MHC1 Analysis in Hawaiian Green Turtles with DADA2- trimmed primers and conservative filtering

Jamie Adkins Stoll

11/25/2020

Background

This document is similar to the previous document "dada2HIGT_analysis.pdf", but I increased the stringency of the filtering max error rate to be 0.1, I completely removed primer sequences (inlcuding degenerate nucleotides) AND I filtered out final variants on a per-sample basis to remove variants that represented less than a certain percentage of total reads for that sample. These filtering steps reduced the total number of variants from 112 to 17.

This workflow is used to analyze MHC sequencing data generated using a targeted amplicon sequencing library protocol from the Komoroske lab, sequenced on an Illumina Miseq platform with paired end sequencing. More information about dada2 can be found in this tutorial

The MHC (major histocompatibility complex) is a highly variable region where we expect high sequence diversity between individuals. This workflow uses the Dada2 package to identify unique sequence variants in indivual samples, and outputs an amplicon sequence variant table that records the number of times each amplicon sequence variant is observed in each sample.

- The input files for this workflow are the demultiplexed fastq files from your sequencing project, with non-biological nucletodies removed (e.g. primers, adapters), and forward and reverse reads in matched order.
- Prior to input in this workflow, I used demultiplex reads using a custom script, and sickle and scythe packages for adaptor trimming on the command line.
- The data here were generated from genomic DNA extracted skin bioposies of Hawaiian green sea turtles.
- This R project is located in my local github repository, "MHC_analysis"

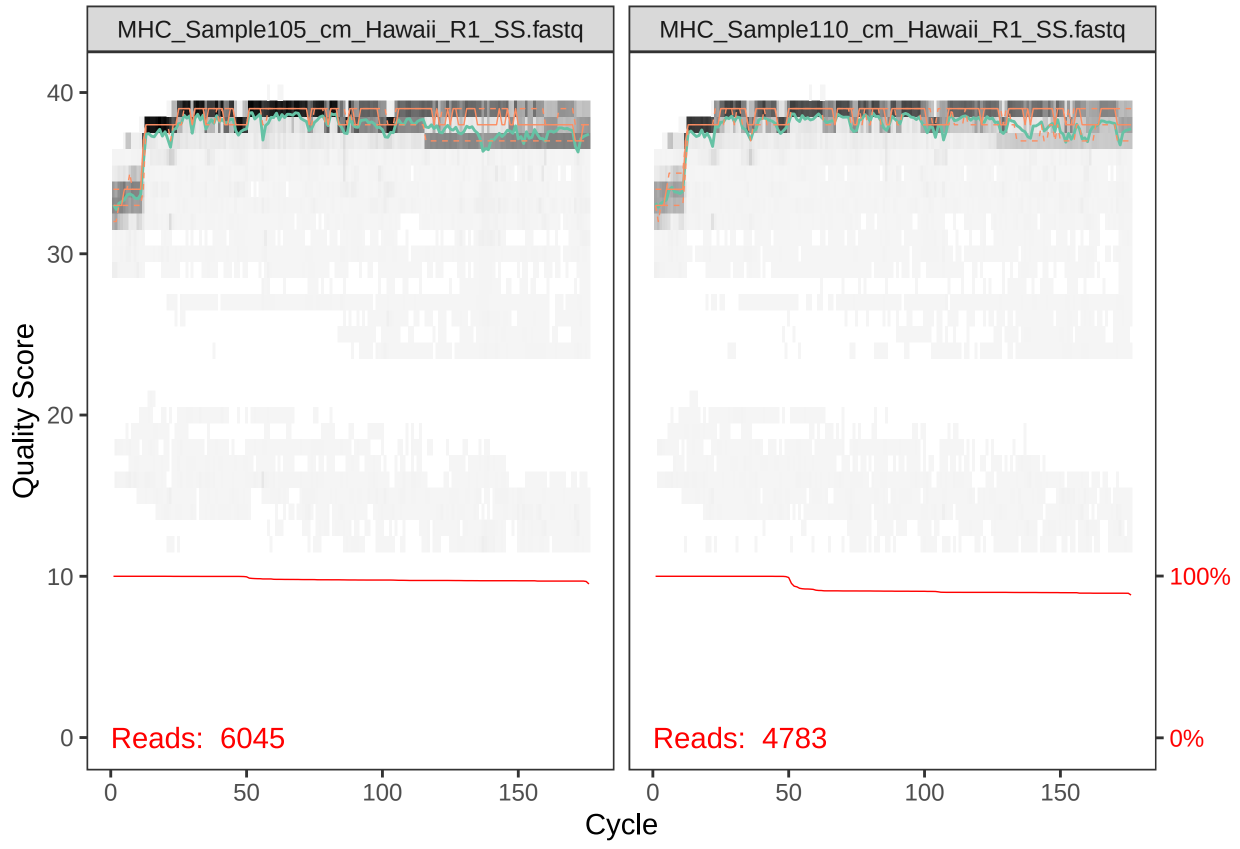
Workflow and Results

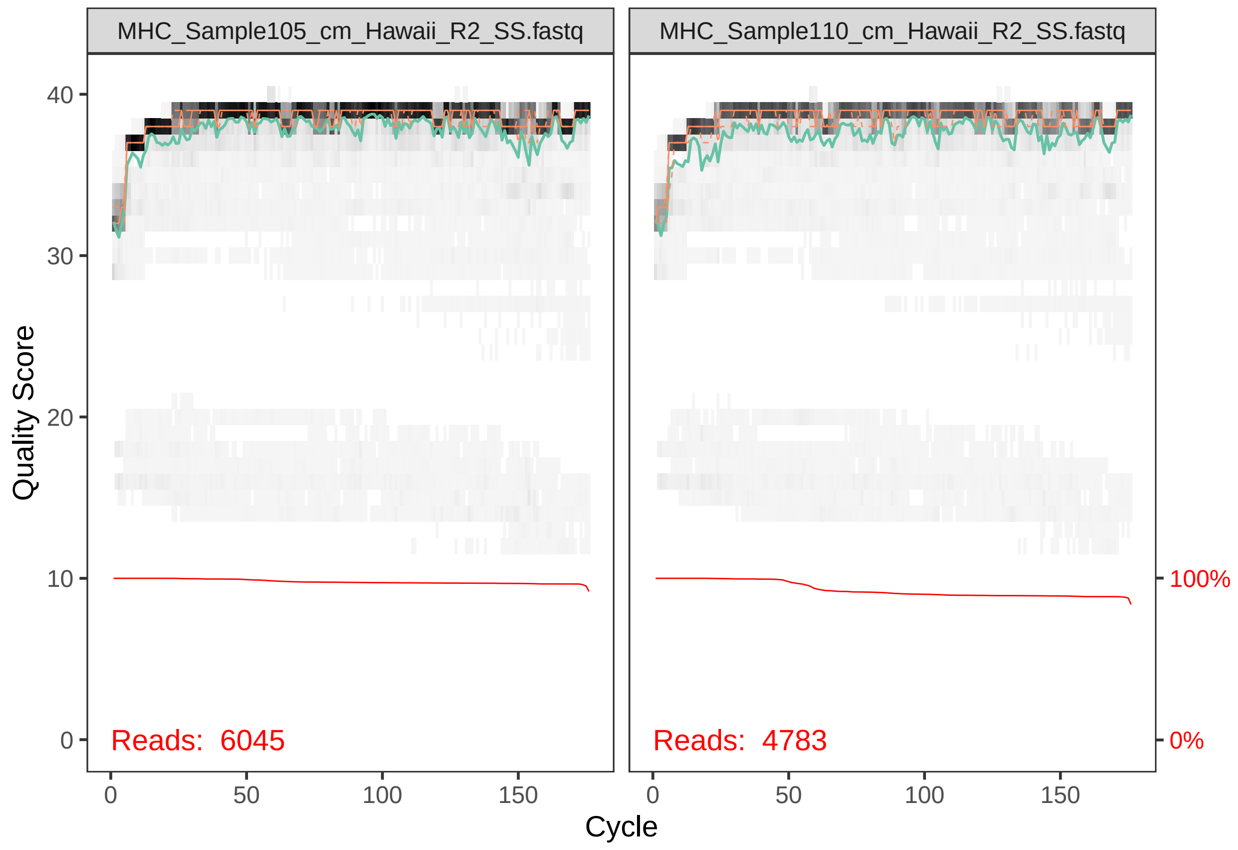
Filtering step overview: 1. Filter out reads with with greater than 0.1 error rate 2. Trim off 15 from right end of reads to remove primer sequences 3. Filter out variants deviating from expected length 4. Remove chimeric sequences 5. Remove variants present in only one individual 6. Remove variants with less than 100 reads supporting it across individuals 7. Within individuals, remove variants that are supported by less than 5% of reads in that individual

1.Read in fastq files and examine quality of forward and reverse sequences.

First we inspect the quality scores of our input fastq files, and determine what needs to be trimmed or filtered out prior to using reads for analysis.

- The first plot displays the sequence quality profile for a single sample, for the forward reads. The average base quality score (depicted as green line) shows that sequence quality on average is high for this sample.
- The second plot displays the sequence quality profile for the sample sample, for the reverse reads. Again, average base quality score is high. I expected this, given that I had previously trimmed my sequences for adaptors and quality, but here I am confirming this worked well.





2. Filter reads

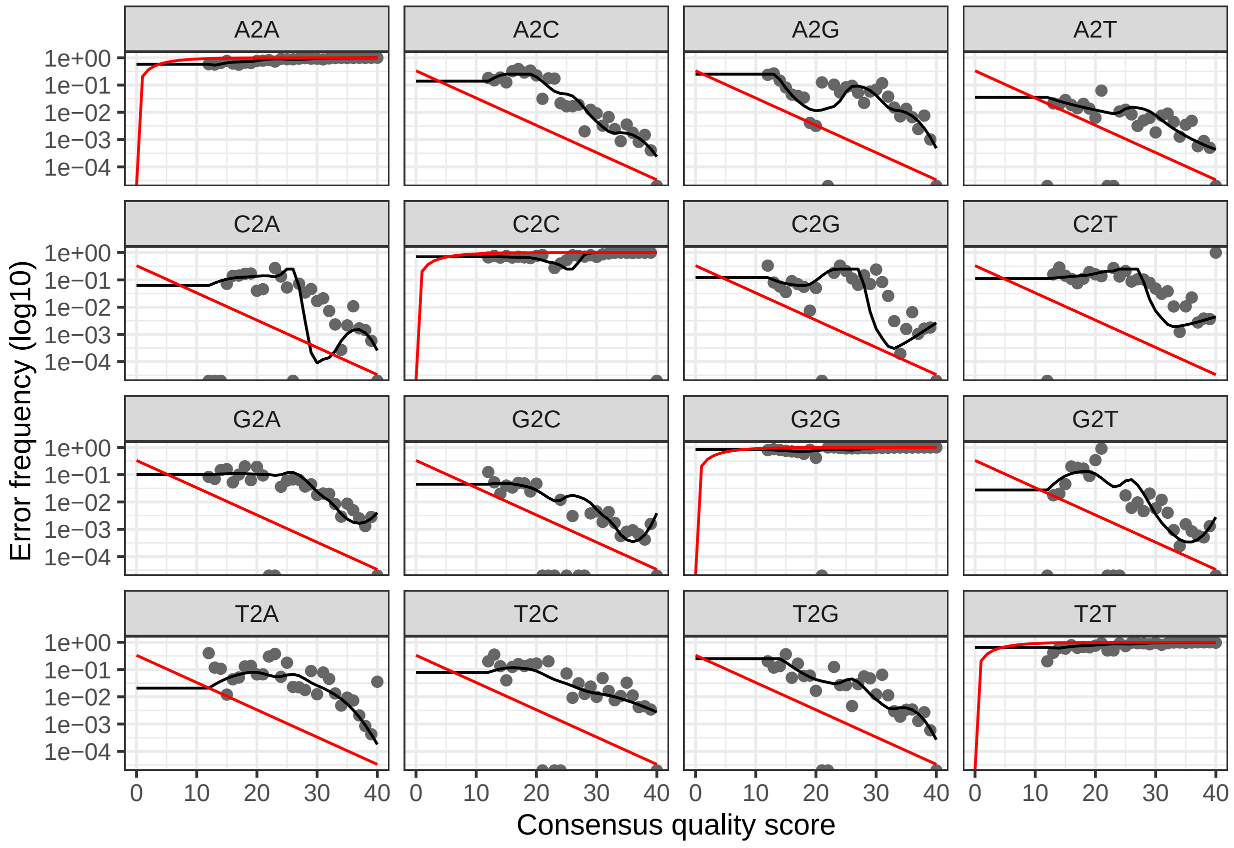
After inspecting the read quality above, we need to filter out reads not meeting a specified quality threshold, as these would introduce bias into called sequence variants downstream.

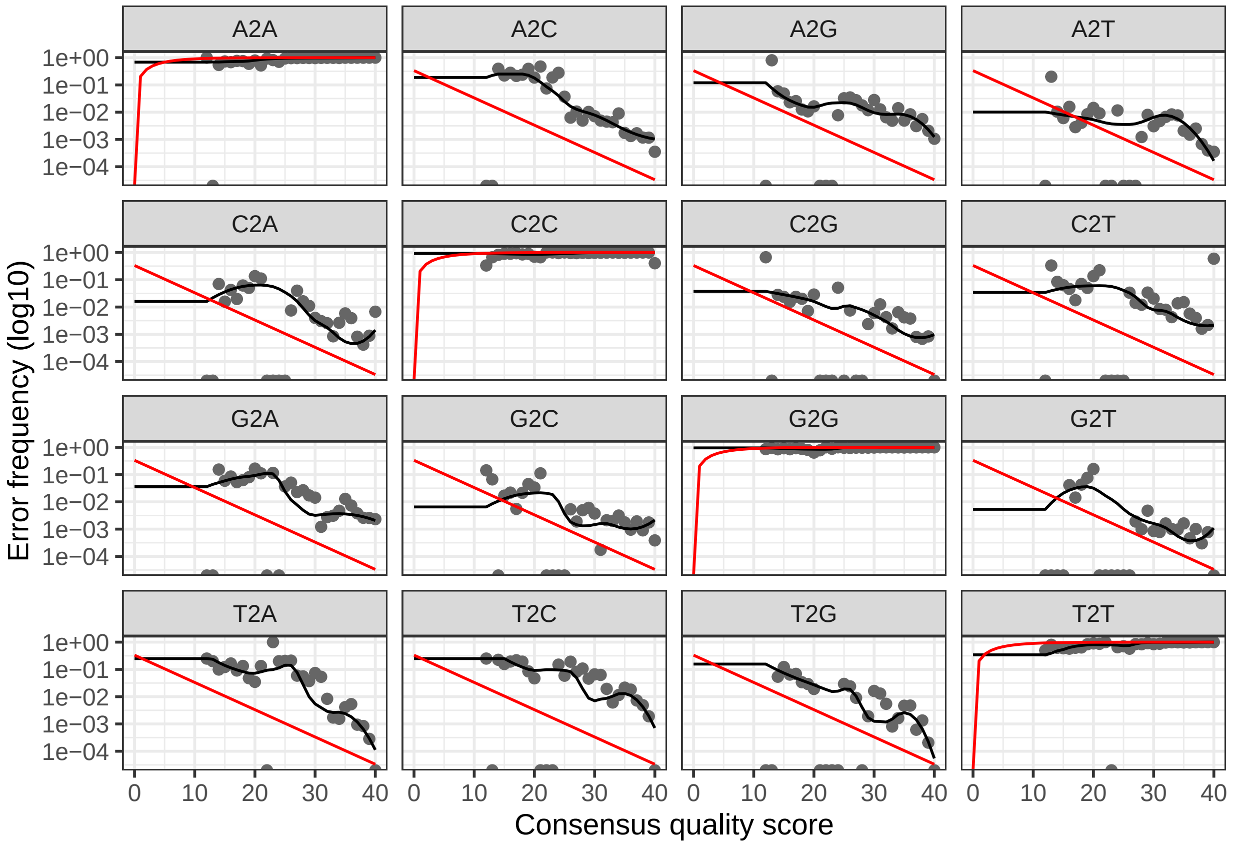
- I initially didn't anticipate the need to further filter the reads prior to the next steps since I had already trimmed based on quality, but I found that some sequences still contained "N" bases, and these sequences cannot be used in this program and throw errors.
- I further filtered reads, setting the maximum number of 'N' bases to 0, and maxEE to 0.1 for both the forward and reverse reads. The maxEE parameter sets the maximum number of expected errors allowed in a read, and this value is very stringent.
- I also set the truncLen parameter to 0 for both forward and reverse reads, which indicates that I do NOT want to truncate my sequences by removing low quality tails. I did not need to truncate reads based on the quality profiles above.
- Finally, I trimmed off the right side of reads to remove 15bp of remaining primer sequence. These sequences were driving sequence variation in previous analysis, and we decided to remove them even though the reverse primer contains two degenerate bases that could contain true biological variation.

After filtering, the DADA2 algorithm uses a parametric error model for every sample to learn error rates. The error rates for each possible base transition are plotted below.

The important things to note before moving forward are: * the estimated error rates (black line) are a good fit to the observed rates (points) * the error rates drop with increased quality

```
## 21651511 total bases in 148308 reads from 35 samples will be used for learning the error rates.
## 21852332 total bases in 148308 reads from 35 samples will be used for learning the error rates.
```





Given that these points are true for my dataset, and that these plots look very similar to those in the dada2 pipeline tutorial, we can move forward to sample inference.

3. Sample inference

Now that we have filtered and trimmed sample reads, and learned error rates, we will use the primary dada2 algorithm for sample inference. Complete details of the algorithm can be found in Callhan et al. 2016. In brief, this alogrithm models and corrects for Illumina-sequenced amplicon errors, and infers sample sequences, resolving differences of as little as 1 nucleotide.

i. Generate a dada-class object First, we generate a dada-class object, which holds unique sequence variants for forward and reverse reads.

ii. Merge reads After creating the dadaclass object that holds the forward and reverse reads, we merge forward and reverse reads to get a full set of denoised samples.

- Parameters can be changed upstream if you find that you are losing too many reads at this merging step, particularly focusing on the truncate length parameter.
- The “mergers” table displays the sequence variants per sample and abundance of each sequence in a given sample.

I found that most reads are merging successfully, though often I was losing around 800-1000 reads during the merge, and there is one sample where only 254 sequences successfully merged.

iii. Construct an amplicon sequence variant table From the merged reads, we construct a sequence variant table that contains all sequence variants for the entire dataset, and the number of variant copies per sample.

There are 35 samples and a total of 318 sequence variants at this point. These sequence variants will be filtered further in the following steps.

3. Filter Called Sequence Variants

i. Filter by length The first filter of sequence variants from the total dataset is by length. First, inspect the length distribution of sequence variants. Sequence variants significantly deviating from the mean or expected sequence length in our study should be removed. Sequences deviating from the expected length significantly are like due to sequencing error or off-target amplification. The exception to this rule is if you were barcoding ITS amplicon seqeces (this does not apply here).

```
##
##  75  76  85  86  96 100 112 123 126 162 165 170 171 172 174 186
##   3   1   3   2   1   1   1   1   1 293   1   1   1   6   1   1
```

In this case, most sequences had a length of 162bp, which is roughly what I expected given our amplicon length and sequencing platform. Given that there were so few sequences outside of this length, I chose to filter out all of them, to retain just the sequence variants of 162bp lengths.

ii. Remove chimeras The core dada algorithm accounts for indels and sequencing errors, but does not automatically remove chimeric sequences. We will use the removeBimeraDenovo command to do this now. There are multiple methods for calling chimeras, but I am using the "consensus" method which identifies chimeras WITHIN samples, rather than between them. We selected this option based on best practices recommended by Camila Mazoni who also works with MHC, as well as the dada2 manual. We also should not expect to see a high number of chimeras and most reads should be retained here.

84.7% of sequences were retained as non-chimeric sequences, but many unique sequence variants are removed at this step- only 86 remain from 293.

iii. Track reads through pipeline After these initial filters, we can track read loss through the pipeline. There should be no single step where there is high read loss, but we expect to lose reads at every stage of the pipeline.

```
##                             input filtered denoisedF denoisedR merged nonchim
## MHC_Sample105_cm_Hawaii      6045     4471      4421      4420   4235    3732
## MHC_Sample110_cm_Hawaii      4783     3488      3454      3439   3084    2806
## MHC_Sample117_cm_Oahu        6653     5134      5086      5116   3873    3732
## MHC_Sample125_cm_East-FFS    4664     3401      3336      3351   2929    2354
## MHC_Sample133_cm_East-FFS    4561     3235      3172      3214   2794    2058
## MHC_Sample14_cm_Hawaii       3880     2695      2672      2689    189     189
```

The output table has a row for each sample, and displays the number of reads retained at each point in the pipeline above. For this dataset, I am seeing the most loss at the merging step, and some loss during chimera removal, but most samples still have greater than 3000 reads retained by the end of the filtering steps above. Even at the merging step, the read loss isn't too significant.

Sample 14 clearly failed on merging, and will be filtered out below

iv. Filter out variants present in only one individual First, I am filtering out any variants that are only being called in one individual. While it is possible that these variants are in fact true, rare variants, this is still a relatively small dataset and we lack confidence that these are true alleles.

The output table “mod_seqtable” contains rows with every sequence variant, and columns of individual samples, with columns on the far right showing the sum of individuals with a given sequence variant, and the total number of reads in support of a sequence variant across individuals.

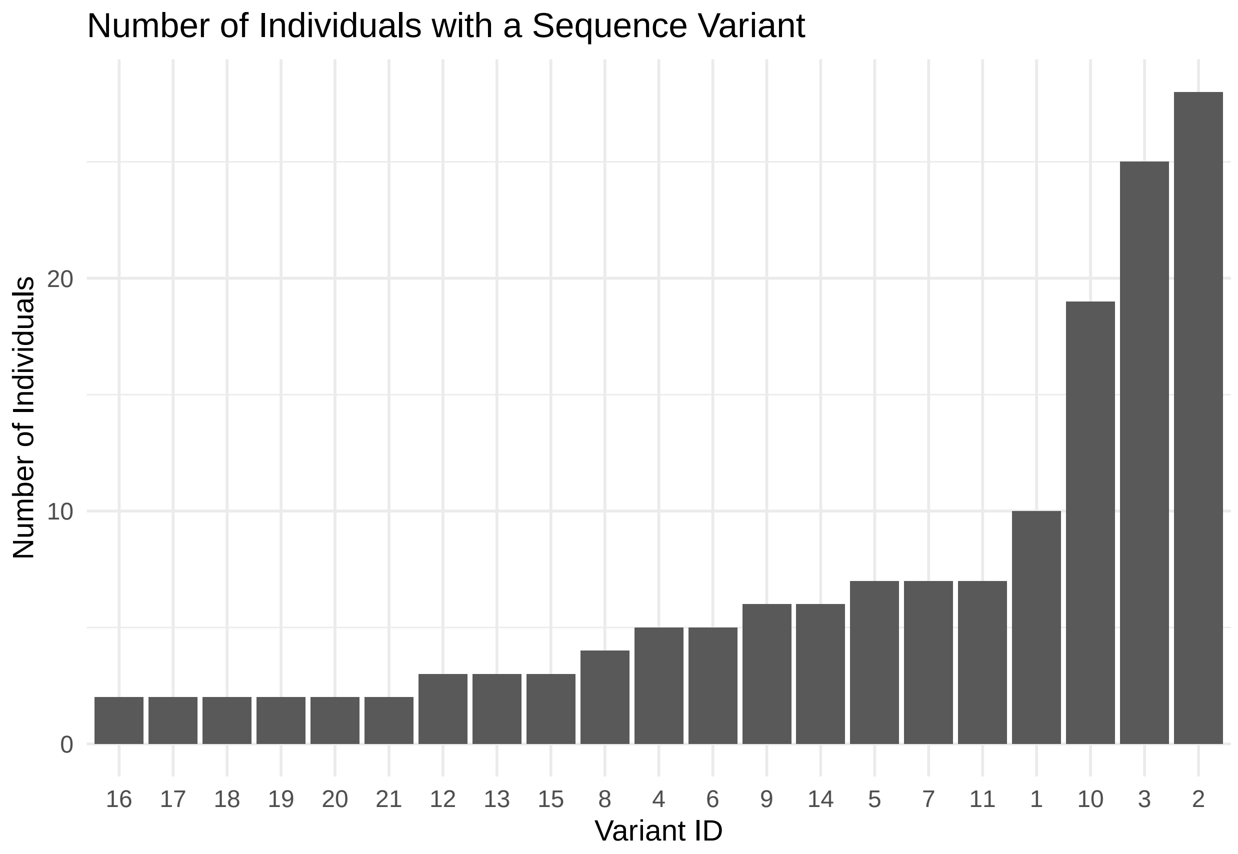
After filtering here, total variants drops from 86 to 54.

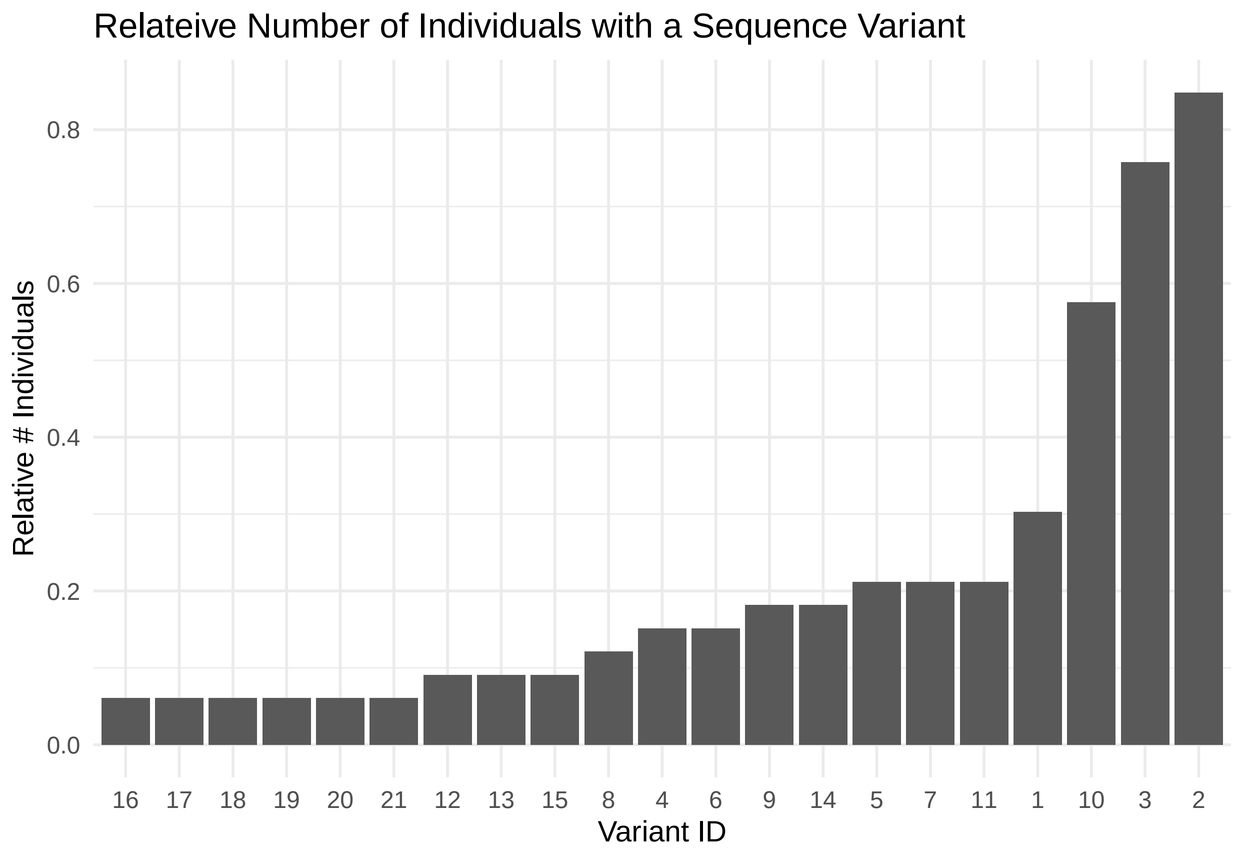
I also removed one sample from the analysis that appears to have failed on merging.

At this point, 34/35 individuals remain in the dataset.

v. Filter out variants calls in individuals with less than 100 reads I then removed variant assignments in individuals where there were fewer than 100 reads supporting that assignment. For individuals with 5000 total reads, that would mean removing variant assignments that comprised less than 2% of total reads. This is similar to filtering done in Stervander et al. where assignments were removed if the supporting reads were less than 5% of total reads for an individual. After this, Sample21 was removed for having too few reads remaining.

At this point, I plotted the number of individuals that contained each variant, and the relative proportion given the total number of individuals in the dataset. This is done to determine if the filter of removing only variants present in one individual should be modified:

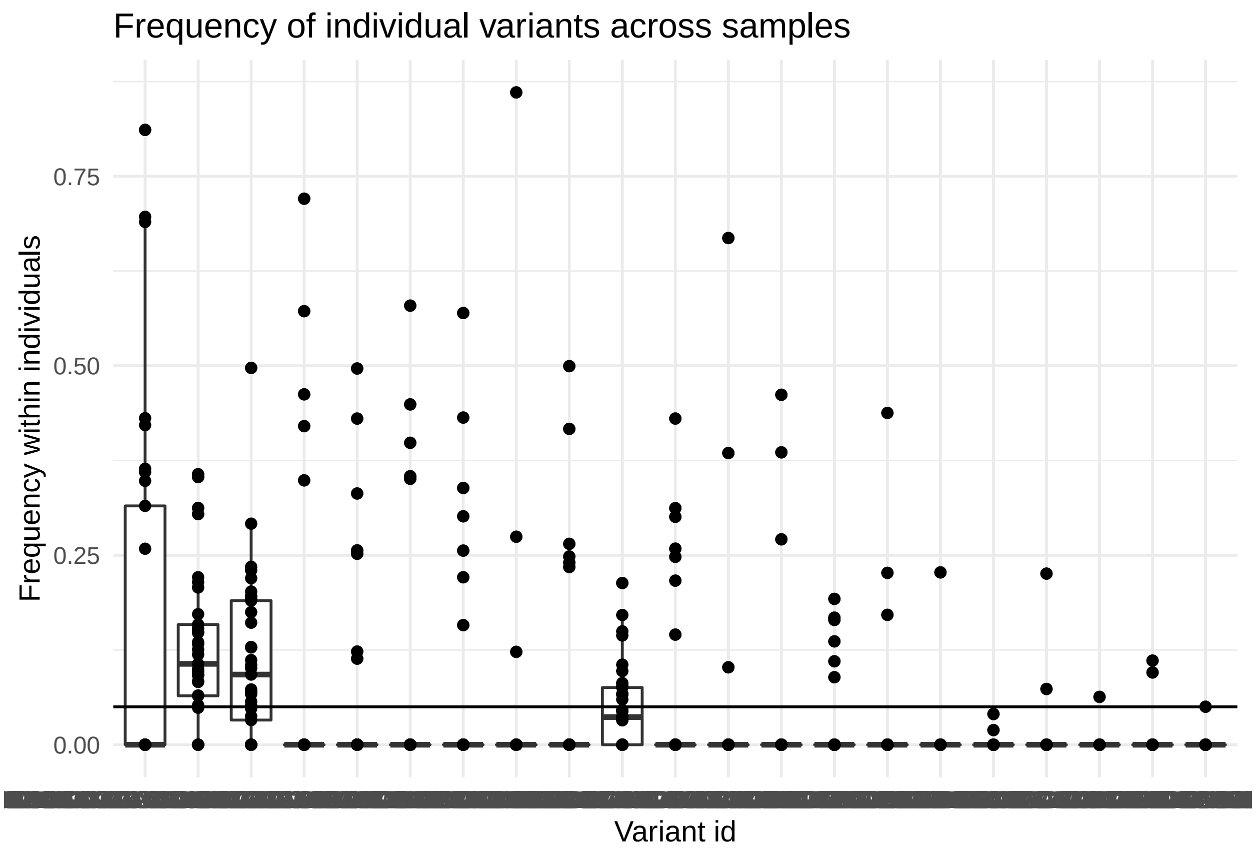




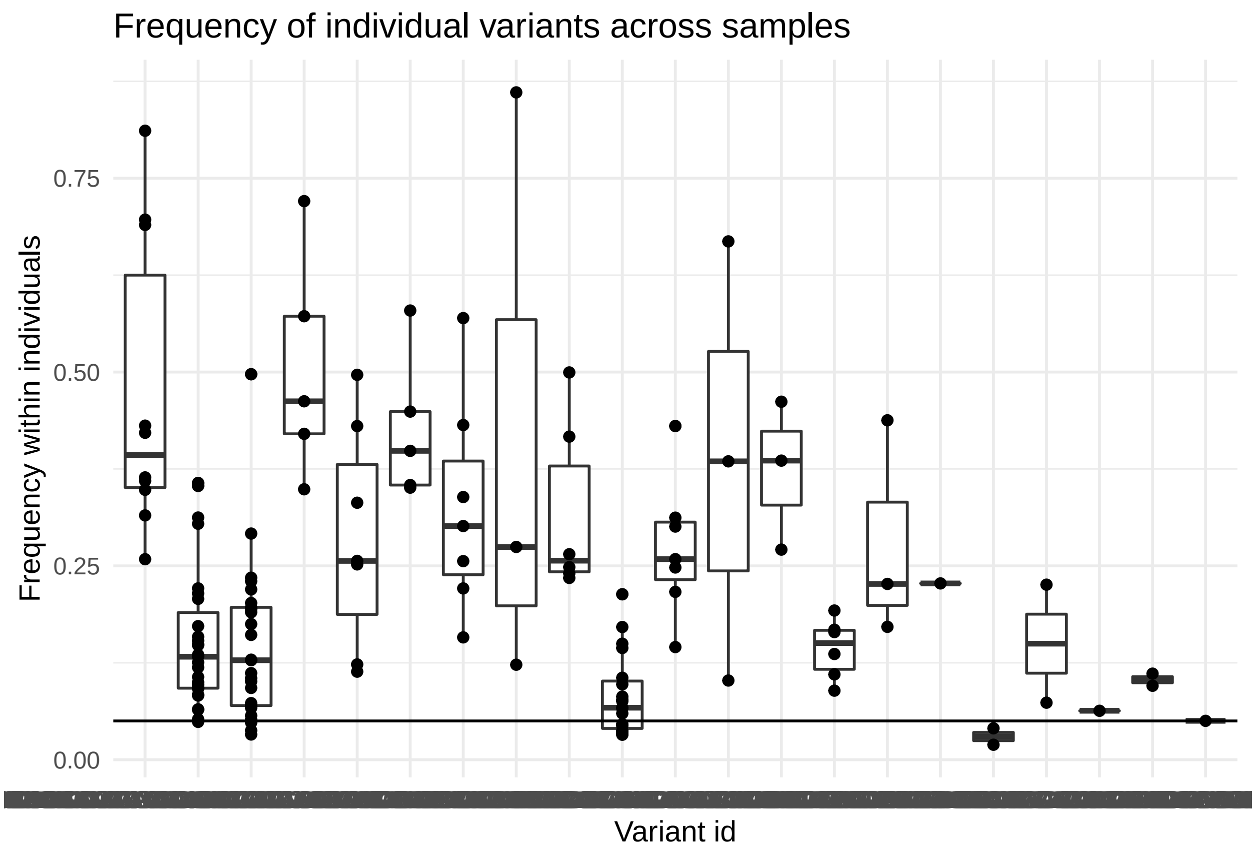
From this, we see that there are 6 variants that are present in just 6% of individuals, but none of the variants fall below a 5% threshold given the total number of individuals in this population. If we were to remove these 6 variants, we would be left with 15 variants.

vi. Filtering by within-individual frequency We also wanted to filter variants on a per-sample basis, removing variants that made up a low proportion of total reads in an individual sample.

I first plotted within-individual frequency statistics for a few samples. For the three samples shown, we see that there are few variants called per individual, and also that generally these exist in relativly high proportions, so that a cutoff of 5% seems reasonable.



Frequency of individual variants across samples
Frequency within individuals
0.75
0.50
0.25
0.00
Variant id



THe black line on the plot of Frequency of individual variants across samples indicates a 0.05 frequency theshold, or the point at which a variant is supported by less than 5% of reads in an inidividual. we see that if we filter out certain variants from the datasets at a threshold of 5% within individuals, only one variant should be removed entirely from the dataset, and a few variant calls will be removed within individuals but remain in the dataset.

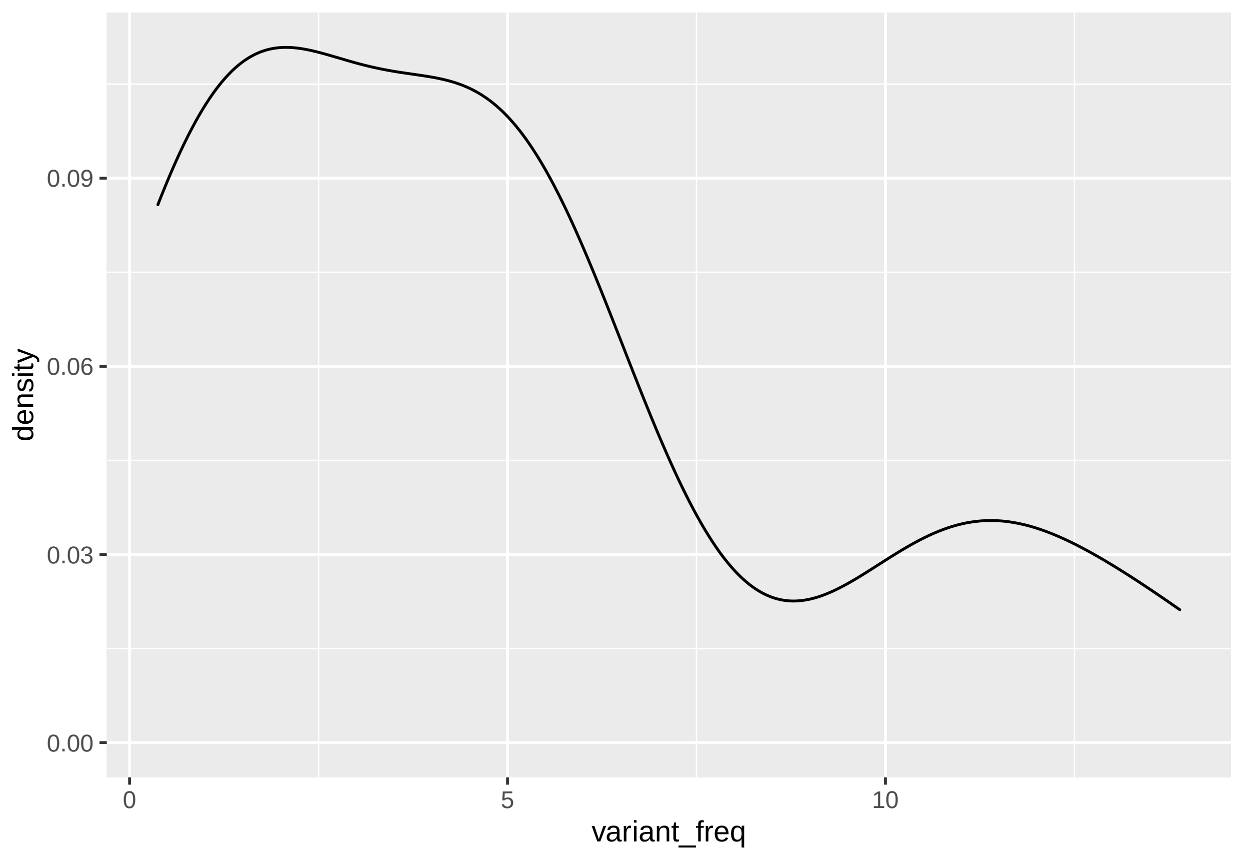
The first plot includes frequencies of 0 (individuals who do not have a variant), whereas in the second plot these 0 values have been replaced with NA so only individuals who have the variant are plotted.

Summary statistics

```
#write.csv(mod_seqtable2,"filt_seq_table_trimreads.csv")

#get column of frequency in dataset for each variant
total_reads<-sum(mod_seqtable$total_calls) #101231 total reads
mod_seqtable$variant_freq<-mod_seqtable$total_calls/total_reads*100


ggplot(mod_seqtable,aes(variant_freq))+
  geom_density()
```



```
mean(mod_seqtable$variant_freq) #4.76
```

```
## [1] 4.761905
```

```
min(mod_seqtable$variant_freq) #0.374
```

```
## [1] 0.3743299
```

```
max(mod_seqtable$variant_freq) #13.89
```

```
## [1] 13.89395
```

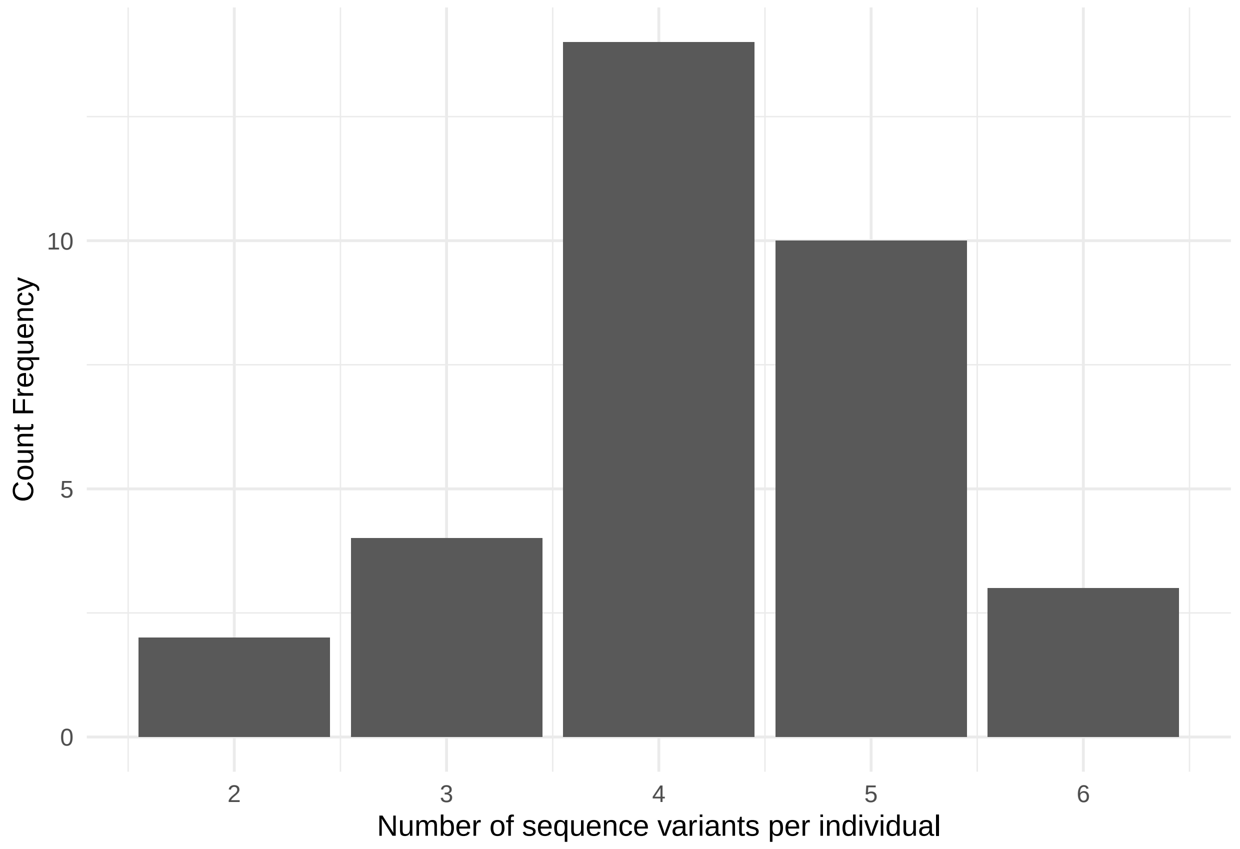
```
median(mod_seqtable$variant_freq) #4.37
```

```
## [1] 4.375998
```

Most sequence variants exist at low frequency (frequency given by total number of reads in the dataset, NOT the individuals in this case) within this dataset. Interestingly, the frequency distribution appears to be biomodal, with a subet of high frequency variants and a large number of lower frequency variants, with a clear break in the middle.

For the total number of variants being assigned to each individual, I summed the number of variants with reads supporting it in each individual. The minimum is 2, maximum of 6, and the mean is 4.24. This indicates that there are up to 3 gene copies within this population.

Below is the distribution of sequence variants per individual.



Nucleotide diversity

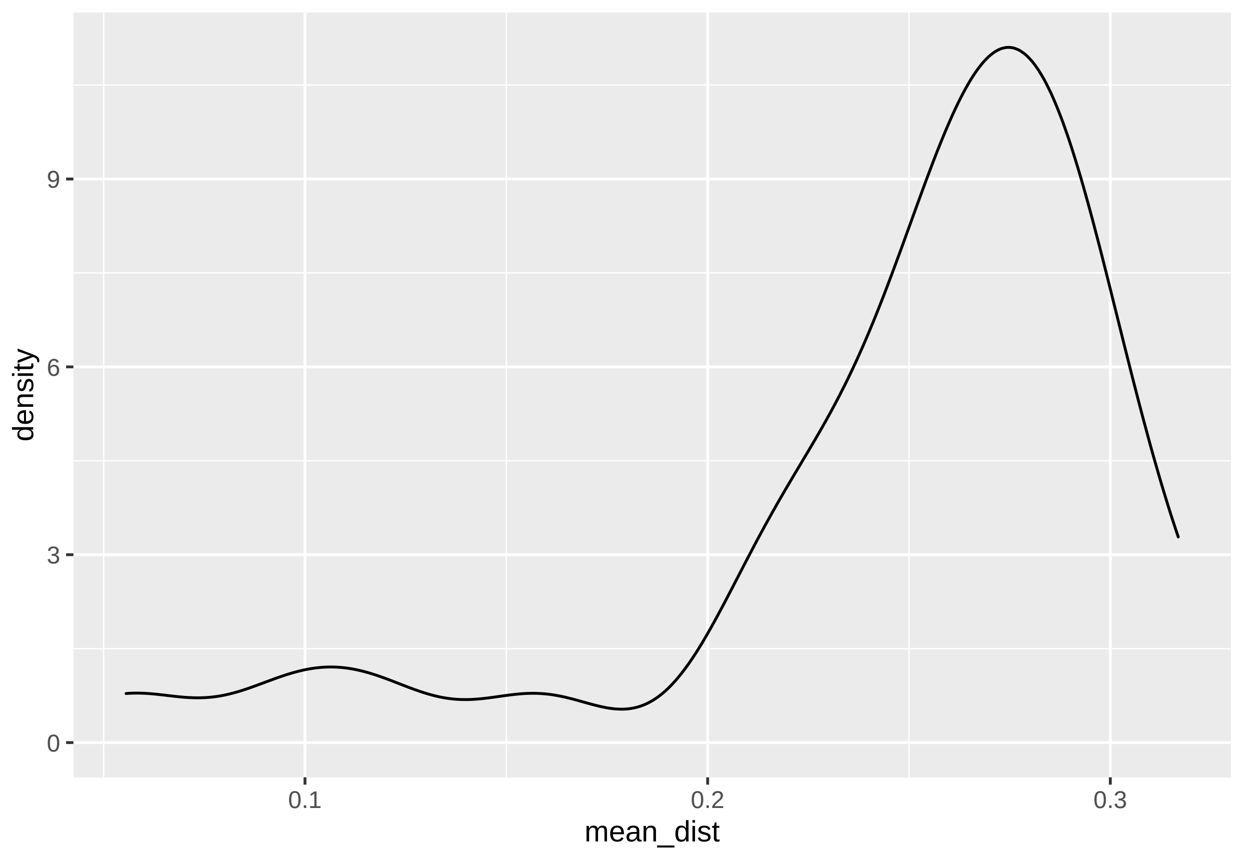
I used the MHCtools package to quantify sequence divergence in a pairwise fashion for all sequence variants, and also get a value of mean sequence divergence per sample.

In MHCtools, I calculated pairwise distances using the DistCalc function, and distance type='P'. There are other distance metrics that can be used, but p distance is the proportion of nucleotide sites at which two sequences are different, or the number of nucleotide differences divided by the number of nucleotides compared. A p-distance of 1 would indicate two sequences were entirely different from each other at every position, whereas a value of 0 would indicate identical sequences.

The output of this funtion are two csv files; one is a matrix of p-distances of all sequence comparisons, and the other is the mean p-distance per sample.

Then I looked at density of mean distances per individual. On average, individuals had a mean p-distance of 0.24, indicating most indivduals had fairly low sequence variation. However, the maximum was 0.31 and the minimum was 0.055, so there are at least some individuals with relatively high sequence diversity as well as some with very low sequence diversity.

For the most part, sequences were quite distant from each other. Sequence 17 and 9 were highly similar. Sequences 16, 12, and 13 were also quite similar. Sequence 9 and 7 were also similar.



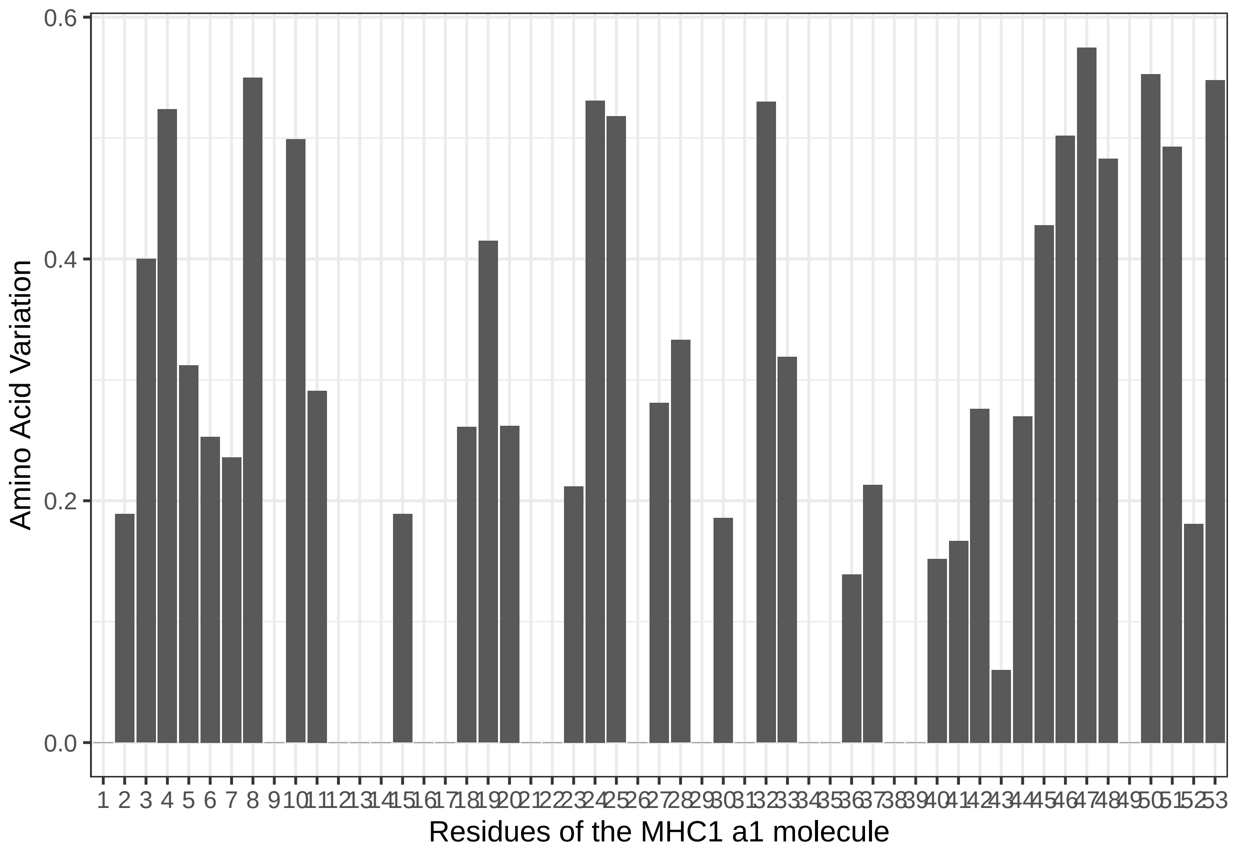
## Amino acid diversity

Using MHCtools, I converted the list of sequence variants to a fasta file. The sequences (1-17) are ordered by total allele calls in the dataset. I had previoulsy used Transdecoder on the command line to identify the longest open reading frame for the sequences, but this doesn't work now with the few sequences for some reason. Instead, I used EMBOSS Transeq for the conversion, and now ORF3 is the longest ORF with no gaps. Based on the length of the sequence and the lack of stop/start codons, ORF3 is the most appropriate ORF for this analysis. I used the ORF3(+) strand rather than (-) as this should be the 5'-3' sequence direction.

I used ScoreCons (as in Stiebens et al. paper characterizing MHC in loggerheads) to get a measure of conservation at each codon when evaluating all sequences in the dataset.Diversity of position scores (or "Dops") considers the number of different scores in the alignment and the relative frequency of each score.

- the overall diversity of position score was 77.6%. This was done by comparing all sequences in the dataset to the first sequence in the dataset
- Default parameters were used for ScoreCons, I used the valdar01 score method, as this accoutns for both variation in residue identity as well as stereochemical diversity at a given position.

I recreated the figure from Stiebens et al. 2013 to see levels of amino acid variation by residue. Here the Y axis "Amino Acid Variation" corresponds to 1- the ScoreCons score for that position. ScoreCons gives a score to each position based on how conserved the residue is, so a value of 1 would indicate that the residue is the same (highly conserved) in all sequnences. Here I want to see how variable a residue is, so a value of 1 would indicate completely different for each sequence.



Based on this figure, 11 amino acid positions are variable at levels of 0.5 or greater, indicating that at 11 out of 53 amino acids the sequences are over 50% variable.

Tests for selection

In Stiebens et al. 2013, the authors also performed tests for selection in the region, as well as by individual residue.

-I have not yet explored how to do this by residue, but I did run a quick test for the entirety of the region using the MEGAX GUI. The input here was the fasta file of all sequence variants in nucleotide format.

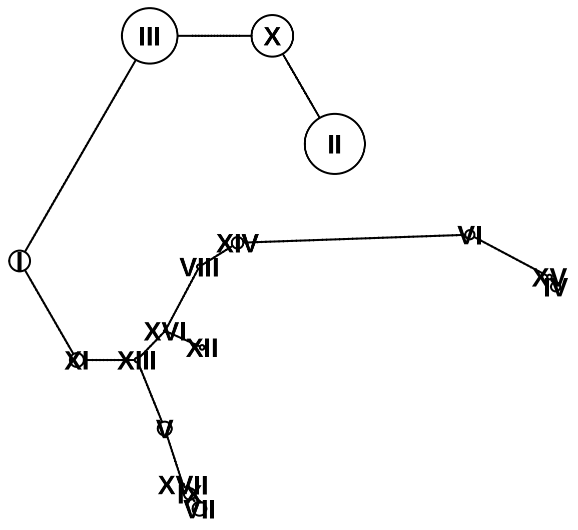
- As in the Stiebens paper, I tested for signatures of neutral and positive selection following the method of Nei and Gojobory with the Jukes-Cantor correction for multiple substitutions. The rate ratio dN/dS was tested for significant deviation from one using a Z-test. The basis of this is that "under positive selection, a relative excess of non-synonymous over synonymous substitutions is expected."
- When testing the hypothesis of neutral selection, the overall probability was 0.12, indicating it would be inappropriate to reject the null hypothesis that synonymous and nonsynomous substitutions are equal
- When testing the alternate hypothesis of positive selection, the overall probability is 0.06, which is very close to 0.05, but likely not able to reject the null hypothesis
- When testing for purifying selection, the overall probability is 1.0, so we definitely cannot reject the null hypothesis in this case.

- I have not yet found a way to test directly for balancing selection, it seems like we might have either balancing selection OR positive selection, but somewhat weaker. This could also be due to the amount of sequences in the dataset. Input for this was simply the 17 nucleotide sequences.

Network analysis of haplotype sequences

```
## 123 DNA sequences in binary format stored in a matrix.
##
## All sequences of same length: 162
##
## Labels:
## 1
## 2
## 3
## 4
## 5
## 6
## ...
##
## Base composition:
##     a     c     g     t
## 0.283 0.200 0.308 0.209
## (Total: 19.93 kb)
```

```
##
## Haplotypes extracted from: chmy_seqs
##
##     Number of haplotypes: 17
##          Sequence length: 162
##
## Haplotype labels and frequencies:
##
##    I   II  III   IV    V   VI  VII VIII   IX    X   XI  XII XIII  XIV   XV  XVI
##    9   26   24    4    6    4    6    2    5   18    6    2    2    5    2    1
## XVII
##    1
```



```
##    Haplotype_ID
## 1             I
## 2            II
## 3           III
## 4            IV
## 5             V
## 6            VI
## 7           VII
## 8          VIII
## 9            IX
## 10            X
## 11           XI
## 12          XII
## 13         XIII
## 14          XIV
## 15           XV
## 16          XVI
## 17         XVII
##
## 1  gtgacggctccatcgggggggttttaccaggatgcatacgatgggcgagacttcataagcctggataaggatctggagacctgggtggcggcagatga
## 2  aagacaacaccacttgggggttttaccaggattcatatgacggacgagactttcttatcttcgataaggagacaatgactttgggagcagaagatat
## 3  aagacggcgtcattcaggtgttttaccaggattcgtatgatggacgagactttcttaccttggataaggagaccatgacttgggtagcagcagatat
## 4  atgacggcaccactggtgcttttcaccagtacgggtatgatgggagagactttgtcagtttcgacaaggacaccctgacctacactgcagcagatgc
## 5  atgacggctccaaaggcggggtttttcagtttgcgtatgatgggcaagacttcctcaacctggataaggaccatgagacctgggtggcagcagatga
## 6  atgacggcaccactgggggggtttgaccagtttgcatatgatgggagagactttgtcagcttcgacaaggacaccctgacctggacagcaacggatgc
## 7  gtgacggctccaaaggtggggtttttcagtttgcgtatgatgggcaagacttcctcaacctggataaggaccatgagacctgggtggcagcagatga
## 8  gtgacggctccaaagggggggttttctcagtatgcgtacgatgggcgagacttcctcagcctggataaggactgggagacctgggtggcggcagatga
## 9  gtgacggctccaaaggcggggtttttcagtttgcgtatgatgggcaagacttcctcaacctggataaggaccatgagacctgggtggcagcagatga
## 10 aagacaacgtcattcaggggttttaccaggattcgtatgacggacgagactttcttaccttcgataaggagaccatgacttgggtagcagcagacat
## 11 atgacggctccatcgggggggtttcgccaggatgcgtacgatgggcgagacttcctcagcctggataaggatctggagacctgggtggcggcagatga
## 12 gtgacggctccaaagggggggtttcgccagtatgcgtacgatgggcgagacttcctcagcctggataaggaccatgagacctgggtggcagcagatga
## 13 atgacggctccatcgggggggtttttcagtatgcgtacgatgggcgagacttcctcagcctggataaggaccatgagacctgggtggcggcagatga
## 14 gtgacggctccaaagggggggttttctcagtatgcgtacgatgggcgagacttcatcagcctggataagcaccaggagacctgggtggcagcagatga
## 15 atgacggcaccactggtgcttttcaccagtacgcgtatgatgggagagactttgtcagtttcgacaaggacaccctgacctacactgcagcagatgc
## 16 gtgacggctccaaagggggggttttctcagtatgcgtacgatgggcgagacttcctcagcctggataaggaccatgagacctgggtggcagcagatga
## 17 atgacggctccaaaggcggggtttttcagtttgcgtatgatgggcaagacttcctcaacctggataaggaccatgagacctgggtggcagcagatga
```

Future ideas

- Figure out how to test for balancing selection, selection at specific codons
- Linear model of age class as it relates to individual mean p-distance (look at age structure of MHC region and whether diversity has increased over generations)
- Compare within-individual diversity and copy number to tumor score if/when available
- Linear model of nucleotide diversity as it relates to copy number within an individual
- can I do network analysis by individual? Is the above network analysis helpful?