1. INTRODUCTION

In a hot spring located in Iceland, episodes of high temperature (close to 90 degrees) occur coinciding with regular activities in a nearby volcano. After such episodes, a bloom of algae arises in this environment. An algal bloom is a rapid increase in the density of algae in an aquatic system. Algae can multiply quickly in waterways with an overabundance of nitrogen and phosphorus, particularly when the water is warm. This proliferation causes blooms of algae that turn the water green.

This phenomenon has already been observed previous times. M. J. Winterbourn, already in 1969, studied the presence of algae in hot springs, being the upper temperature limit for algal growth 68ºC and having optimum growth at a temperature of 59-62ºC (Winterbourn, 1969). This would explain why the bloom of algae occurs after the increase in temperatures.

A more recent example can be found on the work done by Rozanov *et al*, on 2017, in which some species of algae were detected in the Garga hot spring (Baikal rift zone, Russia) by their chloroplasts' 16S rRNA (Rozanov *et al*., 2017).

To investigate this effect, I will perform an in-depth genomic exploration of the ecosystem.

2. METAGENOMICS

2.1. SHOTGUN SEQUENCING

As a first step, we extract the DNA from two prokaryotic microbiome samples (one taken during the high temperature episodes and another right after the episodes, when the temperature is back to normal and there is a bloom of algae) and then run a shotgun metagenomic sequencing.

Shotgun sequencing is a classic method which involves the following steps:

1. The DNA is fragmented in many small sequences.
2. The sequences are organized in vectors with known flanking sequences (allowing the fragments to be sequenced).
3. The vectors are introduced in bacterium (usually *E. coli*), which are cultivated separately. As a result, we obtain different colonies.
4. We obtain a sequence library, from which we can ensemble the sequences.

2.2. FASTQ FILES ANALYSIS

As a result, we get the sequencing results from our two metagenomic samples. The raw read files (reverse and forward) were produced by Illumina paired-end sequencing. Paired-end sequencing is a technique that sequences both ends of each fragment, so that the intermediate sequences are solved by overlapping. This method minimizes the errors, because not many consecutive nucleotides are sequenced.

Our files are in a FASTQ format. This format stores biological sequences with its corresponding quality score. Line 1 begins with a '@' character, and contains the sequence identifier. Line 2 contains the sequence letter. Line 3 begins with a '+' character and is optionally followed by the same as in line 1. Finally, line 4 contains the quality values, with one symbol for each nucleotide in the sequence. In Illumina, the best score corresponds to symbol 'I' (Q score of 40).

The first thing that must be done is analyze the quality of our reads. To do this, I use FASTQC, which is a quality control tool for high throughput sequence data (Wingett & Andrews, 2018). FASTQC is a Java application, so instead of running it on the environment for the task (where it was not installed), I had to do it on my own device. I then saved the results as an html and copied them on the server. I get the following results:

- *Basic statistics*: I get the results shown on *Table 1*. No sequences are flagged as poor quality, and the %GC is greater in the *hightemp* files than in the *normal* files (this will be discussed later). The number of sequences is greater in the *hightemp* files, and they are also longer sequences.

Table 1. **Basic statistics of the FASTQ files obtained from shotgun sequencing.**

	Total sequences	Sequences flagged as poor quality	Sequence length	%GC
***Hightemp* forward**	1.133.559	0	45-150	44
***Hightemp* reverse**	1.133.559	0	45-150	44
***Normal* forward**	286.690	0	45-92	34
***Normal* reverse**	286.690	0	45-98	34

- *Per base sequence quality*: all four files have very good values, ranging from Q scores of 40 to above 30.
- *Per sequence quality scores*: all the files sow, again, good results; the *hightemp* files have most of the sequences with a Q score of 40, and the *normal* files have most of the sequences with a Q score of 38 or 39.
- *Per base sequence content*: here, I find interesting results, shown on *Figure 1*. As can be seen, the value of G and C content in the *hightemp* files (*Figure 1A* and *1B*) ranges between 40 and 60%, while in the *normal* files (*Figure 1C* and *1D*) it is always less than 40%. This makes sense, because CG pairs are more stable than AT pairs, and are therefore more suited for extreme conditions such as very high temperatures.

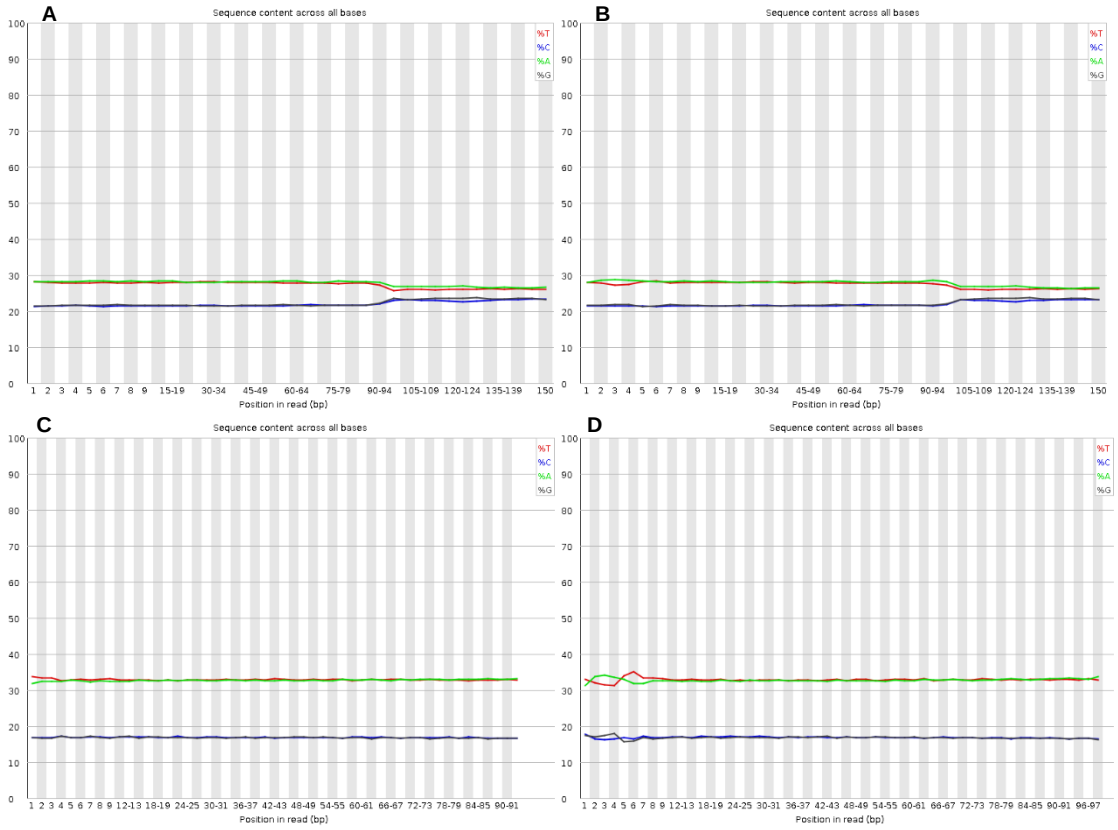

Figure 1. **A, C, G and T content for the FASTQ files obtained from the shotgun sequencing.** Panel A shows the results for the forward reads of the high-temperature microbiome. Panel B shows the results for the reverse reads of the high-temperature microbiome. Panel C shows the results for the forward reads of the normal-temperature microbiome. Panel D shows the results for the reverse reads of the normal-temperature microbiome.

- *Per sequence GC content*: in three cases this model fails (the sum of the deviations from the normal distribution represents more than 30% of the reads), and in one of them it raises a warning (the sum of the deviations from the normal distribution represents more than 15% of the reads). The reason is that many bases have a higher percentage of GC content than expected. This may be due to the extreme conditions, as explained above.
- *Per base N content*: if a sequencer is unable to make a base call with sufficient confidence, then it will normally substitute an N rather than a conventional base. In all files, for every position the N content was 0.
- *Sequence length distribution*: the reads are of varying lengths. In the *hightemp* files, there is a peak of around 250.000 sequences that are around 92 bp long, and the rest are around 150 bp long. On the other hand, in the *normal* files, all the sequences are around 92 bp long.
- *Sequence duplication levels*: in the *hightemp* files, there are too many repetitions of sequences (more than 30% of sequences have been sequenced more than 10 times), which means that the same regions have been resequenced multiple times. Thus, we could have effectively got the same amount of information by doing less sequencing. On the *normal* files, however, there are not too many duplications.
- *Overrepresented Sequences*: there are no overrepresented sequences in any file.

2.3. RELATIVE ABUNDANCE OF ORGANISMS UNDER EACH CONDITION

Next, I am going to use mOTUs2, which is a tool to profile metagenomic or metatranscriptomic samples (Milanese *et al.*, 2019). A phylogenetic profile is used to check the presence or absence of a protein in a set of reference genomes, comparing phylogenetic data across gene families (Pellegrini, 2012).

mOTUs uses phylogenetic markers to taxonomically profile shotgun metagenomes. Phylogenetic markers are genes that can be used to profile the taxonomic composition of environmental samples. The mOTU profiler uses 10 universal marker genes (that occur in single copy in the vast majority of known organisms) that are used to delineate prokaryotic organisms at the species level. The resulting database contains 2.494 metagenomic mOTUs and 5.232 reference mOTUs. The reads are aligned against this database to taxonomically classify them.

On bash, I run the script *motus_script.txt*. This script takes the two pairs of FASTQ files and does the taxonomic profile. Then it generates two files, each with the 10 most abundant organisms of each profile, and two more files in which the relative abundances on both profiles of these 10 organisms are written. I run the script with this command:

```
./motus_script.txt hotspring-hightemp.1.fq.gz hotspring-hightemp.2.fq.gz hotspring-normaltemp.1.fq.gz hotspring-normaltemp.2.fq.gz
```

The command used to generate the profile is the following:

```
motus profile -f [forward read] -r [reverse read] -o [name]
```

Where *[forward read]* is the name of the FASTQ file containing the forward read, *[reverse read]* is the FASTQ file containing the reverse read, and *[name]* is the name given to the output file (typically finished in *.motus*).

Then, I sort the result of both files generated by relative abundance. The five most abundant organisms under the *hightemp* and *normal* conditions are shown on *Table 2* and *Table 3*, respectively. The most outstanding thing is the fact that, under high temperatures, the most abundant organism is *Aquifex aeolicus*, which is not present (at all) under normal conditions. Furthermore, the relative abundance of this organism under high temperatures is 0’619, which means that more than 60% of the present microorganisms in the prokaryotic microbiome are *Aquifex aeolicus*. Other than that, in both conditions are abundant the *Pelagibacteraceae* species, an unknown organism (which might or might not be the same in both conditions) and the

Porticoccaceae species. As can be seen, the ranking is pretty much the same, being the relative abundances in normal conditions higher due to the lack of *Aquifex aeolicus*.

Table 2. **Relative abundance of the five more abundant organisms in high-temperature conditions and in normal-temperature conditions.**

Organism	Relative abundance in *hightemp*	Relative abundance in *normal*
Aquifex aeolicus	0'619	0'00
Pelagibacteraceae species *inc sed.*[a]	0'091	0'239
-1[b]	0'072	-
Pelagibacteraceae species *inc. sed.*	0'066	0'174
Porticoccaceae species *inc. sed.*	0'019	0'051

[a] *inc. sed.* is an abbreviation for *incertae sedis*, which is the term used for a taxonomic group where its broader relationships are unknown or undefined.
[b] -1 is used for unknown organisms, not present in the mOTUs database.

Table 3. **Relative abundance of the five more abundant organisms in normal-temperature conditions and in high-temperature conditions.**

Organism	Relative abundance in *normal*	Relative abundance in *hightemp*
Pelagibacteraceae species *inc sed*	0'239	0'091
-1	0'189	-
Pelagibacteraceae species *inc sed*	0.174	0'066
Porticoccaceae species *inc. sed.*	0'051	0'019
Candidatus aquiluna sp. IMCC13023	0'048	0'018

2.4. AQUIFEX AEOLICUS

Thus, the only notable difference between high and normal-temperature conditions seems to be the organism *Aquifex aeolicus* (*A. aeolicus*), whose presence is really high under high-temperatures. The order *Aquificales* constitutes an important component of the microbial communities at elevated temperatures (Guiral *et al*., 2012).

Aquifex spp. Are thermophilic, and often grow near hot springs. *A aeolicus* requires oxygen, but some species can grow anaerobically thanks to the reduction of nitrogen. However, *A. aeolicus* has not been shown to be able to grow this way.

A. aeolicus is one of the most thermophilic bacteria known; it is a chemolithoautotrophy (which means that uses inorganic carbon for biosynthesis and an inorganic chemical energy source) that can grow on hydrogen, oxygen, carbon dioxide, and mineral salts (gotten from https://www.genome.jp/dbget-bin/www_bget?gn:T00013). This organism grows at 95ºC, which explains why it is only present under the high-temperature conditions. It has a really small genome, one-third the size of the *E. coli* genome, which results in reduced metabolic flexibility (Deckert *et al.*, 1998).
The genome of *A. aeolicus* has been completed; it possesses many enzymes related to the energy sulfur metabolism, as well as membrane-embedded hydrogenases and oxidases (Robert *et al*, 2013). Some of these proteins have been characterized as super-resistant enzymes.

2.5. PRESENCE OF ALGAE

Algae are eukaryotic organisms (blue and green algae are not algae but *cyanobacteria*). There are a number of reasons why we cannot detect the presence of algae, even if they were present when we took the samples. First, it is stated that we just made a shotgun sequencing of the prokaryotic microbiome. Furthermore, the mOTU database is only comprised of prokaryotic organisms, and therefore eukaryotic organisms cannot be classified. Therefore, we cannot know whether algae are present or not under neither of the two conditions.

However, as shown in the introduction, eukaryotic algae could be identified using their chloroplast's 16S RNA. Thus, it would be possible to identify the presence of algae, having the required tools.

Nevertheless, as stated on Channing, 2010 and Channing, 2018, chlorophyte algae become more common as cyanobacteria numbers decline. Also, as shown on the introduction, it is common for algae to be present in hot springs. These facts support the idea that eukaryotic algae might be present, since there are no cyanobacteria present in either condition (or if they are, with a very low relative abundance).

3. GENOMICS

After finding out that *Aquifex aeolicus* is very abundant in high-temperature episodes, the lab isolated it and sequenced cDNA (just the exons) of samples from both normal and high-temperature conditions, two biological replicates each (the same test is performed on two samples of the organism). They performed quality checking, providing us only high-quality reads in FASTA format.

Thus, we get 4 samples as FASTA files. I run the script *genomics_script.txt* using the following command:

```
./genomics_script.txt hightemp_01.fasta hightemp_02.fasta normal_01.fasta normal_02.fasta
```

This script generates the file *genomics_stats.txt*, which contains the number of reads and the different lengths of such reads for any number of FASTA files provided. The results are shown on *Table 4*. The length of all the reads in all four files is 100. This is probably due to a previous filtering, where just the reads of 100 bp were kept, and all the others discarded.

Table 4. **Number of reads and different lengths of such reads for the two samples of each condition (normal and high temperatures).**

File	Number of reads	Length of reads
Normal sample 1	291.814	100
Normal sample 2	289.637	100
Hightemp sample 1	290.331	100
Hightemp sample 2	291.324	100

We have just one file per sample, which seems to indicate that the reads are single-end (were it mate-paired or paired-end sequencing, we would have a forward and a reverse file for each sample, as in the metagenomics section). However, if we take a look at the headers, we can see that it includes a *mate1* and a *mate2*. This, nonetheless, seems to be an error produced by the program, so I just ignore it.

Single-read sequencing consist on the fragmentation of the cDNA and size-selection (by 100 bp, it seems). Then two adaptors, are ligated to the fragments, one on each strand, one of the including a *sequencing primer site* (SP1). The fragments attach to the flow cell through the adaptors; afterwards, the cell is sequenced from the SP1 while it remains attached to the flow cell through the second adaptor (Illumina Sequencing, 2009).

Single-end reading worsens the ability to identify the relative positions of various reads in the genome, making paired-end reading much more effective in resolving structural rearrangements as well as the alignment of repeated sequences. However, single-end reads are cheaper and less time-consuming to perform.

Following the consensus, all the reads are in the 5'-3' direction. As for the strand, and due to the way the single-read sequencing is done, the file must contain a mixture of reads sequenced from the forward and the reverse strands. There is no information of the strand on the headers, so to know which strand each read belongs to we would have to turn to the references. We will come back to the strand of each read on the *SAM files overview* section.

4. READ MAPPING

4.1. ALIGNMENT TO THE REFERENCE SEQUENCE

Next, I want to align all our reads to our reference genome. In this case, that entails aligning FASTA-formatted sequence reads to the whole genome reference, that is also given in the FASTA format.

As an aligner, I will use Bowtie 2, which is an ultrafast and memory-efficient tool for aligning sequence reads to long reference sequences (Langmead & Salzberg, 2012), just what I want to do. It is extremely useful for "aligning reads of about 50 up to 100s or 1.000s of characters", which makes it perfect for our 100-long reads. It is important that the program is suited for short reads, because otherwise our reads may align to multiple genomic locations.

The first thing I have to do is build an indexed database, specifying the file having sequences to be indexed, and the name of the output file. To do this, I use the following command:

bowtie2-build *[reference genome] [output file]*

This index facilitates the following steps for processing the reference. The next step is the alignment of our files to the indexed database.

bowtie2 -x *[reference genome index]* -f *[fasta file]* -S *[SAM output]* 2>> *[stats output]*

[reference genome index] refer to what I called *[output file]* in the previous command. The *-f* argument allows us to use multi-FASTA input files instead of FASTQ input files, and the *[fasta file]* refers to the FASTA file with our reads. The *-S* argument indicates the file to write SAM alignments to, so that they are not just printed on the console, and *[SAM output]* is the name of such file. Finally, *2* >> redirects these mapping stats to the file *[stats output]*.

These commands, along with the ones necessary to obtain the statistics (explained in the next section) are included in the script *mapping_script.txt*. To run it, I write the command

./mapping_script.txt genome.fna hightemp_01.fasta hightemp_02.fasta normal_01.fasta normal_02.fasta

A directory with the indexed genome is created. Also, another directory with the SAM files, and the two stat files (the stats obtained directly from the *bowtie2* command and the stats obtained using the command line, as explained in the next section).

4.2. SAM FILES OVERVIEW

The outputs generated from the *bowtie2* function are a SAM file containing the alignment and a txt file containing information regarding the percent of reads that aligned to the reference.

The SAM (Sequence Alignment/Map) format is the usual format used to store next-generation sequence alignments; it consists of a header (that begins with the '@' symbol) and an alignment section with 11 mandatory fields tab-delimited (taken from [1] https://github.com/samtools/hts-specs):

1. QNAME: query name of the read or the read pair.
2. FLAG: combination of bitwise FLAGs that contains information about the strand, the pairing…
3. RNAME: name of the reference sequence.
4. POS: 1-based leftmost mapping position of clipped alignment. For instance, if the number is 184.321, the first nucleotide of the read is on position 184.321 in the genome. 0 for unmapped reads.
5. MAPQ: mapping quality. 255 set when the mapping quality is not available.
6. CIGAR: CIGAR string; it indicates which whether the bases are an indel or not.
7. RNEXT: reference sequence name of the primary alignment of the next read in the template. This field is set as '*' when the information is unavailable.

8. PNEXT: 1-based position of the primary alignment of the next read in the template. Set as 0 when the information is unavailable.
9. TLEN: query sequence length. Set as 0 for single-segment template or when the information is unavailable.
10. SEQ: segment sequence.
11. QUAL: ASCII of base quality.

SAM files are easily converted to the BAM (Binary Alignment/Map) format, which is a binary representation of SAM. BAM contains the same information as SAM, but in a compressed format.

The stats file, on the other side, contains info about the number of reads, the number of unpaired reads, the number of sequences aligned 0 times, the number of sequences aligned 1 time and the number of sequences aligned more than one time.

To work with SAM and BAM files I will use SAMtools, which is a library and a software package (Li *et al*., 2009) used to analyze this kind of files.

First of all, I will take a quick look our SAM files. *Table 5* shows the number of reads in our mapping, the number of reads unaligned, the number of reads aligned just once, and the number of reads aligned more than once. All this data can be obtained from the stats files, however obtaining them from the command line is also very useful in case we want to retrieve one type of sequences (the ones aligned just once, for example). Therefore, I have added the necessary lines to the script to get these results using bash (we could retrieve the reads by omitting the *-c* option of *samtools* or the *wc -l* in the other lines), and another file with these stats is created.

Table 5. **Read's statistics of the SAM alignments.**

	Total reads	Reads unaligned	Reads aligned once	Reads aligned more than once
***Normal* 1**	291.814	17	289.275	2.522
***Normal* 2**	289.637	7	287.101	2.529
***Hightemp* 1**	290.331	8	287.957	2.366
***Hightemp* 2**	291.324	10	289.007	2.307

Each record has a single read, because we have single-end reads. The total reads are the same as seen on *Table 4*. The unaligned reads are really low numbers in all cases (close to 0'00%), and the reads aligned more than once are also low (between 0'79% and 0'87%).

We can also check what was said in *section 3*, where we hypothesized that reads were of both strands. The second mandatory field of the SAM format is the FLAG, which can give us this information. If I take, for example, the SAM file for the sample 1 at normal temperatures, I can write the following command:

```
tail -n+4 normal_01.sam | cut -f2 | sort -g | uniq
```

As a result, we get the flags 0, 4 and 16:

- 0: unpaired, mapped to the forward strand reads.
- 4: unpaired and unmapped reads.
- 16: unpaired, mapped to the reverse strand reads.

We get these same flags for all four SAM files.

4.3. EFFECT OF MULTIPLE MAPPING READS ON DOWNSTREAM ANALYSES

Multi-mapped (MMAP) reads occur when there are multiple best-scoring alignments to the reference genome (Johnson *et al.*, 2016). They usually originate from high-copy number regions of the genome. These reads cause ambiguous information in our alignments, which hinders downstream analyses such as variant-calling and RNA-seq. Counting each MMAP read as many times as it appears will due an over-representation of the loci abundances.

MMAP reads are often dealt with simplistically, either by randomly selecting one of the possible alignment positions (bowtie, by default, does this) or by ignoring them. These methods have big downsides: random selection results in large error rates, and removal of MMAP reads discards a lot of information. In our case, the MMAP reads are less than 0'9% of the total reads in all cases, so the treatment of these MMAP reads will not highly affect the result we obtain.

If the percentage of MMAP reads was higher, we could use some more sophisticated approaches, for example using the ERANGE method. This method measures the expression of MMAP reads as a proportion of uniquely mapped reads within a particular read-cluster (Johnson *et al.,* 2016).

4.4. USEFULNESS FOR COPY NUMBER VARIATION ANALYSIS

A copy number variation (CNV) is a phenomenon where the number of copies of a particular gene varies from one individual to the next (taken from https://www.genome.gov/genetics-glossary/Copy-Number-Variation). The widespread presence of copy number variation in normal individuals has been demonstrated (Freeman *et al.*, 2006), making the analyses on copy number variation gain interest.
The first thing to take into account is the fact that our reads are cDNA, not genomic DNA (which is a far more complex mixture than the mRNA), hindering the analysis of CNVs. In humans, for example, analysis of DNA copy-number variation using cDNA microarrays would require a sensitivity of detection an order of magnitude greater than with genomic DNA (Pollack *et al.,* 1999).

Copy-Number Variants analyses are based on different parameters, depending on their size (Trost *et al.*, 2018):

- Small CNVs detection methods are usually based on paired-end mapping or split reads. Therefore, our single-end reads are not suited for detecting this kind of CNVs.
- Larger CNVs, which are more likely to be clinically relevant, are generally analyzed with algorithms that are primarily based on read depth.

Therefore, to see if our data are adequate to study large CNVs, we must plot the read-depth of our data in small bins. The results are shown on *Figure 2*. This figure was generated using the scripts *depth_script.txt*, for bash, and *depth_plot.R*, for RStudio. The script *depth_script.txt* takes any number of SAM files and creates their corresponding BAM files, their corresponding sorted version and its coverage per base. This script is run as follows:

```
./depth_script.txt hightemp_01.sam hightemp_02.sam normal_01.sam normal_02.sam
```

This script generates the txt files with the coverage (deletes all the other generated files, because I will generate them again later), which are then taken to *depth_plot.R* to generate the plots.

As can be seen on *Figure 2*, most of the bases have a read depth of between 15 and 25. According to Trost *et al*. (2018), the optimal average read depth is around 25-40x. Therefore, our samples' read depth is a little low to perform this kind of analysis at optimum performance. Also, even though it is not important right now, we can see that there are two regions on normal-temperature conditions and two regions on high-temperature conditions that have a really high read depth, which means that they are differentially expressed on each condition. We will come to this later.

If, anyhow, we wanted to perform the analysis, we could use tools such as CNVnator (https://github.com/abyzovlab/CNVnator) or ERDS (https://omictools.com/erds-tool), which are open-source tools for CNV discovery from depth-of coverage by mapped reads.

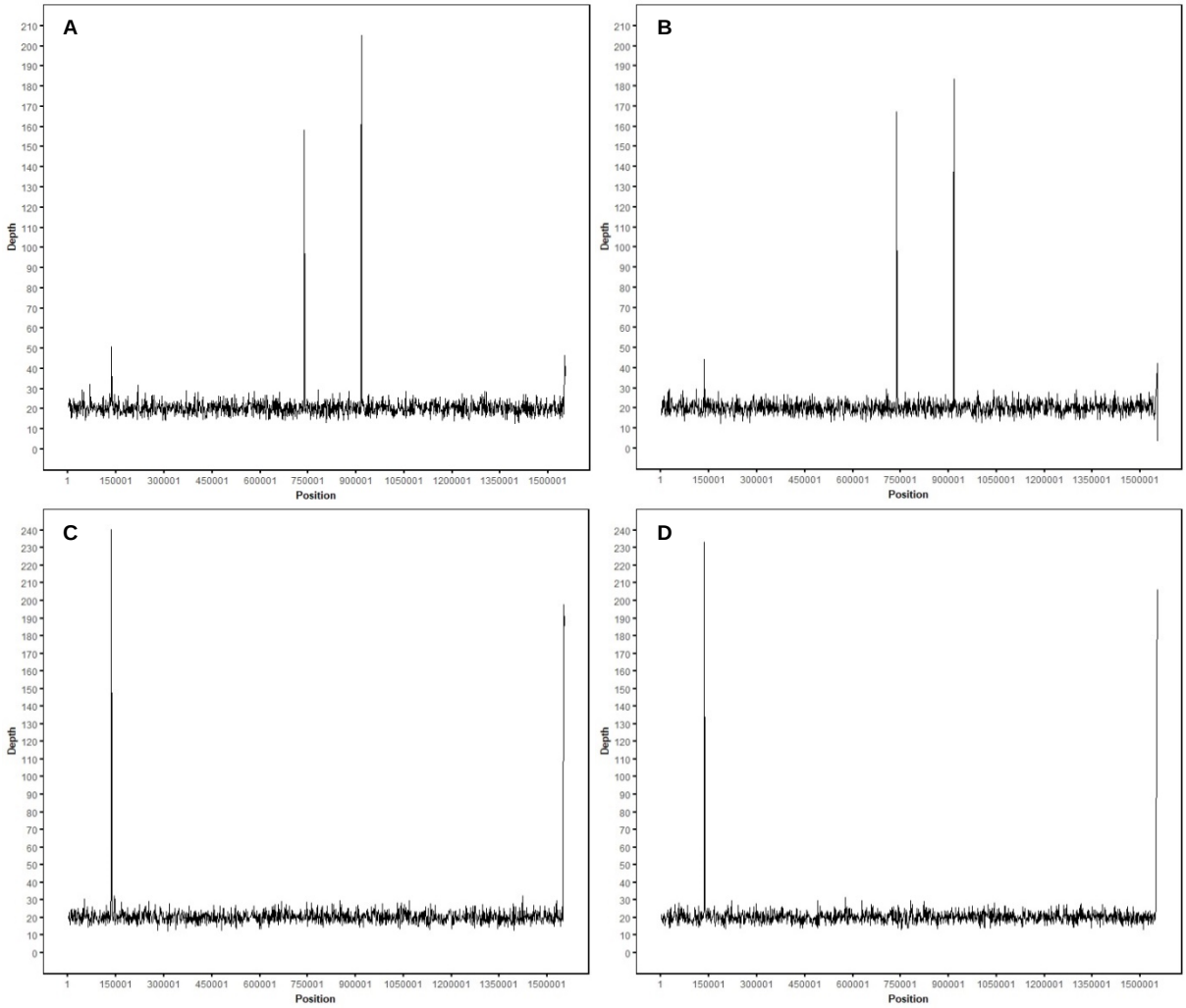

***Figure 2*. Read depth of the BAM files generated by SAMtools.** Panel A corresponds to the first sample at normal temperatures, panel B corresponds to the second sample at normal temperatures, panel C corresponds to the first sample at high temperatures, panel D corresponds to the second sample at high temperatures.

5. VARIANT CALLING

The next step is the analysis and identification of variants in our sample, trying to find whether some mutation is responsible for the sudden proliferation of *Aquifex aeolicus*.

Variant calling is the process by which we identify variants from sequence data. Specifically, a variant call is a conclusion that the sequence we are studying has a nucleotide difference compared to the reference genome at a given position (Nielsen *et al.*, 2011); it is usually accompanied by an estimate of variant frequency, as well as some confidence measures.

5.1. FAIDX INDEXING

The first thing I am going to do is create a new index for the reference genome, that will be used afterwards. To do this, I use the following command:

```
samtools faidx genome.fna
```

When I do this, however, the following error raises: *Different line length in sequence 'NC_000918.1'*, and the indexed file is not created. To check this, I write the following command:

```
awk '/^>/{print;next;}{print length($0);}' genome.fna | uniq
```

The output gives us two lengths, 80 and 1681. In order for *faidx* to work, all the lines must be the same length. To solve this, I create a python script that uses SeqIO to transform our file to FASTA (it is already in FASTA, but this transformation sets the lengths right). To do this, I type the following commands on the shell:

```
python

from Bio import SeqIO

SeqIO.convert("genome.fna", "fasta", "genome_mod.fna", "fasta")

exit()
```

Now, using the file *genome_mod.fna*, the *faidx* command is successful and creates the file *genome_mod.fna.fai*.

5.2. BAM FILES PREPROCESSING

Now comes the preprocessing of our BAM files. I already generated the sorted BAM files in the previous section. Anyways, I will do it again, because I think it makes sense to create a script that both creates the BAM file and the sorted BAM file from a number of SAM files that are samples of the same genome, and then merges them (that is the reason why the script *depth_script.txt*, that calculated the coverage, deleted the BAM files it created). This script is *merge_script.txt*, and to run it I use the command:

```
./merge_script.txt hightemp_01.sam hightemp_02.sam normal_01.sam normal_02.sam
```

This script creates the files *normal.bam* and *hightemp.bam*, which are the merged files. This BAM files have the same fields we saw on section *4.2.*

5.3. BCFTOOLS

Now that we have done the preprocessing, we are ready to call variants. To do this, I will use BCFtools, which is a set of utilities that manipulate variant calls in the variant call format (VCF). This format has the advantage that the information contained in it is not redundant, due to the fact that only the variations need to be stored along with a reference genome (while other formats, such as GFF, store all of the genetic data).

VCF is a file format for storing DNA variation data such as single-nucleotide variants, indels, structural variants (Danecek *et al.*, 2011) … The VCF file includes a header that starts with '##' and a field definition line that starts with '#' that begins the data section.

A VCF file has eight mandatory fields:

1. CRHOM: chromosome where the variant is.
2. POS: 1-based position of the start of the variant.
3. ID: unique identifier of the variant.
4. REF: the reference base (or bases) at the given position on the given position on the given reference allele.
5. ALT: a comma-separated list of alternative nonreference bases of the variation.
6. QUAL: a quality score associated with the inference of the given alleles.
7. FILTER: a flag indicating which of a given set of filters the variation has passed.
8. INFO: A semicolon-separated list of additional information.

9. FORMAT: colon-separated list that defines information in subsequent genotype column. This is mandatory when additional columns are used, and it must always be the first of the optional columns.

BCF, on the other side, is the binary variant call format (the binary version of VCF). Just like BAM files contained the same information as SAM files, BCF files contain the same information as VCF files, but are much more efficient to process (especially when the number of samples is big).

The BCFtools package indentifies variants as follows:

bcftools mpileup -f *[reference genome] [merged BAM file]* -o *[output]* -O b

Where *[reference genome]* is the reference genome I used on the *samtools faidx* command, not the output (even though the exact name of the indexed genome is not written in the command, it must be in the same folder in order for the command to work), *[merged BAM file]* is the BAM files obtained from the script *merge_script.txt*, *[output]* is the desired name for the output and *-O b* indicates that the output must be in a compressed BCF format.

Afterwards, BCFtools can call variants in this way:

bcftools call -vc *[BCF file]* -o *[output]* -O b

Then, using *bcftools stats* I create a txt file with some stats of the created file. The created file is a compressed BCF format (I will use this format because is the lest heavy format, making the processing time less than if I used VCF format); to be able to read it, I use *bcftools view*. Finally, I create another file with some stats obtained directly from the bcf files (although we can find some of them in the previously created stats file); these stats are:

- The number of records: all lines, except for the ones in the header (that start with either '##' or '#', as previously mentioned, are records (each line is a record).
- The number of variants: the different changes that we can find (for example, changing an 'A' for a 'T').
- The number of SNPs, insertions and deletions. We can distinguish these three using the fifth column, which is a single nucleotide in SNPs, a dot in deletions and more than one nucleotide in insertions.
- The number of variants with a depth of coverage equal to or greater than 10.
- The number quality equal to or greater than 10.
- Information regarding the variant with best quality: its position, the gene it belongs to (for this, the GFF file with the genes is needed), the change of bases it entails and its quality.

All of this is done by the script *bcf_script.txt*, which keeps the BCF files in the same folder and creates a new folder for the stat files. As a first argument it takes the reference genome (the *faidx* indexed version must also be in the folder); the second argument must be the GFF file containing the genes, so that the gene ID can be searched automatically, and then as many BAM files as we want.

I introduce the merged BAM files, as well as the sorted BAM files of each sample:

./bcf_script.txt genome_mod.fna genes.gff hightemp.bam normal.bam hightemp_01_sorted.bam hightemp_02_sorted.bam normal_01_sorted.bam normal_02_sorted.bam

The obtained results are shown on *Table 6* and *Table 7*; the latter contains information about the variant with the best quality, which is the exact same in all 6 files, while the former contains the rest of the information.

Table 6. **Number of records, variants, SNPs, indels, variants with coverage equal to or greater than 10 and variants with quality equal to or greater than 10 for the BCF files corresponding to the 4 BAM sorted files, and the two BAM merged files.**

	Number of records	Number of distinct variants	Number of SNPs	Number of indels	Number of variants with coverage >= 10	Number of variants with quality >= 10
***Normal* 1**	397	12	397	0	67	51
***Normal* 2**	418	12	418	0	81	76
***Hightemp* 1**	373	12	373	0	69	82
***Hightemp* 2**	390	12	390	0	78	73
***Normal* merged**	115	12	115	0	18	11
***Hightemp* merged**	134	12	134	0	21	19

Table 7. **Information about the highest quality variant of the BCF files corresponding to the 4 BAM sorted files, and the two BAM merged files.**

Position	Corresponding gene ID	Reference base	Variant's base	Quality
135.330	1192825	A	C	221'999

First, let us take a look at *Table 6*. The first thing to be noticed is the fact that all variants are SNPs in all cases. Thus, we have 12 distinct variants, which correspond to the 12 possible changes of the four bases. The fact that the number of records of the single files about triples the number of records of the merged files is also noticeable. This is probably due to the fact that merged files only contain the common variants to both files. Finally, it is to be noted the fact that the number of records of *normal* and *hightemp* are about the same, so the different conditions do not seem to influence the number of variants.

About *Table 7*, we have got the gene ID, so I can search for it on UniProt (https://www.ncbi.nlm.nih.gov/gene/1192825). This is a protein coding gene called *nifA*, and it is a NifA family transcriptional regulator.

5.4. IGV VISUALIZATION

To gather more information about this SNP, I will use the Integrative Genomics Viewer (IGV) software (Thorvaldsdóttir *et al*., 2012). IGV is a high-performance viewer that allows the visualization of large, heterogeneous data sets. IGV supports the loading of local sequences.

I create a folder that contains the files that I will upload to IGV (the genome in FASTA format, and the BAM merged files under normal and high-temperature conditions) as well as their indexed versions (they need to be on the same folder as the others, even though they do not need to be loaded into IGV). The indexed version of the genome was already created with the command *faidx*; however, I still have to index the BAM files. I do it with the following command:

```
samtools index hightemp.bam

samtools index normal.bam
```

This creates the .bai files needed. Then, we enter the position of our SNP in IGV, and get the result shown on *Figure 3*.

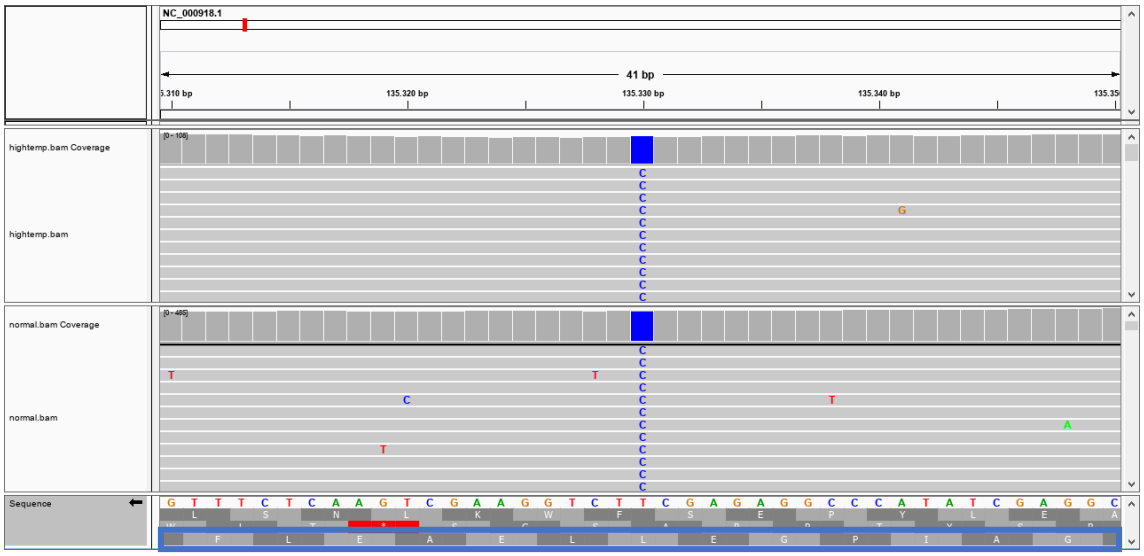

***Figure 3*. Visualization of the higher quality SNP on IGV.** There are two alignments loaded: the one under normal temperature conditions, and the one under high temperature conditions. In blue are the nucleotides of the two alignments in that position ('C' in all cases), and just above them is the reference sequence (which contains a 'T' in that position). At the bottom of the picture is highlighted the correct protein sequence.

As can be seen on this figure, all the sequences in our alignments contain a 'C' in the position of our variant, while the reference genome contains an 'A' (as shown on *Figure 7*). Inside the blue box is highlighted the correct proteic sequence, which can be checked on UniProt as well, in the section *Genomic regions, transcripts and products* (https://www.ncbi.nlm.nih.gov/gene/1192825).

The 'T' in the reference genome forms part of the triplet 'CTT', which encodes lysine. Changing the 'T' for a 'C' creates the triplet 'CCT', which encodes proline. Lysine is an amino acid that tends to form α-helixes, while proline is an amino acid that, due to its structure, gives such a sharp bend to a protein, disrupting helixes. Therefore, depending on the position of the amino acid in the 3D structure of the protein, the structural change might be noteworthy.

6. DIFFERENTIAL EXPRESSION ANALYSIS

The next step involves the comparison the expression differences between the samples grown under normal and high-temperature conditions. In order to do this, I am going to perform a differential expression analysis (DEA).

First, I do a read-count of the sorted BAM files using *htseq-count*, which takes one or more alignment files in SAM/BAM format and a feature file in GFF format and calculates for each feature the number of reads mapping to it. We use the following command:

Htseq-count -i Name -t gene -f bam *[BAM sorted file] [GFF file] > [output file]*

-i Name indicates the GFF attribute to be used as feature ID, *-t gene* indicates the feature type to be used, and *-f bam* indicates the file type of the input files.

The script *htseq_script.txt* performs this command for any number of BAM files given, and then joins all of them. I execute it as follows:

./htseq_script.txt genes.gff hightemp_01_sorted.bam hightemp_02_sorted.bam normal_01_sorted.bam normal_02_sorted.bam

As a result, we obtain a file called *total_counts.txt* that contains on the first column the name of the sequence, on the second column the reads of *hightemp_02*, on the third the reads of *hightemp_01*, on the fourth the reads of *normal_02* and on the fifth the reads of *normal_01*.

Now, I am going to further process this file in R, using the R package *DESeq2* (Love *et al.*, 2014), that allows us to detect differentially expressed genes.

As input, DESeq2 expects count data as obtained, in the form of a matrix of integer values. These values should be un-normalized counts of the sequencing reads of our single-end RNA-seq, because the DESeq2 model internally corrects for library size.

The R script *dea_script.R* does the following:

1. It loads the *total_counts.txt* data, and using it creates a DESeqDataSet object.
2. It removes rows with 0 or 1 reads as a pre-filtering, so that the memory size of the object is reduced and the speed of the posterior functions is increased. However, this only removes 7 rows.
3. It runs the *DESeq* function. This function does three things:
4. The resulting object is a data frame where each row belongs to a different gene:
    - ✓ The first row contains the gene name.
    - ✓ The second row contains the mean of normalized counts for all samples.
    - ✓ The third row contains the log2 fold change (MLE): condition Normal vs High. A value of 2, for example, indicates that the normal-temperature conditions induces a multiplicative change in observed gene expression level of $2^2 = 4$ compared to the high-temperature conditions.
    - ✓ The fourth row contains the standard error: condition Normal vs High.
    - ✓ The fifth row contains the Wald statistic: condition Normal vs High.
    - ✓ The sixth row contains the Wald test p-value: condition Normal vs High.
    - ✓ The seventh column contains the BH adjusted p-values.
5. Orders the results according to the BH adjusted p-values, calculates the number of rows with BH adjusted p-value lower than 0.1 and shows the rows that follow this condition.
6. Plots a histogram of the frequency of genes according to their BH adjusted p-value using the R package *ggplot2*.

As a result, we obtain that four genes are differentially expressed, as shown on *Figure 4*. The data of these genes we are interested in are shown on *Table 8*.

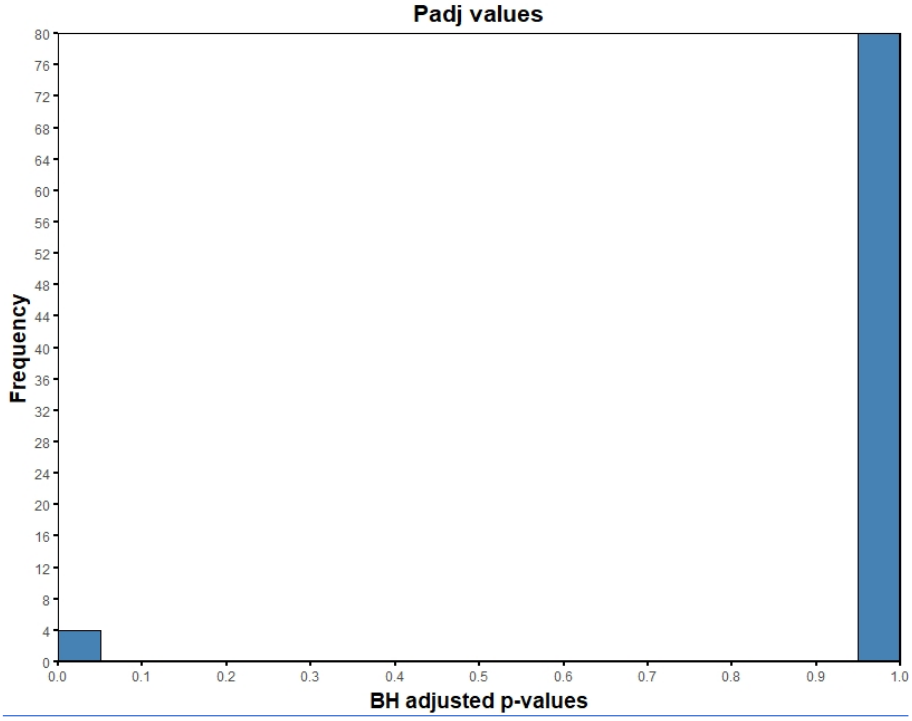

Figure 4. **BH adjusted p-values.** There are four genes with p-values below 0.1; the rest of them (almost 1.500) have values close to 1.

Table 8. **Genes differentially expressed under normal and high-temperature conditions.**

	Log2 Fold change	Multiplicative change in normal-temperature conditions compared to high-temperature conditions	p-value	BH adjusted p-value
NP_213887.1	3'404	10'585	$2'271 * 10^{-154}$	$3'397 * 10^{-151}$
NP_Unk02	-2'295	0'204	$1'58 * 10^{-74}$	$1'182 * 10^{-71}$
NP_Unk01	-2.239	0'212	$3'261 * 10^{-2}$	$1'626 * 10^{-69}$
NP_213724.1	3'011	8'061	$7'064 * 10^{-2}$	$2'642 * 10^{-69}$

Therefore, two of the four genes increase their levels under high-temperature conditions, while the other two decrease their levels. This is consistent with what was shown on *Figure 2*, we could see these exact results (if we check the positions of the known proteins on the GFF file, we can see that they are effectively the same genes). The p-value tells us which is the smallest definition of the threshold of statistical significance at which this result would be statistically significant. The adjusted p-value is the same, but applied to an entire family of comparisons. Since our p-values are extremely small, the confidence in the results is really high.

The two proteins that are highly expressed under normal-temperature conditions are the following:

- NP_213887.1: an alanine—tRNA ligase (https://www.uniprot.org/uniprot/O67323). This protein catalyzes the attachment of alanine to tRNA(Ala).
- NP_213724.1: in turn, this protein is a glyceraldehyde-3-phosphate dehydrogenase (https://www.ncbi.nlm.nih.gov/protein/NP_213724.1). This protein catalyzes the oxidative phosphorylation of glyceraldehyde 3-phosphate to 1,3-bisphosphoglycerate using the cofactor NAD.

7. FUNCTIONAL PREDICTION

The two proteins we are most interested in, the ones overexpressed under high-temperature conditions, are unknown proteins. Since our organisms is almost only present under high-temperature conditions, we are not as interested in the proteins expressed under normal-temperature conditions.

The first thing we have to do is extract the protein sequence. To do this, I use the commands:

```
grep -A 1 NP_Unk01 novel_proteome.faa > NP_Unk01.fas

grep -A 1 NP_Unk02 novel_proteome.faa > NP_Unk02.fas
```

Then, I am going to use BLAST to find candidates for these proteins. BLAST is a tool that finds regions of similarity between biological sequences, comparing nucleotide or protein sequences to databases and calculating its statistical distance (Altschul *et al.,* 1990).

The most important parameters BLAST reports are the following:

- E-value: the number of expected hits of similar quality (score) that could be found just by chance. The smaller the E-value, the better the match. Hits with an E-value smaller than $1e^{-50}$ are of very high quality; hits with E-value smaller than 0'01 can still be considered a good hit-
- Query cover: the percentage of the query sequence aligned to the hit. The smaller the query coverage, less amino acids are being compared, and the chance of error is higher.

The only strain of *Aquifex aeolicus* sequenced seems to be *Aquifex aeolicus* strain VF5; that is the only result we get if we search in UniProt, BLAST and BACMAP Genome Atlas. In fact, I have not been able to find any other strain of the organism.

7.1. NP_Unk01

As a database, I use *Non-redundant protein sequences (nr)*. Then, I enter to which organism this protein belongs to: *Aquifex aeolicus* VF5. Finally, as algorithm I use blastp, which compares protein sequences.

As a result, we get three candidate proteins. Two of them have high E-values, above 2, but the other has an E-value of 0'001. This does not matter, however, because the query cover is below 6%.

Given these results, we cannot identify the protein, so we are going to have to make guesses on its function. I repeat the blast, but this time I do not enter the organism, so that I get hits from any organism in the database.

This gives much better results, with E-values close to 0 and query covers above 94%. Let us start reviewing them.

What I consider to be the best result is the protein AAA family ATPase of *Methanomassiliicoccus luminyensis*, with an E-value of $5e^{-154}$ and a query coverage of 99%. In fact, among the first results, many of them (11 of the twenty first hits) are AAA family ATPase proteins.

AAA proteins (ATPases Associated with diverse cellular activities) are a protein family which share an ATPase domain of about 220 amino acids, highly conserved. There are many activities executed by these proteins, such as membrane trafficking, microtubule regulation…

Looking at its domains, the AAA family ATPase of *M. luminyensis* sees to be a protein that catalyzes the ATP-dependent reduction of dinitrogen to ammonia, because it has a oxidoreductase-nitrogenase domain. The next best hits that are AAA family ATPases (I have checked the six next best hits, belonging to the generus *Methanoplasma, Methanolobus* and *Methanococcoides*) also have the same function.

The other main hit is the nitrogenase iron protein NifH, typical of methanogenic archeas, with roughly the same function.

Therefore, we can assure that the function of this protein is the reduction of dinitrogen to ammonia. There are studies about the nitrogenase activity in thermophilic chemolithoautotrophic bacteria in the phylum aquificae, such as our organism, demonstrating the active nitrogen fixation in thermophilic bacteria at high temperatures (Nishihara *et al.,* 2018).

But, why is this gene so important under high-temperature conditions? We can find the reason by looking at the structure of the nitrogenase enzyme complex. This complex is sensitive to O_2, that irreversibly inactivates the enzyme (Soto & Baca, 2001). Therefore, the oxygen concentration is a truly important factor in the regulation of nitrogenase biosynthesis.

Hot springs' water is hypoxic. This is due to the fact that hot water has less dissolved oxygen than cold water. This is because the hot temperature becomes kinetic energy for the water molecules, which allows the oxygen atoms to break the weak binding forces that form water and scape through the surface. Thus, hot springs provide a perfect environment for the nitrogenase to function.

Therefore, low oxygen concentration will probably induce the production of nitrogenase. Furthermore, there is another condition of hot springs that promotes the production of nitrogenase: hot spring waters are frequently nitrogen-limited (Alcaman *et al*, 2015).

The ability to fix atmospheric nitrogen to ammonia by using the nitrogenase enables diazotrophs to proliferate under conditions of extreme fixed-nitrogen deprivation. Growth in the absence of

fixed nitrogen requires a high concentration of the enzyme. Also, atmospheric fixation of nitrogen to ammonia is a very energy-consuming process; nitrogenase consumes up to 44 mol of ATP per mol of dinitrogen fixed *in vivo* (Martínez-Argudo *et al.,* 2004).

Nitrogen fixation, thus, provides an opportunistic strategy to colonize nitrogen-deficient environments, but the cellular commitment in terms of protein synthesis and ATP consumption is worthy only under low oxygen and fixed nitrogen concentrations. That is the reason why the concentration of this protein is very high in hot springs under extreme conditions.

7.2. NP_Unk02

Again, we start by blasting against the proteins of our organism. The best hit, with a much higher total score than the second one, is the NifA family transcriptional regulator (not the same protein seen on section 5.3). Since we got a very good hit (99% coverage, E-value of 0), we need not do another BLAST, as with the other protein.

We just saw, in the previous section, the importance of regulation of the production of nitrogenase. The enzymes involved in the fixation of atmospheric nitrogen into other forms of nitrogen are encoded by the *nif* genes (both our proteins, NifH and NifA, are encoded by these genes). The signals previously mentioned of low oxygen and fixed nitrogen are integrated to provide transcriptional control of the *nif* gene expression.

The *nif* gene transcription is mostly dependent on the alternative sigma factor σ^{54}. σ^{54} needs a specific class of transcriptional activator, which binds to enhancer-like elements upstream of σ^{54}-dependent promoters and through ATP hydrolysis catalyzes conformational changes in σ^{54} that enable it to start the transcription of genes. One of this transcriptional activator is NifA, that as we have previously explained regulates genes necessary for the synthesis of nitrogenase. NifA has a conserved σ^{54} interaction domain.

The activation of transcription by NifA is regulated in response to oxygen and fixed nitrogen, just as we explained in the previous section. This activates σ^{54}, which in turn results in the transcription of genes of AAA proteins, such as NifH.

7.3. INFLUENCE ON THE BLOOM OF ALGAE

The influence on the bloom algae is clear. As we saw in the introduction, algae can multiply quickly in waterways with an overabundance of nitrogen, particularly when the water is warm. Therefore, the increase in nitrogen produced by *A. aeolicus* sets up the environment for the bloom of algae.

8. PHYLOGENETIC ANALYSIS

As a final step, to delve into the functional roles and the evolutionary origin of our upregulated genes under high-temperature conditions, I am going to perform a phylogenetic analysis comparing them against other prokaryotic proteomes (including the reference genome of the species matching our isolated strain in the hot spring).

The first step in this analysis is running a blast search for each of the two over expressed proteins against all reference proteomes, extracting only the hits with an E-value equal to or smaller than 0’00001. To do this, previously a BLAST database is generated from a FASTA file containing the proteomes we are interested in. Then, I create a FASTA file with all the sequences of selected hits, using the script *extract_sequences_from_blast_result.py* (already given to me). The blast command is used setting the output format to 6, which is the one that works with the python script.

Then, using Clustal Omega, I perform a multiple sequence alignment. Clustal Omega is “a multiple sequence alignment program that uses seeded guide trees and HMM profile-profile techniques to generate alignments between three or more sequences” (Madeira *et al.,* 2019).

All of this is done with the script *phylo_script.txt*. This script is run with the command shown below:

```
./phylo_script.txt all_ref_proteomes.faa
```

This produces as outputs, among others, the files *NP_Unk01_blast.alg* and *NP_Unk02_blast.alg*. Then, we finally construct the tree using *iqtree*, which is a software for phylogenetic inference (Naser-Khdour *et al.*, 2019). This step was not included in the script, due to the high calculation time. The commands are:

```
iqtree -s NP_Unk01_blast.alg

iqtree -s NP_Unk02_blast.alg
```

The final step is the visualization of the obtained trees. To do it, we use the Python package *ETE Toolkit* (Huerta-Cepas *et al.*, 2016). We label the nodes properly.

On *Figure 5* is shown the tree we obtained for Unknown_01. The closest protein is an unknown protein form the NifH/frxC family ('4Fe-4S iron sulfur cluster binding proteins, NifH/frxC family, from *Methanosarcin_acetivorans*). In general, most of the known proteins are from the nifH family, as we predicted in the previous section.

As for the species, the genes are from different species; if the tree were made from genes of the same species, ortholog inference would not be possible. The most common one seems to be *Methanosarcina acetivorans*, which has many duplicated genes.

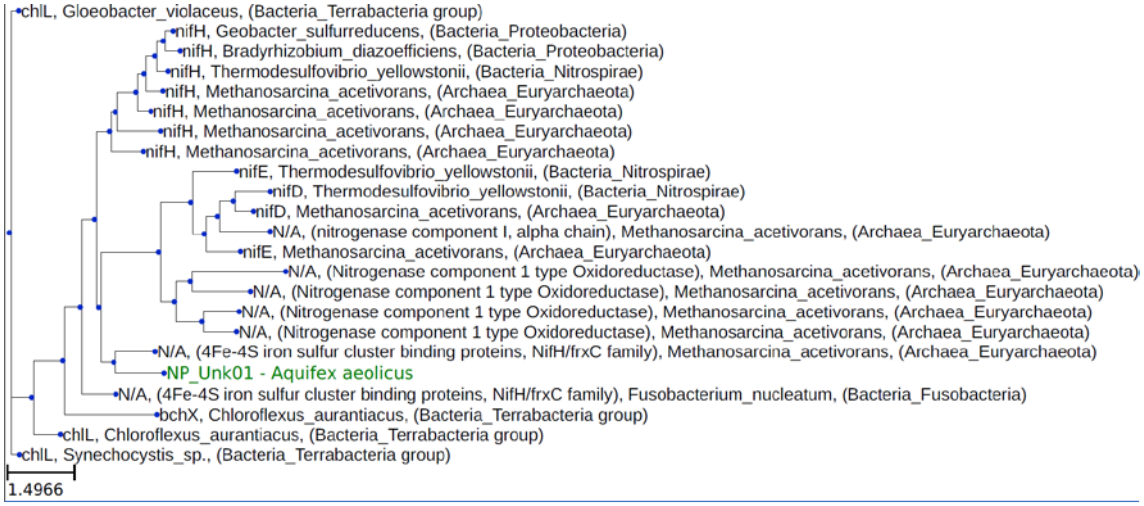

Figure 5. **Phylogenetic tree obtained for Unk01.** The whole tree is shown.

On *Figure 6* is shown part of the tree for NP_Unk02, the part closest to it. The closest protein is norR, from Aquifex aeolicus. However, since this is from the same species, it cannot be considered an ortholog. The closest ortholog is nifA, from *Bradyrhyzobium diaoefficiens*. Again, this concurs with what was previously expressed.

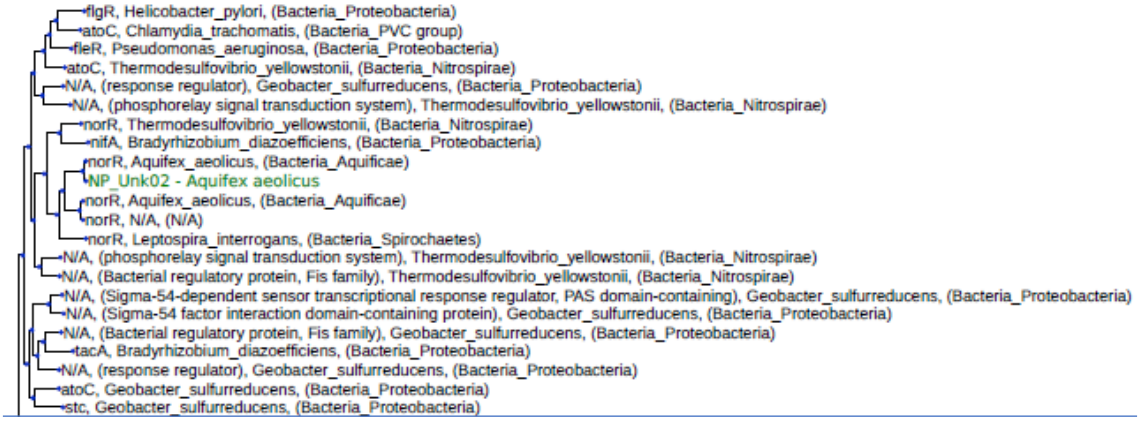

Figure 6. **Phylogenetic tree obtained for Unk02.** Only the closest region to Unk02 is shown.

9. CONCLUSION

The conclusions of the study are the following:

1. Under high-temperature conditions, the organism *Aquifex aeolicus* inhabits the environment.
2. Two genes of *A. aeolicus* are over-expressed under high-temperature conditions.
3. These genes are related to nitrogen fixation, which allows the bloom of algae that happens after the high-temperature episodes (when the temperature is around 60ºC).

REFERENCES

Alcaman, Maria & Fernandez, Camila & Delgado Huertas, Antonio & Bergman, Birgitta & Diez, Beatriz. (2015). The cyanobacterium Mastigocladus fulfills the nitrogen demand of a terrestrial hot spring microbial mat. The ISME journal. 9. 10.1038/ismej.2015.63.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990) "Basic local alignment search tool." J. Mol. Biol. 215:403-410.

Channing, Alan & Wujek, Daniel. (2010). Preservation of protists within decaying plants from geothermally influenced wetlands of Yellowstone National Park, Wyoming, United States. PALAIOS. 25. 347-355. 10.2110/palo.2009.p09-057r.

Channing, Alan. (2018). A review of active hot-spring analogues of Rhynie: environments, habitats and ecosystems. Philosophical Transactions of the Royal Society B: Biological Sciences. 373. 20160490. 10.1098/rstb.2016.0490.

Deckert, Gerard & Warren, Patrick & Gaasterland, Terry & Young, William & Lenox, Anna & Graham, David & Overbeek, Ross & Snead, Marjory & Keller, Martin & Aujay, Monette & Huber, Robert & Feldman, Robert & Short, Jay & Olsen, Gary & Swanson, Ronald. (1998). The Complete Genome of the Hyperthermophilic Bacterium Aquifex aeolicus. Nature. 392. 353-8. 10.1038/32831.

Danecek, P., Auton, A., Abecasis, G. *et al.* 2011. The variant call format and VCFtools. *Bioinformatics* 27(15), 2156–2158. PMID: 21653522.

Freeman, Jennifer & Perry, George & Feuk, Lars & Redon, Richard & Mccarroll, Steven & Altshuler, David & Aburatani, Hiroyuki & Jones, Keith & Tyler-Smith, Chris & Hurles, Matthew & Carter, Nigel & Scherer, Stephen & Lee, Charles. (2006). Copy number variation: New insights in genome diversity. Genome research. 16. 949-61. 10.1101/gr.3677206.

Guiral, Marianne & Prunetti, Laurence & Aussignargues, Clément & Ciaccafava, Alexandre & Infossi, Pascale & Ilbert, Marianne & Lojou, Elisabeth & Giudici-Orticoni, Marie. (2012). The Hyperthermophilic Bacterium Aquifex aeolicus: From Respiratory Pathways to Extremely Resistant Enzymes and Biotechnological Applications. Advances in microbial physiology. 61. 125-94. 10.1016/B978-0-12-394423-8.00004-4.

Huerta-Cepas J., Serra F., Bork P. (2016) ETE 3: Reconstruction, análisis and visualization of phylogenomic data. Mol Biol Evol.

Illumina Sequencing. 2009. Genomic sequencing. http://www.illumina.com/applications/sequencing/rna.ilmn. Accessed January 09, 2020.

Johnson, Nathan & Yeoh, Jonathan & Coruh, Ceyda & Axtell, Michael. (2016). Improved Placement of Multi-mapping Small RNAs. G3 (Bethesda, Md.). 6. 10.1534/g3.116.030452.

Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. *Nature Methods*. 2012, 9:357-359.

Li H.*, Handsaker B.*, Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. [PMID: 19505943]

Love M., W. Huber, S. Anders: Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 2014, 15:550.

Madeira F.; Park Y. mi; Lee J.; Buso N.; Gur T.; Madhusoodanan N.; Basutkar P.; Tivey A. R. N.; Potter S. C.; Finn R. D.; Lopez R. The EMBL-EBI Search and Sequence Analysis Tools APIs in 2019. Nucleic Acids Res. 2019, 47 (W1), W636–W641. 10.1093/nar/gkz268.

Martinez-Argudo, Isabel & Little, Richard & Shearer, Neil & Johnson, Philip & Dixon, Ray. (2004). The NifL-NifA System: A Multidomain Transcriptional Regulatory Complex That Integrates Environmental Signals. Journal of bacteriology. 186. 601-10. 10.1128/JB.186.3.601-610.2004.

Milanese A., Mende D. R., Paoli L., Salazar G., Ruscheweyh H. J., Cuenca M., Hingamp P., Alves R. Costea P., Coelho L., Schmidt T. S. B., Almeida A., Mitchell A:, Finn R. D., Huerta-Cepas J., Bork P., Zeller G. (2019) Microbial abundance, activity and population genomic profiling with mOTUs2. Nature Communications, 10.

Naser-Khdour S., B.Q. Minh, W. Zhang, E.A. Stone, R. Lanfear (2019) The prevalence and pmpact of model violations in phylogenetic analysis, *Genome Biol. Evol.*, in press.

Nielsen, R., Paul, J.S., Albrechtsen, A., Song, Y.S. 2011. Genotype and SNP calling from next-generation sequencing data. *Nature Reviews Genetics* 12(6), 443–451. PMID: 21587300.

Nishihara, Arisa & Haruta, Shin & Thiel, Vera & McGlynn, Shawn. (2018). Nitrogenase Activity in Thermophilic Chemolithoautotrophic Bacteria in the Phylum Aquificae Isolated under Nitrogen-Fixing Conditions from Nakabusa Hot Springs. Microbes and Environments. 33. 10.1264/jsme2.ME18041.

Pellegrini, M. (2012). Using Phylogenetic Profiles to Predict Functional Relationships. Methods in molecular biology (Clifton, N.J.). 804. 167-77. 10.1007/978-1-61779-361-5_9.

Pollack, Jonathan & Perou, Charles & Alizadeh, Ash & Eisen, Michael & Pergamenschikov, Alexander & Williams, Cheryl & Jeffrey, Stefanie & Botstein, David & Brown, Patrick. (1999). Genome-wide analysis of DNA copy-number changes using cDNA microarrays. Nature genetics. 23. 41-6. 10.1038/12640.

Robert J.M. Eveleigh, Conor J. Meehan, John M. Archibald, Robert G. Beiko, Being *Aquifex aeolicus*: Untangling a Hyperthermophile's Checkered Past, *Genome Biology and Evolution*, Volume 5, Issue 12, December 2013, Pages 2478–2497

Rozanov A., Bryanskaya A., Ivanisenko T., Malup T., Peltek S. (2017). Biodiversity of the microbial mat of the Garga hot spring. BMC Evolutionary Biology.

Soto, Lucía & Baca, Beatriz. (2001). [Mechanisms for protecting nitrogenase from inactivation by oxygen]. Revista latinoamericana de microbiología. 43. 37-49.

Thorvaldsdottir, H., Robinson, J.T., Mesirov, J.P. 2012. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics* 14(2), 178–192. PMID: 2251747.

Trost, Brett & Walker, Susan & Wang, z & Thiruvahindrapuram, Bhooma & Macdonald, Jeffrey & Sung, Wilson & Pereira, Sergio & Whitney, Joe & Chan, Ada & Pellecchia, Giovanna & Reuter, Miriam & Lok, Si & Yuen, Ryan & Marshall, Christian & Merico, Daniele & Scherer, Stephen. (2018). A Comprehensive Workflow for Read Depth-Based Identification of Copy-Number Variation from Whole-Genome Sequence Data. The American Journal of Human Genetics. 102. 142-155. 10.1016/j.ajhg.2017.12.007.

Winterbourn M. J. (1969) The distribution of algae and insects in hot spring thermal gradients at Waimangu, New Zealand, New Zealand Journal of Marine and Freshwater Research, 3:3, 459-465.