”, the file\_names text file contains the following list:

$ cat arctica\_E026/file\_names

/disk2/hbecher\_tmp/trimmed/a2j-trimmed-pair1.fastq.gz /disk2/hbecher\_tmp/trimmed/a2j-trimmed-pair2.fastq.gz /disk2/hbecher\_tmp/trimmed/a2m-trimmed-pair1.fastq.gz /disk2/hbecher\_tmp/trimmed/a2m-trimmed-pair2.fastq.gz

* The working directory should also contain the *plastid\_genome* and *mito\_genome* directories which contain the assemblies and *kmc* kmer datasets for the *Euphrasia* plastid and mitochondrial genome. These directories contain the required mito\_assembly.kmc\* and plastid\_kmer.kmc\* kmer databases required to extract the nuclear genome kmers.
* The pipeline is designed to delete files created in intermediate steps to use storage space efficiently.

**Step 1 – Generation of kmer multiplicity histograms for overview of kmer database and genome size**

* The bash script *kmc\_script\_histograms\_from\_read.sh* will generate three kmer multiplicity histograms for each *Euphrasia* population specified.
  + A histogram of the raw kmer database.
  + A histogram of the kmer database with the plastid genome removed.
  + A histogram of the kmer database with the mitochondrial and plastid genome removed (which should roughly correspond to the nuclear genome kmer database)
* To specify for which populations the histograms will be generated, the “POPULATIONS” variable in the bash script “kmc\_script\_histogram\_from\_reads.sh” must be manually edited. The variable should contain a list of *Euphrasia* population IDs which match the directory names which have been created for each population.
* Once the populations have been specified, run the script *kmc\_script\_histogram\_from\_reads.sh*

**Step 2 – Histogram analysis to determine haploid coverage, genome size and contamination threshold.**

The purpose of creating multiplicity histograms for each of our populations of interest is to determine three parameters which will be useful for subsequent analysis. Our multiplicity histogram can be plotted on R by running the import\_histogram.R script. This script will generate a simple line plot where “Coverage” is represented on the x-axis and “Frequency” is on the y-axis. The x-axis maximum limit is 100. One or more peaks should be visible. To understand this plot, we must understand *Euphrasia’s* ploidy.

*Euphrasia* can be either diploid or tetraploid. A diploid genome contains two genome copies (2n) and a tetraploid contains four genome copies (4n). A kmer which only appears in one genome copy will be present *n* times. The parameter *n*, or haploid coverage, will be determined by the sequencing depth. The peaks on the graph will generally correspond to the 2n and/or 4n coverage peaks. See Becher et al. (2020) for a more detailed understanding of these plots.

To determine the haploid coverage, we can roughly estimate it by finding the 2n peak, and dividing by two. However, this is not completely accurate. A better estimate can be achieved by using the R script Tetmer (Becher, 2020).

I estimated the haploid coverage by using the auto-fitting mode on Tetmer and adjusting the slider to remove the low multiplicity sequences on the x-axis and reducing the parameter ranges for fitting below what we believe to be the 2n peak. A few more modifications to the sliders may be required to fit the data to the predicted spectrum. **It is important to be patient and achieve the best fit possible as the haploid coverage will be essential for steps 4 and 5.**

When visualising the kmer multiplicity curve, it is likely for the lowest values of x, the frequency will be considerably high, and this will be followed by a steep drop until the next peak. These coverage values will be below the haploid coverage and therefore, are unlikely to be biologically relevant. We should discard these values for all subsequent calculations. **It is important to take note of the value of x, at which this curve is no longer significantly high.**  This value must always be below the haploid coverage, and should not be above around 0.7n, to avoid removing biologically relevant kmers.

Once we have estimated the haploid coverage, we can estimate the polyploid genome size for our population of interest. To do this, we can run the genome\_size\_calculation.R script. We will need to edit the script to adjust the haploid\_coverage and the low multiplicity kmer threshold. The script will subsequently return the polyploid genome size in Mb. To calculate the haploid genome size, we can divide the result by four.

We can also estimate the genome size by using GenomeScope 2.0 (Ranallo-Benavidez, Jaron and Schatz, 2020). I uploaded the multiplicity histogram file to their online web application (<http://qb.cshl.edu/genomescope/genomescope2.0/>). It is important to adjust the kmer length to 27, the ploidy to the correct value, the max kmer coverage to 1.5\*107 and the average kmer coverage for the polyploid genome to our estimate of the haploid coverage.

**Step 3 – Join kmer dumps of populations to be compared**

The next step involves joining a set of kmer dump files (a list of kmer sequences and their respective multiplicities) from different populations, which we are interested in comparing. In principle, there is no limit to how many kmer dumps can be joined together. However, in practice, dump files are extremely large (> 20 GB), and therefore, I have limited the number of different populations’ dumps in a single file to five.

Sometimes, it can be useful to only join two kmer dumps together and carry out direct pairwise comparisons. Alternatively, if you are interested in comparing a whole range of populations (for instance, a set of populations from the same species), it can be more efficient to create a single joined kmer dump file and select the relevant columns for subsequent analyses.

When generating a kmer dump, I find it is useful to discard low multiplicity kmers which are below a certain threshold (always lower than the haploid coverage) and therefore not likely to be biologically relevant.

To generate two kmer dumps and join them together, we can run the joined\_dump.sh script from our working directory. The working directory must contain the two populations we are interested in (in their respective sub-directories) as well as the Euphrasia chloroplast and mitochondrial assembly. The population directories must contain the paths to the corresponding raw sequencing data under *file\_names* (as described above)

To run this script, we must enter the following variables:

bash joined\_dump.sh <population 1> <low multiplicity threshold 1> <population 2> <low multiplicity threshold 2> <output file>

Where population 1 and 2 are the names of the directories of our populations of interest, the low multiplicity threshold is the multiplicity limit below which kmers will be discarded and the output file is the name of the joined kmer dump.

To join more than two dumps together, the required dumps must be generated by running the dump.sh script

bash dump.sh <population ID> <low multiplicity threshold> <output file>

Where population ID is the name of the directory containing file\_names for the population of interest, the low multiplicity threshold is the multiplicity limit below which kmers will be discarded and the output file is the name of the kmer dump.

After running dump.sh for the populations we are interested in, we must join the dump files one at a time. This can be done by running the join.sh script.

join.sh <input file 1> <input file 2> <output file>

Where input file 1 and 2 are the paths to the dumps we are interested in joining and the output file is the name of the joined dump. If we were joining four dumps (A, B, C and D) we can firstly join A and B to create A\_B and subsequently add A\_B to C and then A\_B\_C to D. The final product would be A\_B\_C\_D.

**Step 4 – Generation of pairwise comparison plots between populations’ kmer frequency differences**

* The bash script *kmer\_cov\_difference\_plots.sh* is designed to create plots based on R between kmer dump datasets from two different *Euphrasia* populations to observe the differences in kmer coverage for each different kmer sequence.
* The script is designed to analyse a joined kmer dump from a set of different populations.
* The script must be manually edited to specify the kmer\_dump input file, the name of reference population and the comparison population and the columns which correspond to their coverage information. Additionally, the previously estimated haploid coverage for each population must be specified.