University of South Bohemia in České Budějovice

Faculty of Science

and

Johannes Kepler University in Linz

Faculty of Engineering and Natural Sciences

Different approaches of metatranscriptome assembly

Bachelor Thesis

Author : Anna Drechslerová

Supervisor: Ing. Jiří Bárta, Ph.D.

České Budějovice 2020

Drechslerová, A., 2020: Different approaches of metatranscriptome assembly. Bc. Thesis, in English -48p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic and Faculty of Engineering and Natural Sciences, Johannes Kepler University, Linz, Austria.

Annotation

This paper was created by analysing 24 samples produced by Next-Generation Sequencing (Illumina). These samples were acquired from different places in Arctic permafrost as a part of MiCryoFun and CRYOvulcan project. The tested data contains forward and reverse sequences which always form pairs for given sample. Four methods were used to evaluate the quality of our samples FASTQC, FLASH, PRINSEQ and TRIMMOMATIC. This test was followed by comparing the work of different assemblers - concretely SPAdes, Megahit and Velvet. The data was then assessed by Quast. As a last step the results were blasted using Diamond and the final organisms classified.

I hereby declare that I have worked on my bachelor thesis independently and used only the sources listed in the bibliography. I hereby declare that, in accordance with Article 47b of Act No. 111/1998 in the valid wording, I agree with the publication of my bachelor thesis, in full form resulting from deletion of indicated parts to be kept in the Faculty of Science archive, in electronic form in a publicly accessible part of the IS STAG database operated by the University of South Bohemia in České Budějovice accessible through its web pages.

Further, I agree to the electronic publication of the comments of my supervisor and thesis opponents and the record of the proceedings and results of the thesis defence in accordance with aforementioned Act No. 111/1998. I also agree to the comparison of the text of my thesis with the Theses.cz thesis database operated by the National Registry of University Theses and a plagiarism detection system.

In České Budějovice, 20.5.2020

.....

Signature

Acknowledgements

I would like to take this opportunity to thank my supervisor Ing. Jiří Bárta, Ph.D. who enabled me to work on this interesting project and provided guidance and expertise throughout the whole time. I would also like to express my gratitude to Jiří Pech Ph.D. whose great work and assistance was very useful, whenever any server problems occurred. Dr. med. univ. Dr. rer. nat. Gundula Povysil, MSc and Univ. Prof. Dr. Sepp Hochreiter, whose course of sequence analysis not only improved my programming skills in Python, but it also further sparked my interest in bioinformatics.

1. Introduction 1
1.1. Quality control, filtering and trimming of FASTQ reads 1
1.2.Assembly 2
1.2.1.De Bruijn Graph (DBG) based Assemblers 2
1.2.2.Greedy-Approach Assemblers 2
1.2.3.Overlap-Layout-Consensus (OLC) Assemblers 3
1.2.4.Greedy Graph-based Assemblers 3
1.2.5.Quast 4
1.3. Annotation 4
1.4. Selecting of samples 5
2. Materials and Methods 5
2.1. FastQC (Version 0.11.8; Andrews, 2010) 5
2.2. FLASH(Version 1.2.11; Magoc & Salzberg, 2011) 5
2.3. Prinseq (Version 0.20.4; Schmieder, & Edwards, 2011) 6
2.4. TRIMMOMATIC (Version 0.39; Bolger, Lohse, and Usadel, 2014) 6
2.5. SPAdes (Version 3.12.0; Bankevich et al., 2012) 6
2.6. MEGAHIT (Version 1.2.9; Li et al., 2015) 6
2.7. VELVET(Version 1.2.10; Zerbino& Birney, 2008) 7
2.8. QUAST(Version 5.0.2.; Gurevich et al., 2013) 7
2.9. DIAMOND (Version v0.9.25.126; Buchfink et al., 2015) 7
3. Results and Discussion 8
3.1. Quality control, filtering and trimming 8
3.1.1.FastQC 8
3.1.2.FLASH 20
3.1.3.Prinseq 21
3.1.4.TRIMMOMATIC 22
3.2.Assembly and its evaluation 23
3.2.1.Statistics after FLASH 23
3.2.2.Statistics after Prinseq 24
3.2.3.Statistics after Trimmomatic 24
3.2.4.Overall comparison 26
3.2.4.1.Number of contigs 26

3.2.4.2.Longest contig 27
3.2.4.3.Total length 28
3.3.Alignment and classification 29
4. Conclusion 31
5. Bibliography.....32
6. Supplement Content.....35

1. Introduction

This paper works with samples acquired from Arctic permafrost and aims to evaluate the importance of quality control, evaluate several assembly algorithms and in the end identify the content of our samples. For the accurate predictions of the activities (e.g. CO_2, N_2O and CH_4 emissions) of different microbial assemblages we need to precisely evaluate the sequencing data. The process of sequence analysis can be divided into three following subcategories:

1.1. Quality control, filtering and trimming of FASTQ reads

As the initial quality statistics and quality control was selected generally accepted FASTQC algorithm, which provided quick and broad analysis with many different evaluation criteria. The process continued with Fast Length Adjustment of Short reads (FLASH). This program was utilised to merge overlapping sequences and provided additional quality analysis. The data resulting from FLASH was then quality filtered and trimmed by PRINSEQ and TRIMMOMATIC.

The previous point is closely connected to sequence trimming problematics, which is an important part of this paper. In general, trimming is a process of removing parts of sequenced data with the aim to acquire sequences of better quality. In our case, there can be many occurrences of unwanted figures such as poly-N tails, vectors, duplicates, artefacts, introns etc. that are unnecessary and often even harmful towards further analysis (Arbor, 2008). Many of these errors can be created by sequencing (in this case Illumina) or contamination of the sample. In case sequences would be left untrimmed the work with them might be more time consuming (since the size of our data will be unnecessary large). It can also result in alignment errors, because the unwanted artefacts can create additional patterns. Quality control and trimming allow us to identify and exclude the unwanted sequences of our samples. The reduction or deletion of these sequences makes it possible to find the real variety in the dataset.

Different programs (e.g. Prinseq) offer various options for trimming for example cutting off tails, sequence ends, specific data patterns, length or based on quality score (Andrews, 2010). The data analysis could be inaccurate without applying trimming methods to the sequences. Therefore this step is a very important part of the sequence quality analysis process.

1.2.Assembly

In this work three assemblers were applied on the samples, namely SPAdes, Megahit and Velvet. Selecting an adequate assembler can be a difficult task especially considering the number of programs to choose from. There are different groups of assemblers, categorised by their methods.

1.2.1.De Bruijn Graph (DBG) based Assemblers

Assemblers working on the bases of de Bruijn graphs are very popular nowadays (Miller, Koren, and Sutton, 2010). The DBG method first selects a certain k-mer and then it gathers all possible k-mers of the said length out of our sequences. Once those results are acquired it compares the occurrence of identical k-mers across the reads. This then leads to building an assembly.

All three of our selected assemblers are based on DBG method, however the remaining specifications differ. SPAdes has a great advantage as it is able to create single-cell and multicellular assemblies (Bankevich *et al.*, 2012). This method can be compared to e.g. SoapDeNovo2, which also works with DBG and was created to assemble short reads originating from Illumina GA (Luo *et al.*, 2015). Megahit can relatively fast go through large sets of data and deliver complex assemblies with comparatively long contigs (Li *et al.*, 2015). Velvet is a bundle of algorithms, which works with primarily short k-mers and reads(Zerbino, & Birney, 2008). There are many other examples of DBG based assemblers - another popular one is Trinity. This assembler focuses on transcriptomic data originating from RNA sequences and is also widely popular (Grabherr *et al.*, 2011).

1.2.2.Greedy-Approach Assemblers

The principle of these assemblers is quite straightforward : A random sequence is chosen from samples and all the remaining sequences attempt to match it. If a match is found the sequences are added to a bundle together with the original sequence. Once there are no other possible matches the program labels this collection of sequences as a contig. The contents of such a contig are omitted from further queries as they already belong to one contig. After that the program moves on to match the remaining samples. This approach is not often used as it is very naive and trivial. It has many issues - one of them being its inability of working with microbial genomes.

As an example of such an assembler can serve TraRECo, which works with transcriptomic data and its arguably main advantage is the usage of a consensus matrix as means for read error correction (Yoon *et al.,* 2018)

1.2.3.Overlap-Layout-Consensus (OLC) Assemblers

These methods map our sequences and determine connections between them. This results in an overlap graph, which is basically a potential arrangement of the reads together (Miller *et al.*, 2010). This approach can be efficient when dealing with long reads, however it can be problematic when it comes to its speed. Many of these assemblers work slowly, which may be caused for example by the multiple sequence alignments they need to perform. This is one of the reasons why OLC assemblers are not often used on sizeable datasets nowadays and were up to certain degree replaced by DBG method.

Newbler can be selected as the first example of OLC algorithms. This program uses three modules (overlapped, unitigger and multi aligner) as well as flowgrams in order to achieve the desired result (Elloumi, & Zomaya, 2014).

Omega, is another example of OLC. It is mostly used to work with metagenomic samples originating from Illumina (Haider *et al.*, 2014). Last but not least, Celera is worth mentioning. This algorithm performs a whole-genome shotgun assembly (Denisov *et al.* ,2008). This approach works with overlapping reads parallel, which improves its speed (Adams, 2008). However, it could get into problems with samples containing many repeating regions(Adams, 2008). Luckily, there exists a hybrid version of this method, which creates clones of overlapping reads and concludes by presenting an arranged layout scheme of our samples (Adams, 2008). This assembler also serves as a basis together with Newbler for CABOG assembler (Elloumi, & Zomaya, 2014).

1.2.4.Greedy Graph-based Assemblers

These algorithms utilise approaches common in OLC and DBG Assemblers (Miller *et al.*, 2010). The main idea is to identify two sequences with the highest overlap score and merge them. This is achieved by first evaluating a pairwise alignment of our reads, which then leads us to the overlap score. The graphical methods is applied in a way where reads are forming nodes and overlaps create edges. First selected example is SSAKE (Short Sequence Assembly by K-mer search and 3' read Extension; Warren, Sutton, Jones, and Holt, 2006). This tool is PERL based and it utilises a hash table filled with short reads (Warren *et al.*, 2006). SHARCGS (Dohm, Lottaz, Borodina, and Himmelbauer, 2007) and VCAKE (Verified Consensus Assembly by K-mer Extension, Jack *et al.*, 2007) are also assembler belonging to this group. Their work is similar as the previously mentioned algorithm and they primarily assemble short reads.

1.2.5.Quast

As was mentioned before, one of our goals in this paper was to evaluate performance of different assemblers. For this purpose Quast was selected. This assessment tool provides many statistical data to choose from, when applied on the assembly. There are e.g. amounts of contigs in different bp ranges, the larges contig located, total length (again also with different bp ranges) , N50, GC%, mismatches and more (Gurevich, Saveliev, Vyahhi, and Tesler, 2013). Based on results from Quast additional statistics were performed together with a comparison of the used assemblers.

1.3. Annotation

As a last part of the pipeline Diamond was used to align our sequences. This tool was selected, because it can achieve results much faster than regular BLAST (Official Diamond Documentation written by Buchfink, Xie and Huson in 2015 presents a comparison for pairwise alignment of proteins and translated DNA resulting in 500x-20,000x speed of BLAST.). This step resulted in tabulated files in .m8 format which contained among other things, code of our sample, accession number, e-values and other statistical data. Since the goal was to also identify the source genes the samples were blasted in order to acquire this information. This step resulted in files containing again sample codes, accession numbers and e-values, however it also included gene definitions and source organisms. When dealing with the problematics of classification of organisms and genes NCBI Entrez is worth mentioning. This method was tested and it certainly proved very efficient for shorter files. However, since our testing contained tens of thousands of samples there was a need for selection of another system as NCBI Entrez would be slow and this amount of data would overload the server.

Our work continued by writing a python script. This program simply went through the file and gathered names of the source organisms as well as counted the occurrence of said organisms together with the corresponding percentages and printed these results into a table. The outcome gave us a good overview of contents of our samples as well as the potential distribution of the organism representation.

The organisms from which our sequences originate are presumably sensitive to changes of their environment, therefore the current trend of rising temperatures could significantly affect microorganisms inhabiting arctic cryosols and their activity reflected in distinct gene expression patterns (i.e. microbial metatranscriptome).

1.4. Selecting of samples

Last but not least, sample selection was important part of this paper, as it directly influences the results. The aim was to display as much variety as possible and therefore the samples scoring the best and the worst were picked to demonstrate the outcomes of our testing.
First, the file 16NQ040_JB25_ACTGAT_L004_R2_001 (“B”) acquired the highest score in overall FASTQC evaluation and represents the best of our samples.
Secondly, the file 116NQ040_JB26_ATGAGC_L004_R2_001 (“W”) is presented. This sample was a direct opposite of the previously mentioned one and it was evaluated as the worst out of our dataset.
As was mentioned earlier, this contrast provides a good option for comparing our data as well as the tested methods and that is why samples B and W were used. In the enclosed supplement statistics number-based system for naming samples was used and therefore the data connected to sample B can be found under number 25 and sample W under 26 (this is a direct relation to the original codes of the used files).

2. Materials and Methods

2.1. FastQC (Version 0.11.8; Andrews, 2010)

This program provides quick analysis of data and delivers results for eleven different categories of evaluation. Among these categories we can find for example per base sequence quality or per sequence GC content . As input files in this project FASTQ files were used. The analysis is later delivered in zip and html format.

2.2. FLASH(Version 1.2.11; Magoc & Salzberg, 2011)

Fast Length Adjustment of Short reads (FLASH) is a program allowing us to acquire merged overlapping pairs of sequences from our samples.As an input file FASTQ format is used.The resulting merged sequence is prolonged to the length of the original sequence.The algorithm should examine all possible options of overlapping the sequences and present the best possible result.

2.3. Prinseq (Version 0.20.4; Schmieder, & Edwards, 2011)

This program is used for filtering sequences based on various parameters - such as length or quality score. It is also able to trim or alter sequences. It provides results in form of a summary of the main filtering criteria.The input file can be in FASTA or FASTQ format, the latter was used in this project. The output in this case was also a FASTQ format.

2.4. TRIMMOMATIC (Version 0.39; Bolger, Lohse, and Usadel, 2014)

Trimmomatic is a useful tool, which offers many different options for trimming of sequences.
One of them is ILLUMINACLIP, which removes all Illumina related parts from the sample. Another example is SLIDINGWINDOW, which removes parts below set quality average. Other useful options are e.g. LEADING and TRAILING, both remove bases in case the fall under a set quality level (the first from the start of sequence the latter from the end), or MINLEN, where reads are omitted once they do not reach a set length). The input file can be FASTQ or FASTA file (also can be in compressed formats). The output is usually compressed FASTQ file (fq.gz).

2.5. SPAdes (Version 3.12.0; Bankevich *et al.*, 2012)

SPAdes is a genome assembler based on the De Bruijn graph method with many options for usage. There are many pipelines to choose from, e.g. in case of transcriptomic RNA data we can select rnaSPAdes and for metagenomci data metaSPAdes. It also contains a tool for rectifying read errors originating from Illumina called BayesHammer. It supports input files from e.g. Illumina, Nanopore, Sanger or IonTorrent in a form of FASTA, FASTQ or BAM. The output can also have many formats, e.g compressed FASTQ (fastq.gz), FASTG or GFA.

2.6. MEGAHIT (Version 1.2.9; Li *et al.*, 2015)

This assembler also utilised De Bruijn graphs and it is used for sizeable metagenomic samples. It is able to work with different arguments connected to e.g. k-mer size, pruning etc.
Input can be in the format of FASTA/FASTQ,plain text or certain compressed formats. The output can be also e.g. in FASTA/FASTQ format.

2.7. VELVET(Version 1.2.10; Zerbino& Birney, 2008)

This assembler is used for further sequence analysis. Its most important advantages are its speed and wide usage. On the other hand this tool is able to produce only consensus sequences and problems might occur when we use error models such as e.g. IonTorrent (based on information provided by Geneious Prime)[7]. In our case the input files were results of FLASH(Version 1.2.11; Magoc & Salzberg, 2011) , Prinseq (Version 0.20.4; Schmieder, & Edwards, 2011) and Trimmomatic (Version 0.39; Bolger *et al.*, 2014) analysis delivered in FASTQ format.

2.8. QUAST(Version 5.0.2.; Gurevich *et al.*, 2013)

This tool is able to evaluate the quality of assemblers based on different criteria. Quality Assessment Tool (QUAST) is able to gather a relatively vast statistics for individual assemblers containing information about contigs, sequence length, N50, N75, GC%, L50, L75 and other categories. The input can be based, similarly like in case of SPAdes, on e.g. Illimina or Nanopore, where the files are in FASTA,FASTQ, BAM or other formats as well as their compressed forms. The output can be generated in many different forms e.g. a text file(.txt), tab-separated file (.tsv), a file compatible with LaTex (.tex) or PDF.

2.9. DIAMOND (Version v0.9.25.126; Buchfink *et al.*, 2015)

DIAMOND is an alignment tool, which uses double indexing in order to increase its blasting abilities . Thanks to this principle the algorithm is performing very well in speed as well as sensitivity when it comes to comparison e.g. with BLASTX. The input file can be again in FASTA and the output is in a tabulated .m8 format. The program needs to work together with a database, which in our case was a diamond nr in dmnd format.

3. Results and Discussion

3.1. Quality control, filtering and trimming

3.1.1.FastQC

FASTQC allowed us to acquire 11 categories for evaluation of our samples, which we decided to discuss in detail in the following pages. A table summarising problematic signs was created as well and its contents are going to be addresses in the pertinent categories (Supplement :Tab. A-1 .).

3.1.1.1. Basic Statistics

Provides elementary information about our samples as shown in Fig.1 and Fig.2. Based on the file characteristics the program is able to presume type of encoding and analysis the data based on this information. In this case the presumed encoding corresponds to Sanger/Illumina sequencing. In our dataset the kind of encoding should be the same for all sequences. Other parameters vary among sequences (for example GC% is unique for each sequence) however sequence length should be always the same.

Measure	Value
Filename	16NQ040_JB25_ACTGAT_L004_R2_001.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	10625763
Sequences flagged as poor quality	0
Sequence length	150
%GC	53

Figure 1. Basic Statistics for B

Measure	Value
Filename	16NQ040_JB26_ATGAGC_L004_R2_001.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1635208
Sequences flagged as poor quality	0
Sequence length	150
%GC	28

Figure 2. Basic Statistics for W

3.1.1.2. Per base sequence quality

Provides a graphical representation of per base sequence quality of a sample. The graphs in this category are always divided into three levels of sequence quality - green, orange and red. The yellow objects stand for inter-quartile range and each of them contain a red line which represents median value.Whiskers stand for 10%(upper) and 90%(lower) points. There is a blue line plowed throughout the graph which describes mean quality of the sample.

Based on this information we can evaluate the two graphs below (Fig.3 , Fig.4). The per base sequence quality of B is generally good since all inter-quartile ranges are located in the green level ,some whiskers also sink into orange level but none of the bases is present in the lowest level of quality. Also the median value in most cases is on the very top of the graph and mean quality stays more-less constant throughout majority of the graph.

The per base sequence quality of W is very different and is worse than sequence of B. The inter-quartile ranges are partially located in the orange level of quality and many of their whiskers are sinking into the lowest (red) quality level. Mean quality is not constant and it is mostly decreasing in the second half of the graph. However, the sequence quality is in general not too poor and can still provide valuable information. Moreover, the quality value mainly refers to the sequencing and therefore high quality sequence doesn't necessarily have to be correct - other tests must be conducted before it is clear that the dataset is of a good quality.

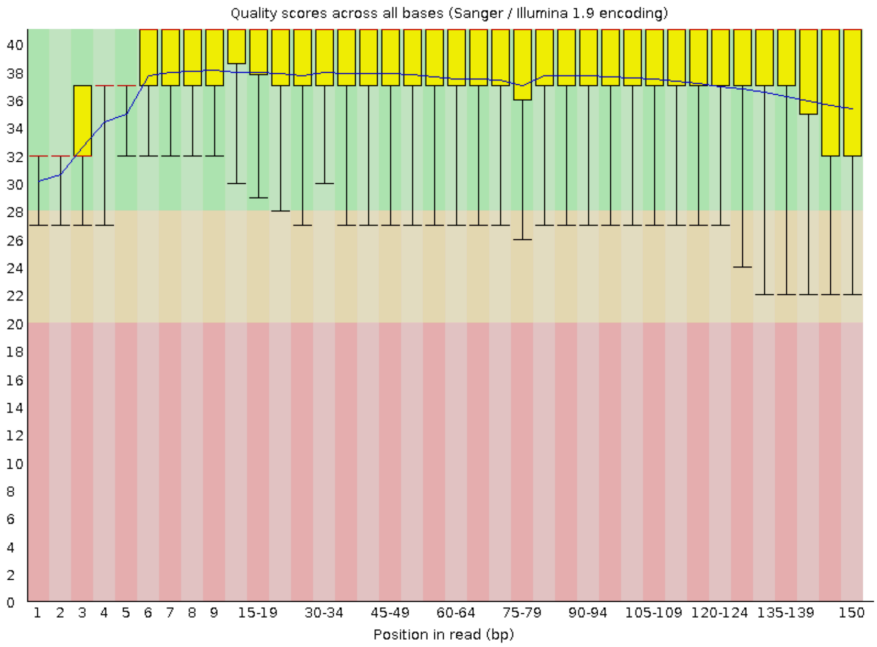

Figure 3. Per base sequence quality of B

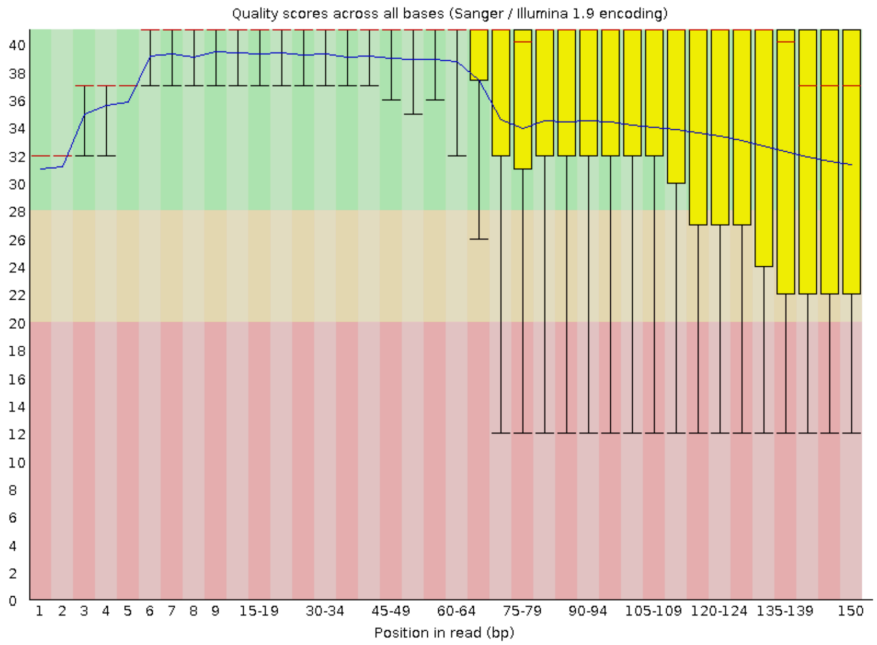

Figure 4. Per base sequence quality of W

3.1.1.3. Per tile sequence quality

Provides a graphical representation of per tile sequence quality of a sample. In this category both sequences have quite good value and shouldn't be considered poor. Per tile sequence quality of B is a bit lower as can be seen by the brighter colouring on the right side of Fig.5 in contrast to non-problematic results for W (Fig.6). B acquired a warning in this case. Based on the information acquired from the official FASTQC documentation written by Andrews in 2010 this is due to the fact that scores of any of the tiles are showing a mean Phred score exceeding 2 less comparing to the mean for that base across all tiles. The displayed problems could also be caused by bubbles, flowcell contamination or other temporary issues.

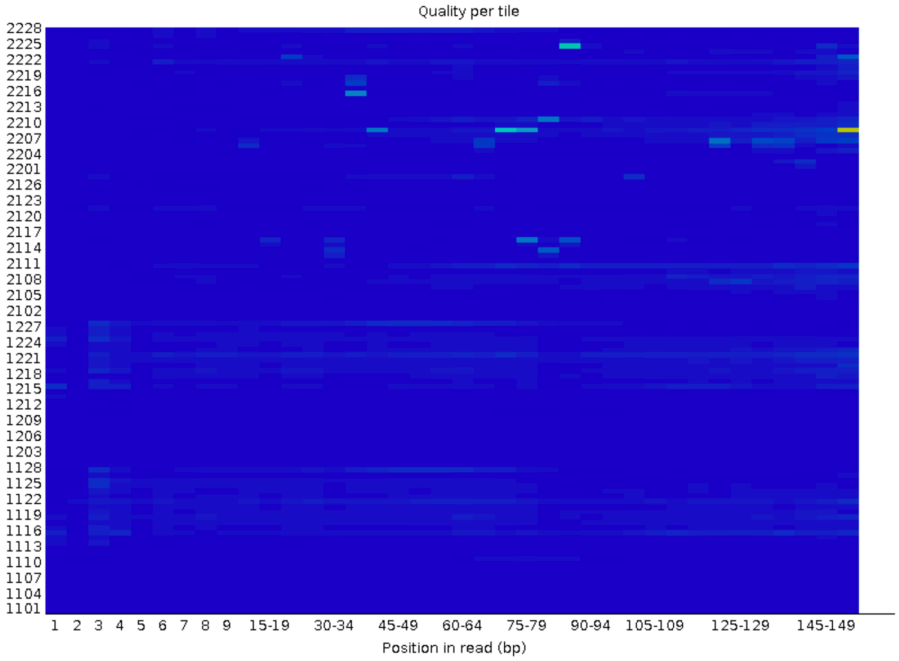

Figure 5. Per tile sequence quality of B

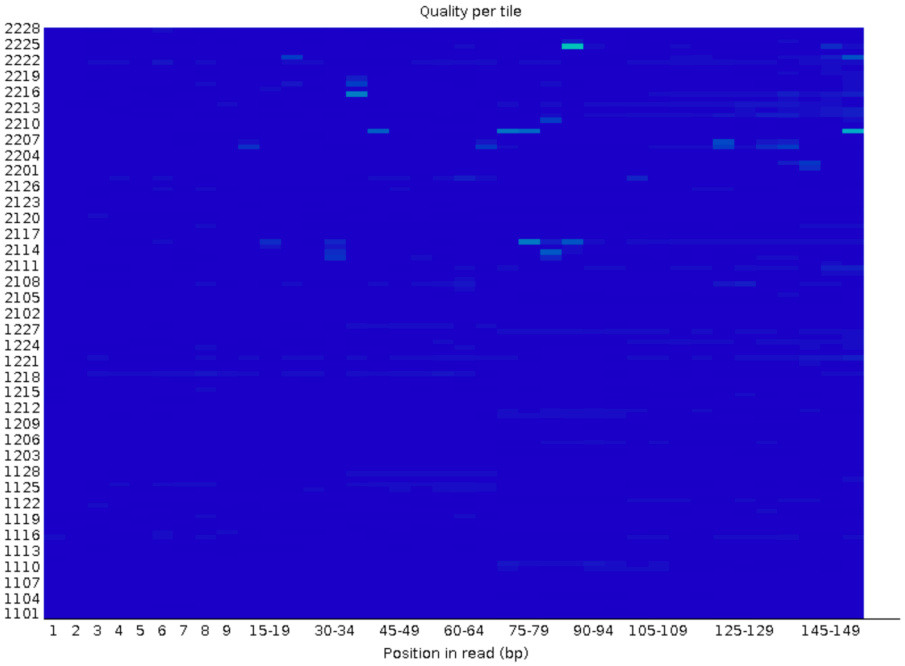

Figure 6. Per tile sequence quality of W

3.1.1.4. Per sequence quality scores

Provides a graphical representation of per sequence quality scores of a sample. In this category the two samples have again different results. Sample B has a quite good score, despite a small abnormality in the middle of the graph (around mean sequence quality value 25 in Fig.7) the graph follows prescribed curve and therefore resembles a good quality dataset. However sample W has more significant abnormality (again around mean sequence quality value 25 in Fig.8.). This doesn't have to present an issue with our data. This problematic region could be caused by a systematic problem e.g. there could have been an air bubble present on a certain part of a flow-cell or an other temporary problem .

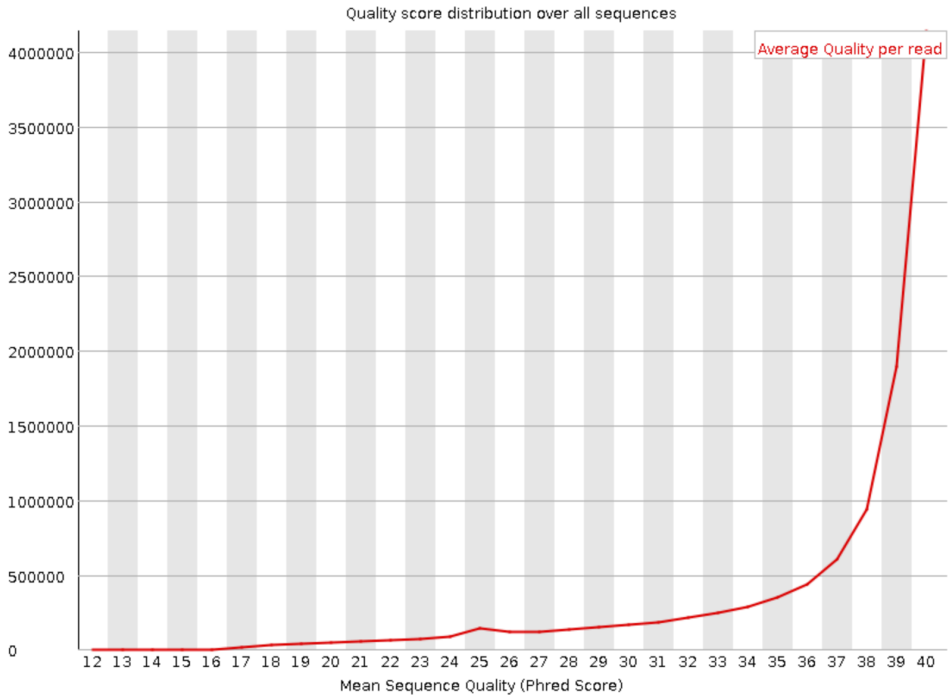

Figure 7. Per sequence quality scores of B

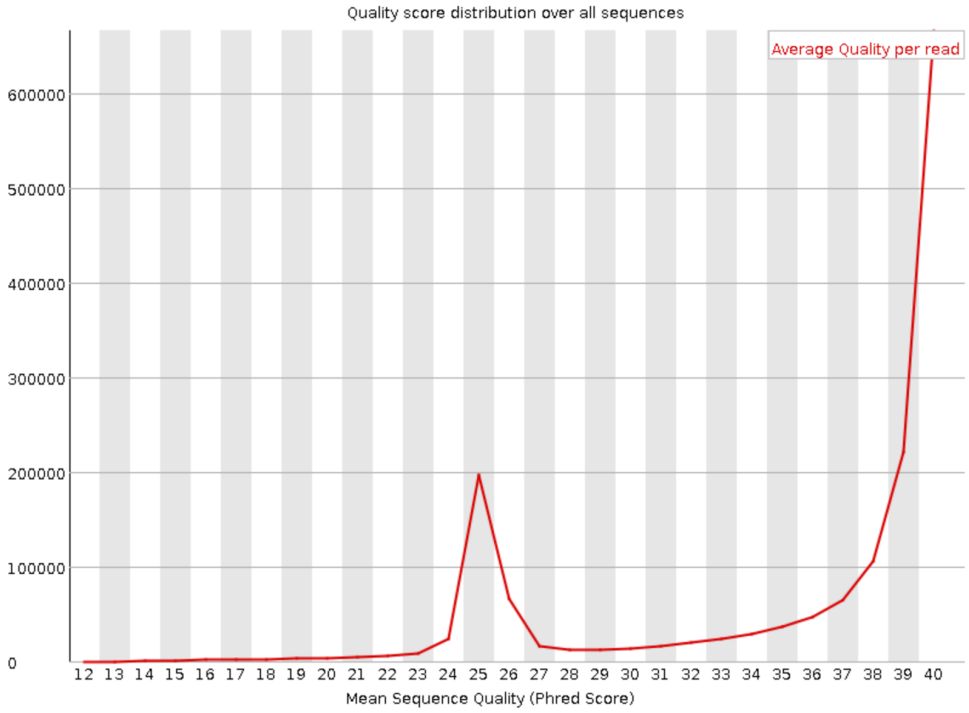

Figure 8. Per sequence quality scores of W

3.1.1.5. Per base sequence content

Provides a graphical representation of per base sequence content of a sample. The ideal results for this test should be lines representing individual bases running in parallel. In a real dataset that can be difficult to achieve. However, the results for sample B are promising since the only more problematic area seems to be between 1-9 bp and the rest of the plot runs more-less in parallel (Fig.9). The results acquired from W display more defects which lasted until 105-109bp, in other words for majority of the plot (Fig.10). It still doesn't indicate that the W dataset would be too poor, since the results are still not critical, but it is more problematic when compared to B results. Both of our samples displayed issues in this category (3x failure and 1x warning). The warning seems to be due to a difference between A-T, G-C bases, which exceeded a threshold of 10% (Andrews, 2010). A failure means that a threshold of 20% was overreached (Andrews, 2010).

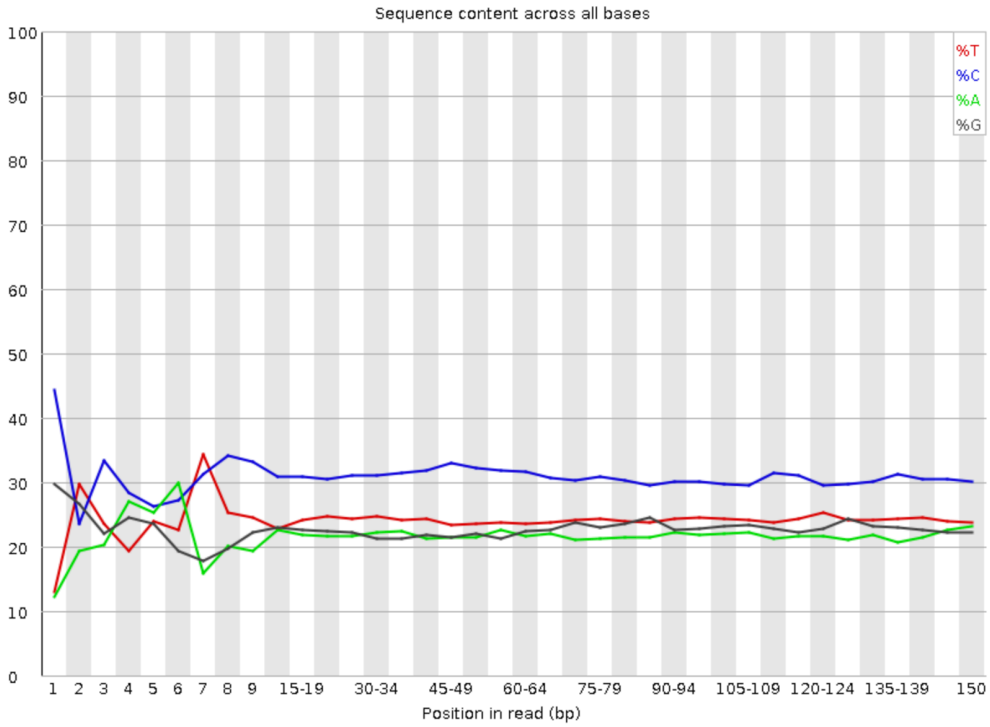

Figure 9. Per base sequence content of B

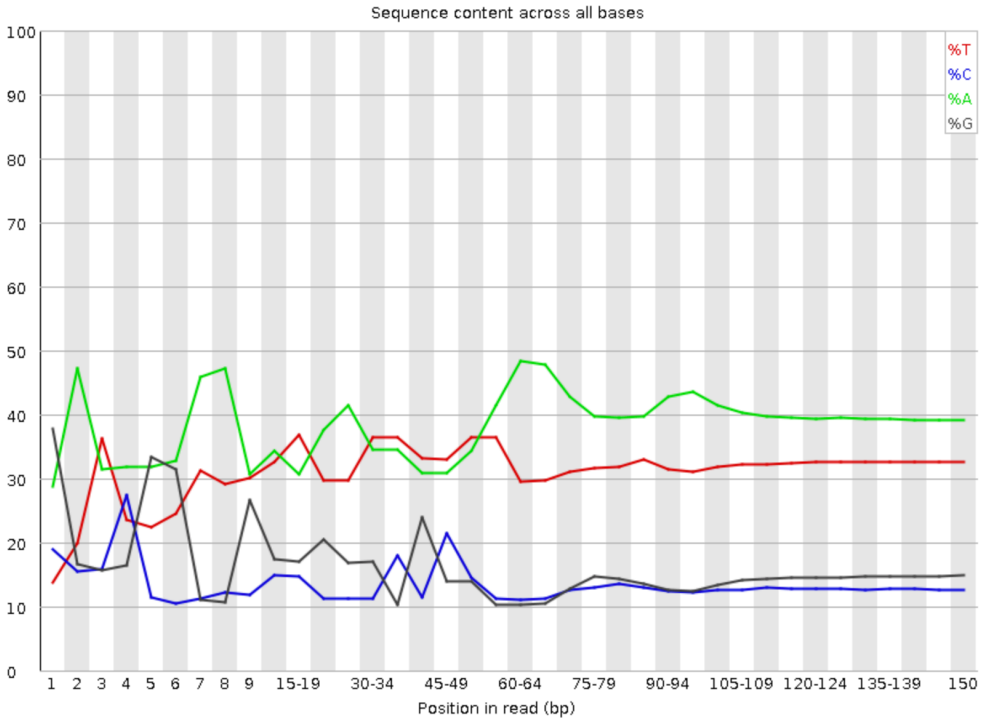

Figure 10. Per base sequence content of W

3.1.1.6. Per sequence GC content

Provides a graphical representation of per sequence GC content of a sample. Ideally the graph of GC content and its peak should match the GC content of the genome. In the graphs the theoretical distribution and GC count per read should correlate as much has possible.

In this category both of the samples are not ideal. Sample B aligns with the theoretical distribution quite well but its peak is too high and that suggest a problematic region (Fig.11). Since the peak shows too high GC values the sequences might have undergone a procedure that increased its GC content. In conclusion the dataset doesn't seem to be of a good quality based on this category. The sample W shows even more problematic values since the two curves are very different and mostly not align(Fig.12). Based on the official FASTQC documentation written by Andrews in 2010 in case a sum of deviations from the normal distribution comprises more than 15% of the reads the program presents a warning. If the distribution crosses a threshold of 30% FASTQC rates this category as a failure (Andrews, 2010).

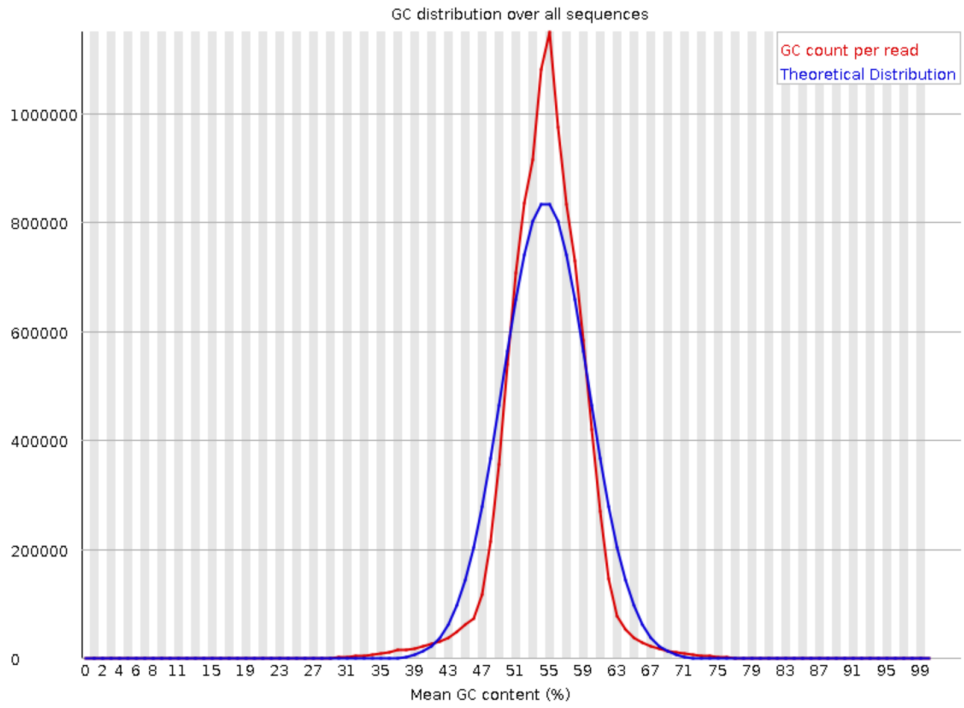

Figure 11. Per sequence GC content of B

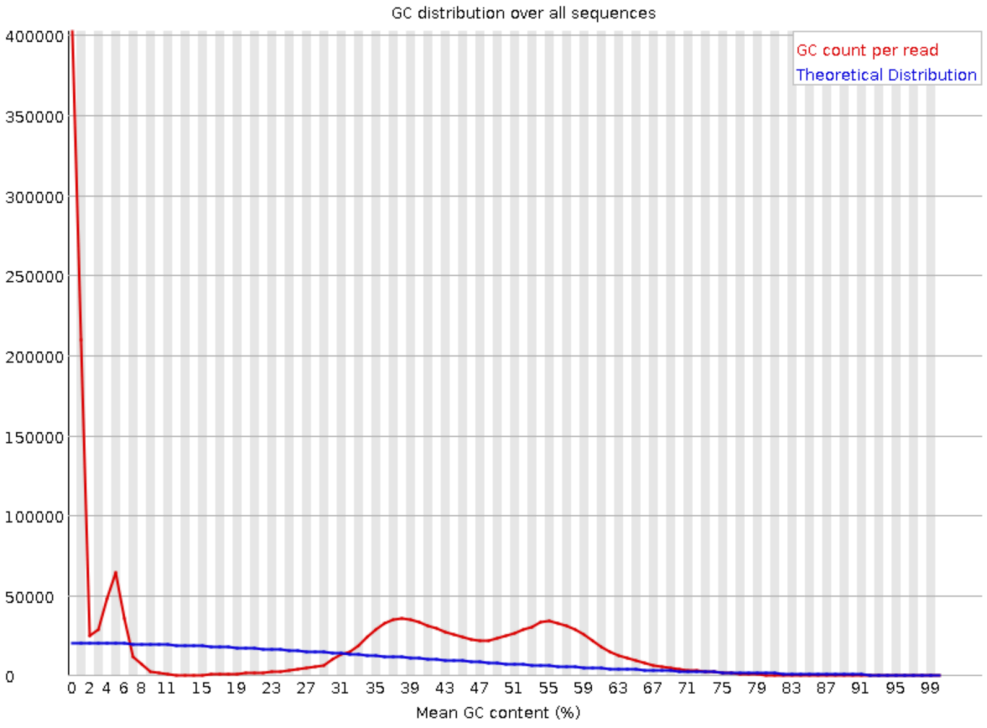

Figure 12. Per sequence GC content of W

3.1.1.7. Per base N content

Provides a graphical representation of per base N content of a sample. In this case both samples B and W have the same results and N content is constantly of the value zero (or very close to zero so the analysis doesn't distinguish it in the graph). This means that the used sequencer didn't have problems with identifying bases in both of our sequences. The graphs are contained in the supplement (Fig. A-1, Fig. A-2).

3.1.1.8. Sequence Length Distribution

Provides a graphical representation of the sequence length distribution of a sample.

Both samples have the same length (as was already mentioned in the first category - Basic Statistics) and their length distribution is also similar. Both samples have sequence length ranging from 149 bp to 152 bp with the majority being 150 bp. The only difference is the number of sequences as B goes up to 1.0E7 and W ends at 1600000. Since these sequences were identified as Illumina encoded the raw reads should all have the same length and therefore this result is to be expected. The graphs are contained in the supplement (Fig. A-3, Fig. A-4)

3.1.1.9. Sequence Duplication Levels

Provides a graphical representation of sequence duplication levels of a sample.

In ideal case the plot for deduplicated sequences and total sequences should be more-less aligned with a slight deviation acceptable, since e.g. high level coverage of sequence can also cause duplication. However, in many cases duplication is caused by unwanted results of certain procedures (e.g. too high level of amplification in sequencing). Both B (Fig.13) and W (Fig.14) showed a failure in this case which means that non-unique sequences make up more than 50% of these samples(Andrews, 2010).

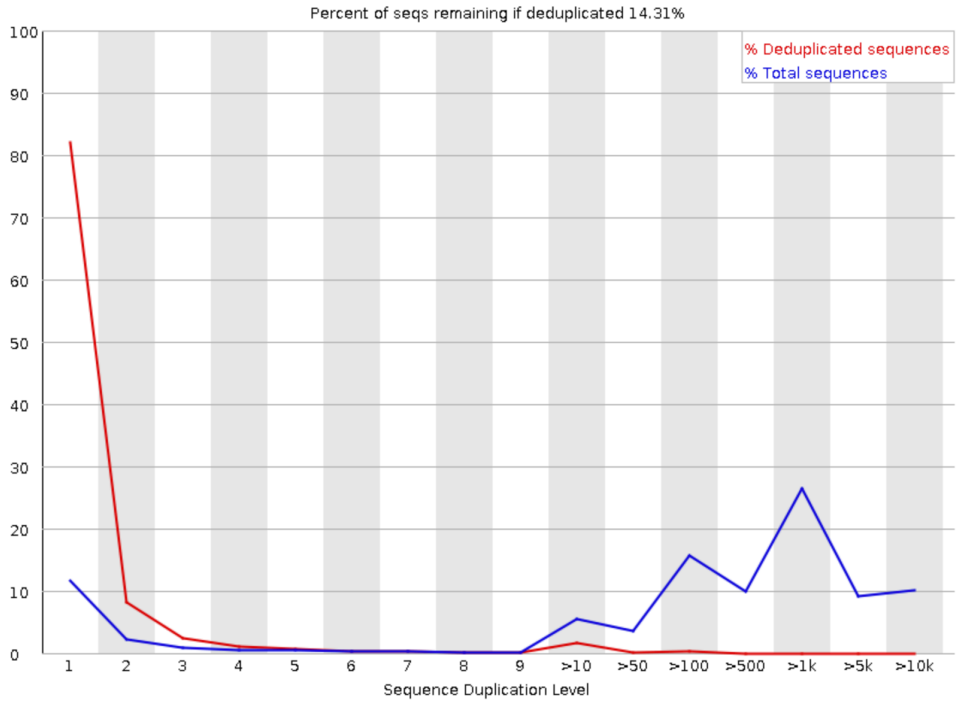

Figure 13. Sequence Duplication Levels of B

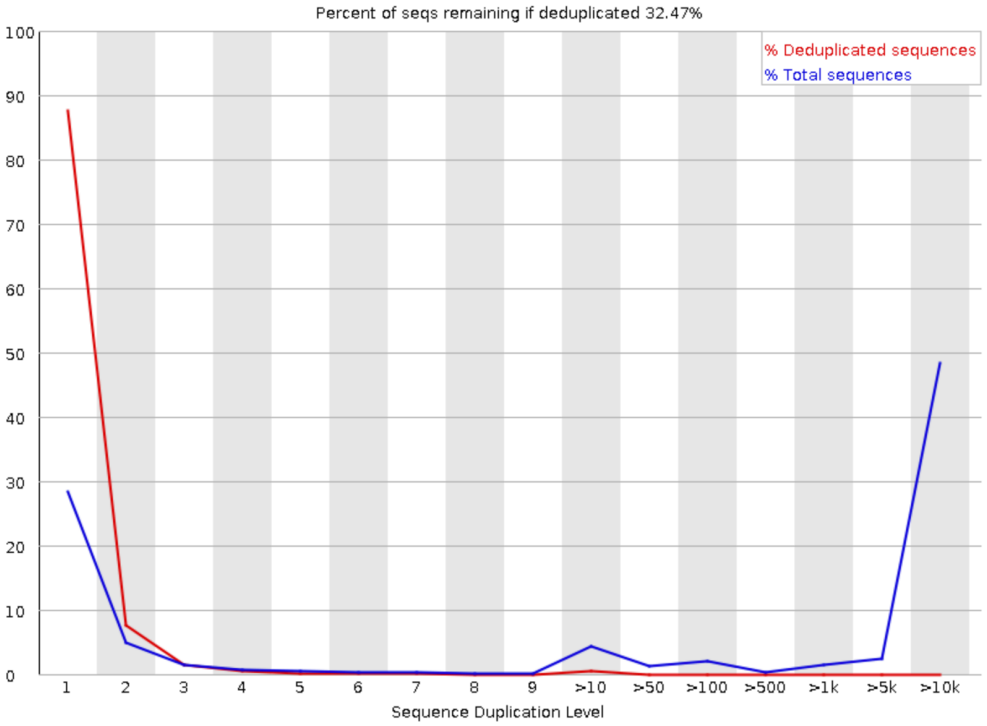

Figure 14. Sequence Duplication Levels of W

3.1.1.10. Overrepresented sequences

Provides a list of overrepresented sequences of a sample. The program lists the sequences together with their count and percentage. It also tries to identify the possible source of these sequences. In our two examples the only identified source was Illumina single end PCR primer, all the other sequences were not identified. B acquired a warning, which is triggered in case any of the present sequences makes up for more than 0.1% of the sample (Andrews, 2010). Failure is cased by more than 1% representation, which was the case of W (Andrews, 2010).

3.1.1.11.Adapter Content

Provides information about adapter content. In graph for sample B (Fig.15) there is a minimal adapter activity percentage, it appears the SOLID small RNA adapter is present but only in very low amount. In sample W the SOLID small RNA adapter is also present only mildly, however Illumina universal adapter shows higher results (Fig. 16). Despite this higher percentage the overall distribution is still relatively low. Only sample W acquired a warning from FASTQC. Andrews states in the official documentation for FASTQC from 2010 that in case a sequence makes up more that 5% of all reads the program will display a warning.

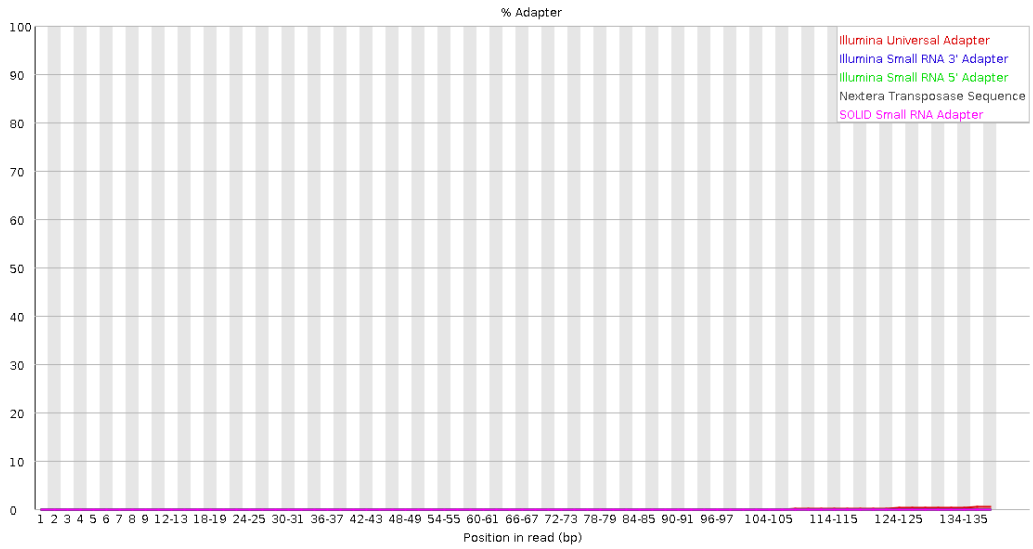

Figure 15. Adapter Content of B

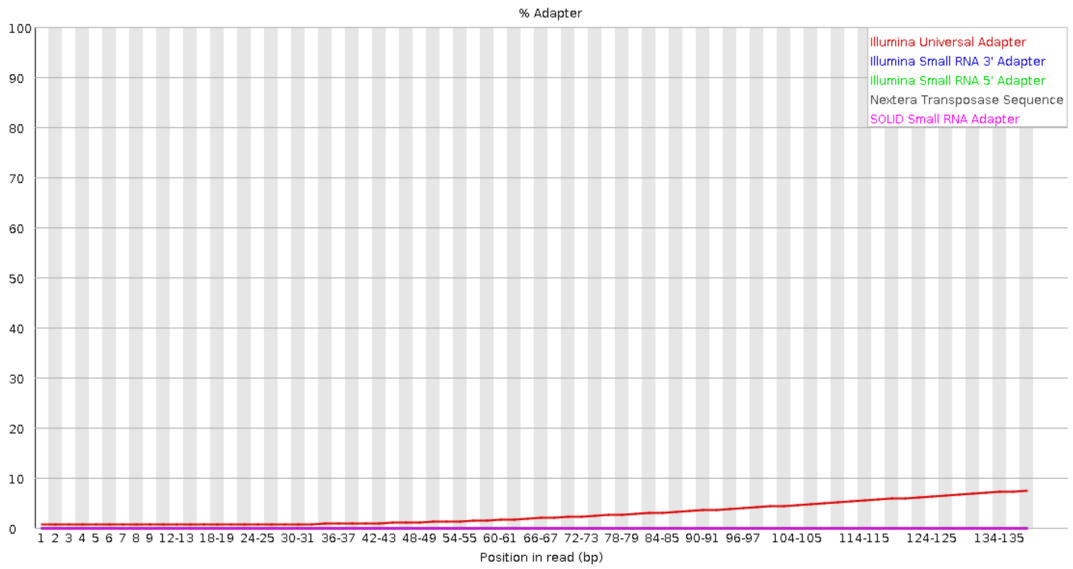

Figure 16. Adapter Content of W

3.1.2. FLASH

In the upper part of the figures (Fig. A-5, Fig. A-6) below the program displays the input sequences that it is working with and also the output files (in this case named by default). This is followed by a list of parameters, which can be altered to acquire more specific results (e.g. by changing minimal and maximal overlap of the sequences). In this project the minimal overlap for analysis of all sequences was set to 10bp and the maximal overlap to 65 bp. In the bottom of the report the program displays read combination statistics, which is where our two samples differ. As visible on a comparison table below (Tab. 1), the sample B has almost 9 million more pairs, over 2 million more combined pairs and almost 7 million more uncombined pairs. The one category where W has higher results is in the percentage of combined pairs, where it scored 27,48% more than B. This is a very interesting outcome, which can have several causes. On of them might be the fact, that samples B and W were chosen based on only FASTQC evaluation. However, it is possible that this assessment was incorrect or too harsh and sample W is actually better than originally predicted. Another possible reason might be that the size difference, which is fairly large, played a role in this case. However, these are merely our speculations.

Table 1. Comparison of FLASH results

	B	W	Difference (ABS)
Total pairs	10625763	1635208	8990555
Combined pairs	3049899	918724	2131175
Uncombined pairs	7575864	716484	6859380
Percent combined	28,70 %	56,18 %	27,48 %

3.1.3. Prinseq

Prinseq allows us to change many parameters for filtering sequences e.g. minimum length or number of bases that will be trimmed at the end of a sequence.

The minimum length was set to 200 bp. The minimum mean quality score was set to 25. The maximum number of ambiguous bases was set to 5 bp. The number of bases that will be trimmed at the end of the sequence was set to 15.

Based on this criteria the Prinseq analysis concluded that our sample B contains 88,24 % of good sequences (Fig.A-7). However, the result for W was drastically different, reaching only 22,80 % (Fig.A-8). As was mentioned before this sample was rated as our worst one, so this low percentage only further proves our previous evaluation. The full prinseq results are contained in the figures below (Fig.A-7, Fig.A-8) together with a comparison table (Tab.2). The above mentioned “good sequences” were saved by Prinseq in a fastq format to the location specified by the user. The program also allows to save the “bad sequences” into a separate file so there is no danger of loosing or mixing samples.

Table 2. Comparison of Prinseq results

	B	W	Difference (ABS)
Input sequences	3049899	918724	2131175
Input bases	750512348	170368034	580144314
Input mean length	246,08	185,44	60,64
Good sequences	2691317	209444	2481873
Good sequences (%)	88,24 %	22,80 %	65,44 %
Good bases	686024150	51707212	634316938
Good mean length	254,90	246,88	8,02
Bad sequences	358582	709280	350698
Bad sequences(%)	11,76 %	77,20 %	65,44 %
Bad bases	64458539	118344667	53886128
Bad mean length	179,76	166,85	12,91

3.1.4.TRIMMOMATIC

This trimming program gave us quite straight forward results, dividing the outcome into two categories - surviving and dropped. In this case both of our samples scored very high i
n surviving sequences and only 0,02% of B(Fig.A-9) and 0,04% of W (Fig.A-10) were dropped. In total comparison (Tab.3) B achieved only 0,02% more of surviving sequences compared to W.

Table 3. Comparison of TRIMMOMATIC results

	B	W	Difference (ABS)
Input reads	3049899	918724	2131175
Surviving	3049296	918373	2130923
Surviving(%)	99,98 %	99,96 %	0,02 %
Dropped	603	351	252
Dropped (%)	0,02 %	0,04 %	0,02 %

3.2.Assembly and its evaluation

All the assemblers were applied on our samples three times. First application took place directly after FLASH, second after Prinseq and last after Trimmomatic. This allowed us to evaluate the work of all these tools at once. To demonstrate this part of the process we again present a comparison of samples B and W. In the complementary tables, we omitted several categories displayed by QUAST, where all our assemblers scored zero (such as # contigs (> = 10 000 bp)).

3.2.1.Statistics after FLASH

Flash results are depicted in the graph below (Fig. 17) and in two enclosed tables (Tab. A-2, Tab. A-3) found in the supplement.

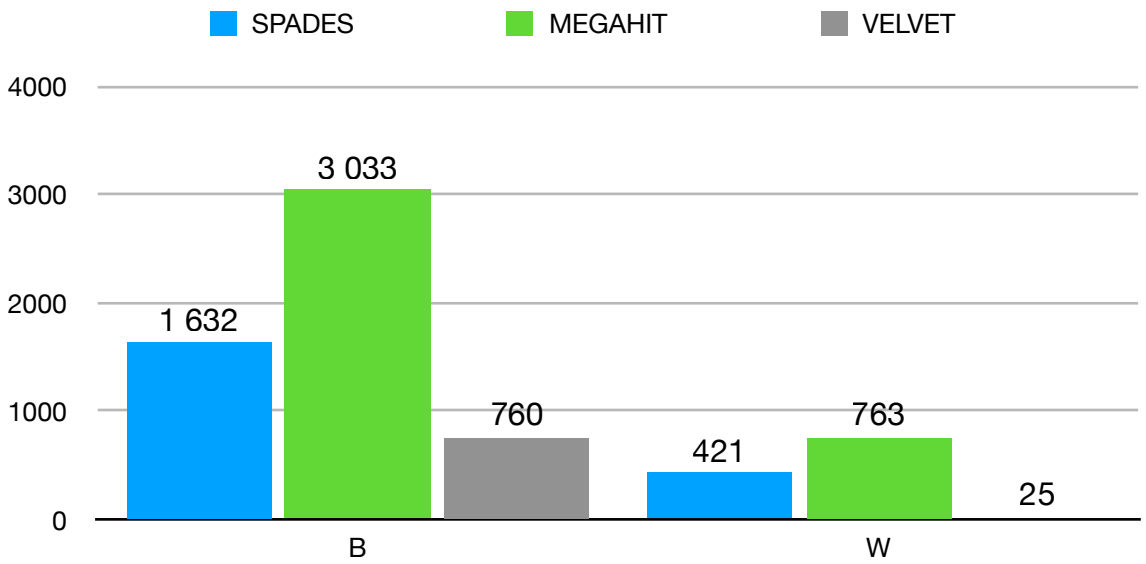

Figure 17. Comparison of numbers of contigs for B and W after FLASH

3.2.2.Statistics after Prinseq

We display the results of Prinseq in the graph contained in Fig. 18 and in two tables (Tab. A-4, Tab. A-5**)** , which can be located in the supplement.

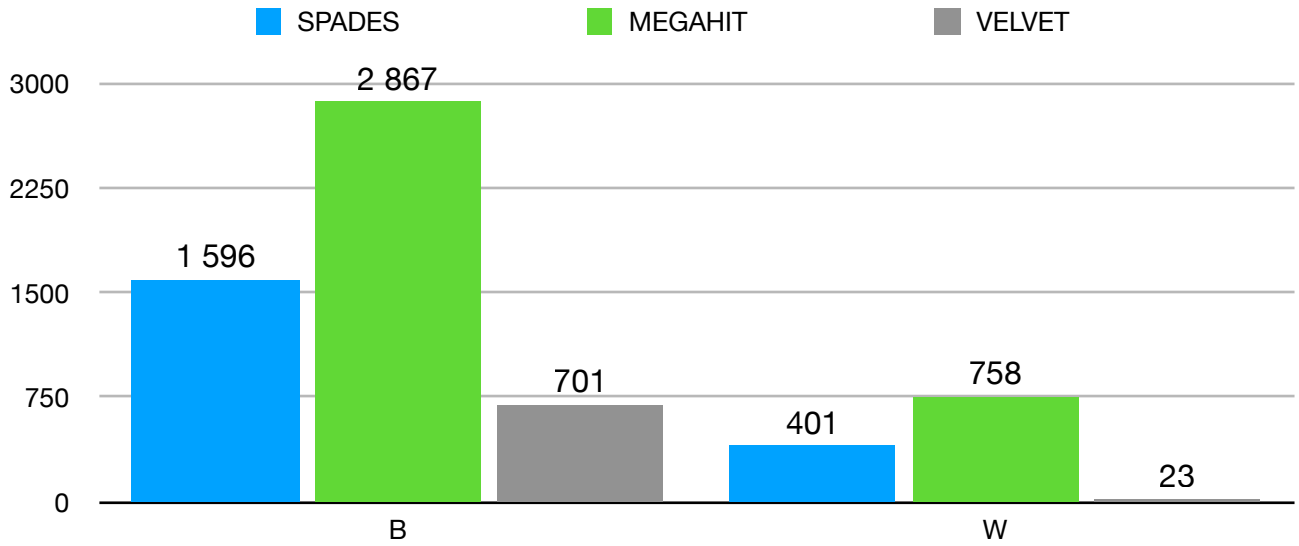

Figure 18. Comparison of numbers of contigs for B and W after PRINSEQ

3.2.3.Statistics after Trimmomatic

Statistics for Trimmomatic are visualised in Fig. 19 and more detailed statistics are in the supplement (Tab. A-6, Tab. A-7)

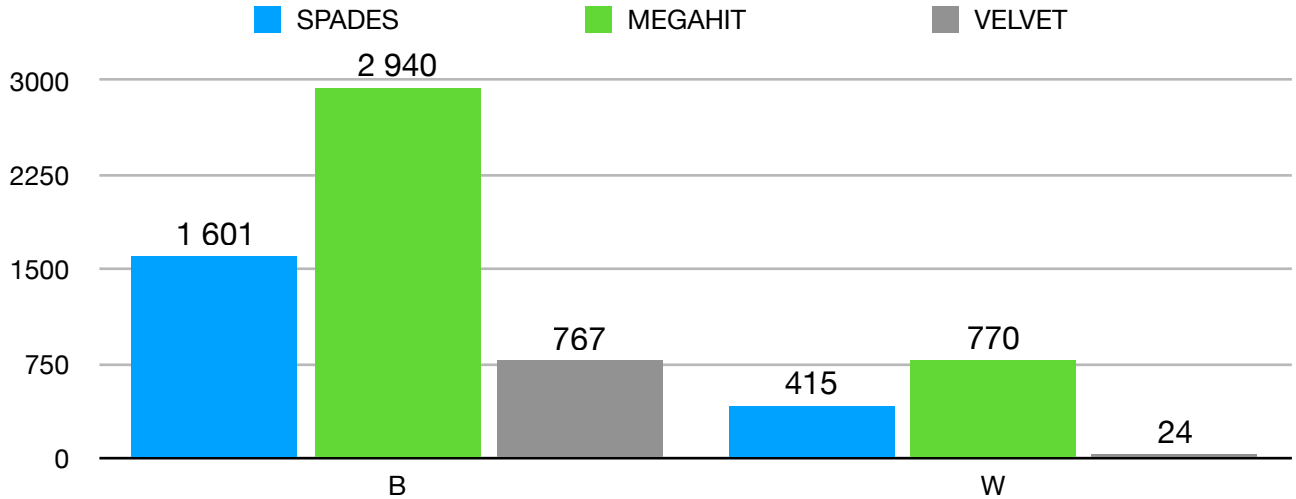

Figure 19. Comparison of numbers of contigs for B and W after TRIMMOMATIC

As it is visible from graphs (Fig.17 , Fig.18 , Fig.19) and enclosed tables the overall distribution of contigs in assemblers is similar for both samples (in other words Megahit works with the most contigs in B and W, Velvet with the least coatings and Spades fits in between the two). This characteristics doesn't change throughout the pipeline. Such results are corresponding to the assembler propositions, as for example Megahit is known to deliver large assemblies with many long contigs (Li *et al.*, 2015).

3.2.4.Overall comparison

3.2.4.1. Number of contigs

As was already mentioned earlier, the distribution of number of contigs among assemblers stays consistent as predicted for both samples and throughout the whole pipeline (Fig. 20, Fig. 21).

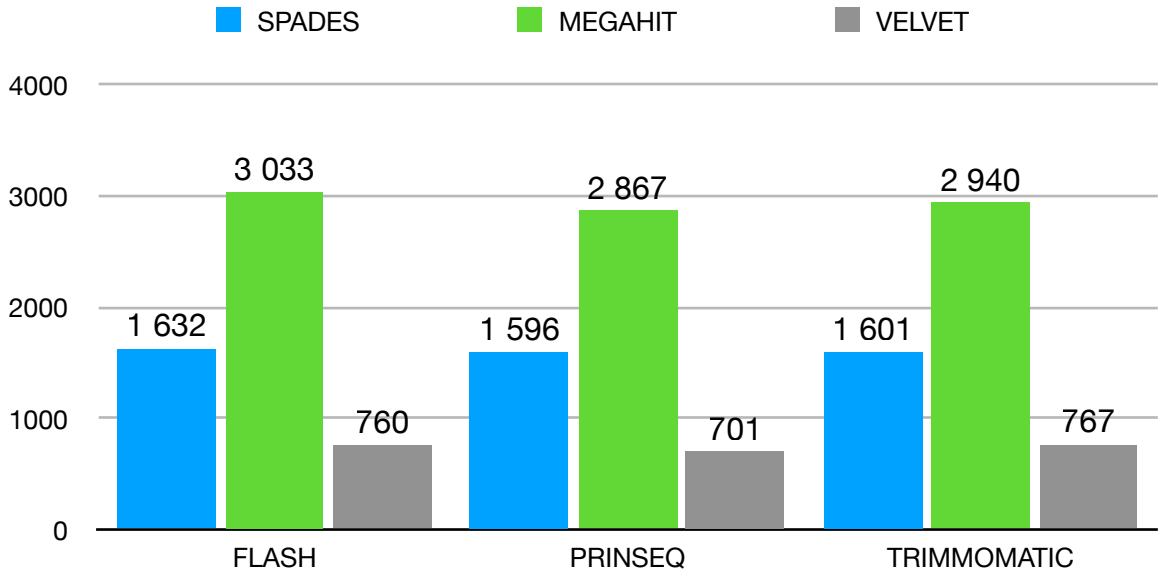

Figure 20. Overall comparison of numbers of contigs for B

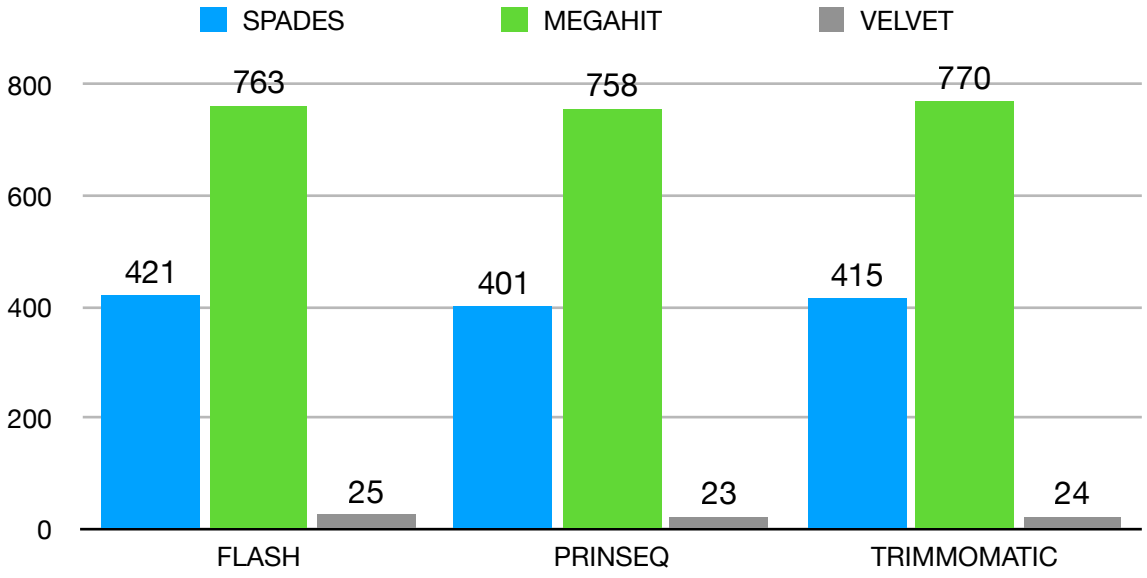

Figure 21. Overall comparison of numbers of contigs for W

3.2.4.2. Longest contig

In case of longest contig evaluation the results are less unanimous. For sample B the outcome of Megahit and Spades is quite similar and the first mentioned doesn't have as clear predominance over the other as in the previous category (Fig.22). The situation is even more interesting in case of W, where Spades possesses the longest contigs after FLASH and Prinseq, however it is again replaced by Megahit after Trimmomatic (Fig. 23). This could be another distinction showing the quality of our samples as B seems to have fairly consistent results throughout the process, but W experiences quite dramatic changes. This development also shows the effect that quality control, filtering and trimming have on the final outcome.

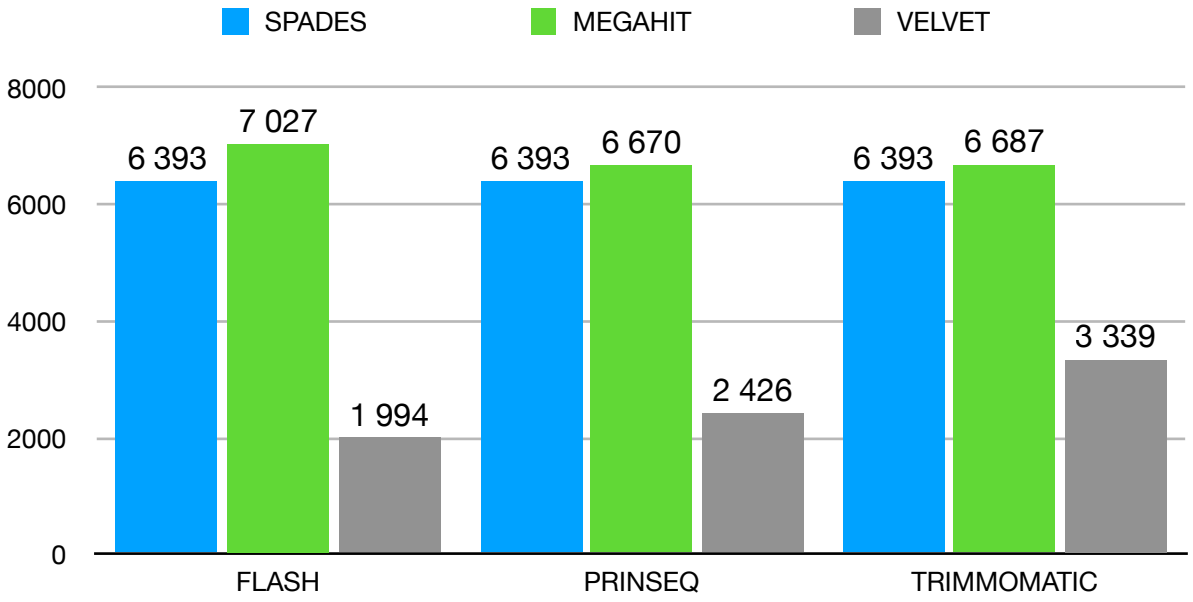

Figure 22. Overall comparison of longest contigs for B

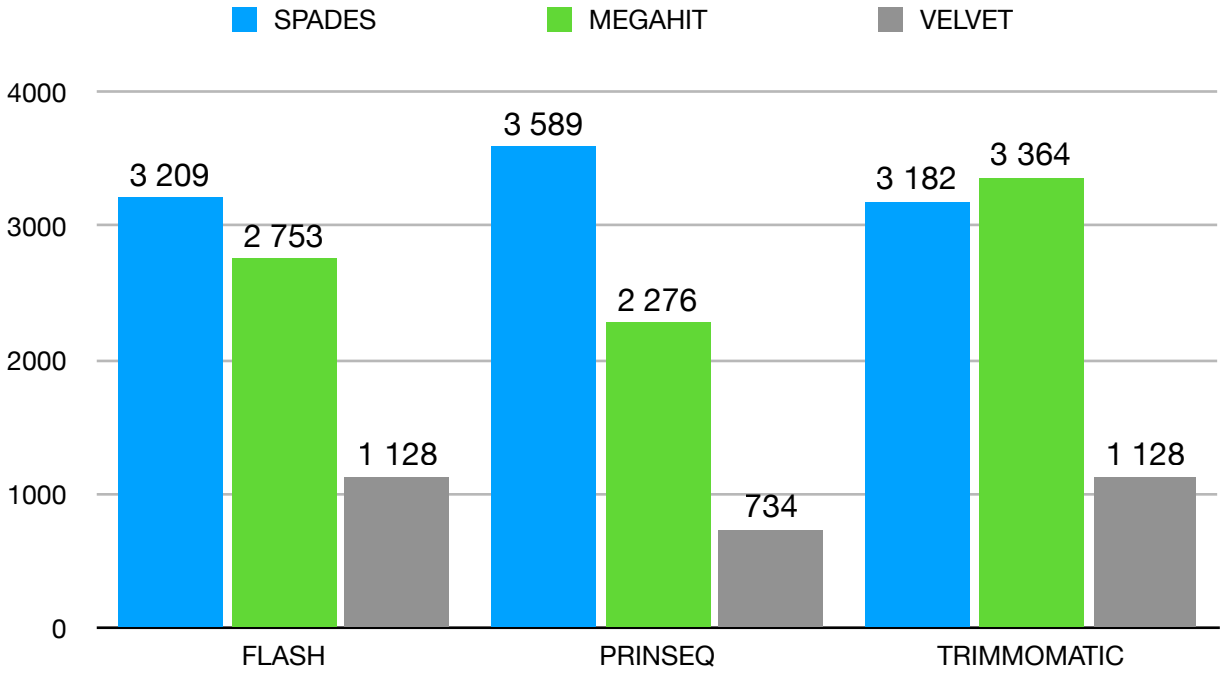

Figure 23. Overall comparison of longest contigs for W

3.2.4.3. Total length

This category also shows an interesting trend, as all the assemblers display a decrease in total length between FLASH and Prinseq, however there is always an increase after Trimmomatic (Fig. 25). There is one peculiarity in case of the Velvet results for B (Fig.24), where the total length after Trimmomatic actually surpasses the one acquired after Flash.

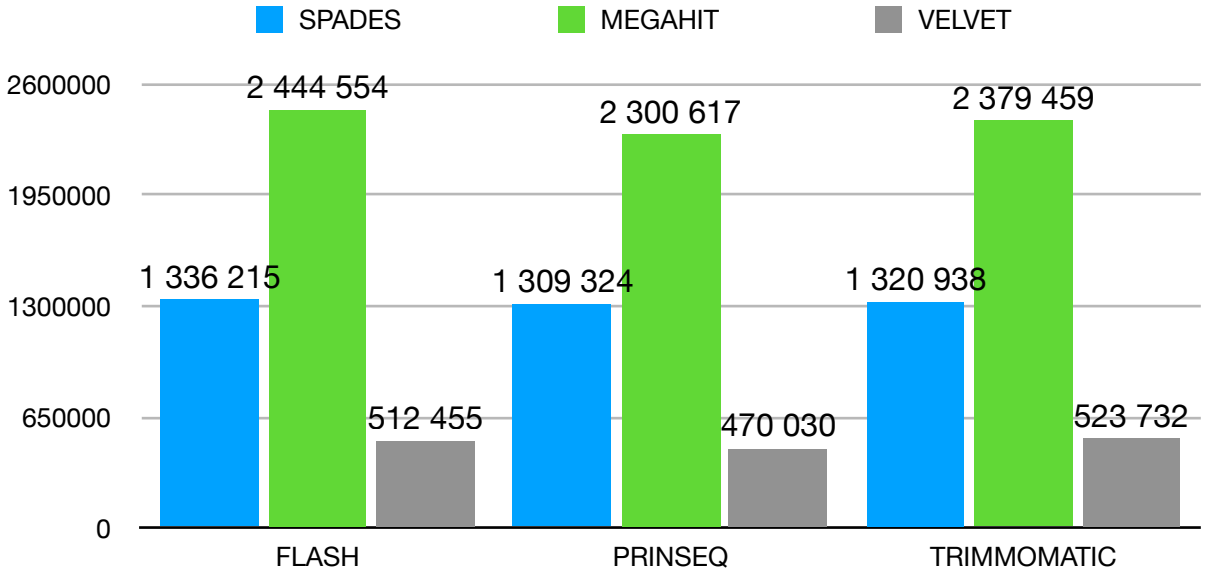

Figure 24. Overall comparison of total length for B

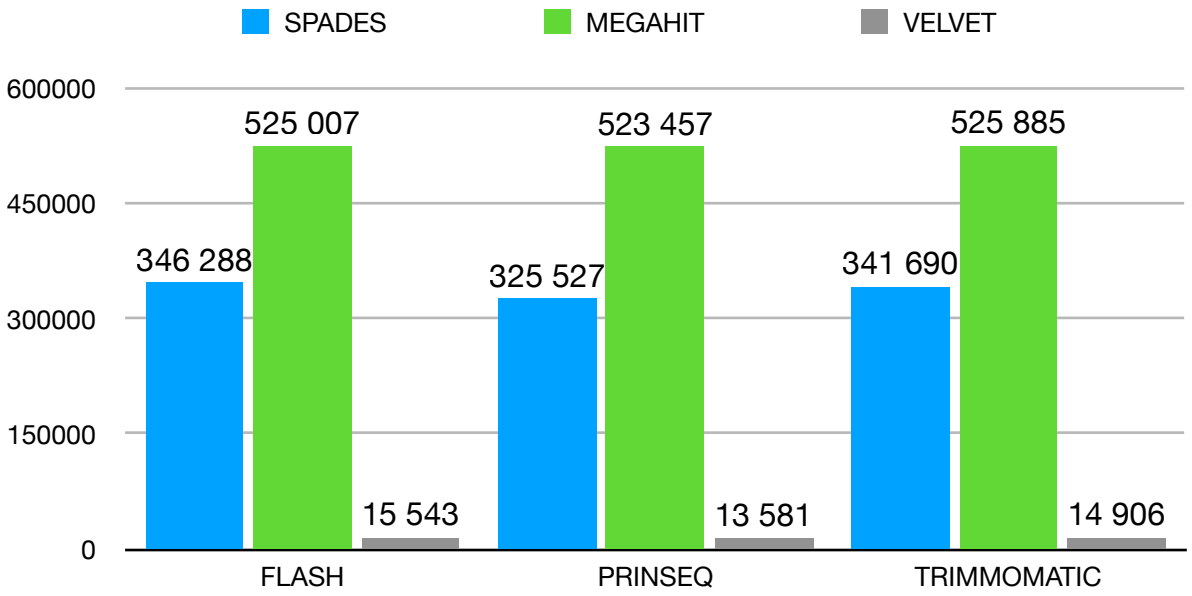

Figure 25. Overall comparison of total length for W

3.3.Alignment and classification

After blasting our samples with DIAMOND, we acquired tabulated, structured files.

Each of these files contained sample codes, accession numbers and most importantly gene identifications and organisms of origin. The last mentioned was crucial for us in the following step. In this part of our work, we applied a python program of our own making, which listed all present organisms and counted their occurrence in the file. The table (Tab.4) below demonstrates the result, which we acquire for set threshold 0.5%.

This table (Tab.4) is obtained together with a plain text file containing extracted organism names. This can be useful as the user can directly paste the text file into e.g. NCBI Taxonomy Browser and acquire taxonomy and a phylogenetic tree.

However, we were also interested in finding out whether the different programs change the overall organism content of the samples. In supplement we enclosed two files ORG_C(0.5)[25]_SUMMARY and ORG_C(0.5)[26]_SUMMARY, which contain all organisms fitting our set threshold 0.5% after all steps of the pipeline as well as a summary. In the last mentioned part, we can clearly see that the total number of organisms decreased from the starting point, however the change is not drastic in this nor the other selected categories. To compare even further we decided to select the top five organisms for each assembler and see

Table 4. Organisms present in VelTrimSW25Full with percentage values 0.5% and higher.

ORGANISM	COUNT	PERCENTAGE
Pseudomonas	32375	5.6119
Pseudomonas fluorescens	24615	4.2668
unclassified Pseudomonas	15029	2.6051
Pseudomonas chlororaphis	8159	1.4143
Collimonas arenae	5473	0.9487
unclassified Duganella	5000	0.8667
Betaproteobacteria bacterium	4925	0.8537
Pseudomonas viridiflava	4857	0.8419
Pseudomonas frederiksbergensis	4833	0.8378
Oxalobacteraceae bacterium	4441	0.7698
Collimonas fungivorans	4340	0.7523
Pseudomonas brassicacearum	4178	0.7242
Acidobacteria bacterium	3320	0.5755
unclassified Massilia	3120	0.5408

how the different programs agree with each other. For both our files the five highest scoring organisms didn't change throughout the process, nevertheless there were differences across the assemblers.

As it is visible from the tables, in case of B the top five organisms from Megahit can be found in the results from Spades and vice versa. Velvet lists majority of these organisms, but not all of them. However, all top five samples from Velvet can be located in results of the two previously mentioned assemblers. As these selected organisms scored very high in case of all assemblers we believe it is a good verification of the actual file contents.

When we look at W the results are less clear. Top five results from Megahit mostly correspond to Velvet (5 - 4 matches), however, Spades starts with 3 matches and after Trimmomatic agrees only with one organism. Surprisingly, all top five results from Spades can be found in Megahit data. Velvet in that case agrees with 3-2 matches. Finally, the top hits for Velvet correspond with Megahit, but Spades found only one match after FLASH analysis and later there are no common matches. Based on this outcome we can see how big impact the sample quality, selected pipeline and assembler can have.

4. Conclusion

Let us begin by stating that the quality control should be considered an important step of any pipeline working with sequences. As our results display, there has been many aspects, which were affected by implementation of filtering programs, however it seems that the biggest role plays the quality of the samples in the very beginning. This might appear as a trivial assessment, nevertheless it is important to stress out that there is only a limited amount of impact, which the quality control programs can have on our samples.

When we look at the results for B there is a certain improvement throughout the pipeline, however the changes are not drastic, when it comes to quality control and they do not seem to significantly change in the context of assembly and alignment either.

Our worst sample W, seems to benefit from quality control more than B, however even these results are not fully conclusive. We dare to say that it's worsened quality at the start had a great impact later on and even our pipeline didn't seem to significantly change this state. When it comes to our assessment of this sample based on alignment and classification the assemblers do not seem to agree the same way as in B, which is another thing that we predicate to the quality of the sample.

In conclusion, for our dataset the quality control pipeline didn't demonstrate a consequential improvement as we originally expected. This makes us believe that the quality control is an important aspect of work with samples like ours, but not necessarily obligatory and could even be omitted in case our dataset has a sufficient quality. However, since the amount of samples we worked with wasn't very large and their quality was still relatively good, we would like to in the future test the acquired assessment on an other potentially more sizeable and diverse set of samples.

5. Bibliography

1. Andrews S (2010) FastQC: A quality control tool for high throughput sequence data. Available: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Retrieved May 13, 2019
2. Magoc, T., & Salzberg, S. L. (2011). FLASH: Fast length adjustment of short reads to improve genome assemblies. Bioinformatics Original Paper, 27(21), 2957-2963. doi:https:// ccb.jhu.edu/ software/FLASH/FLASH-reprint.pdf
3. Schmieder, R., & Edwards, R. (2011). Quality control and preprocessing of metagenomic datasets. Bioinformatics (Oxford, England), 27(6), 863–864. doi:10.1093/bioinformatics/btr026
4. Zerbino, D. R., & Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research, 18(5), 821–829. doi:10.1101/gr.074492.107
5. Miller, J. R., Koren, S., Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, *95*(6), 315–327. https://doi.org/10.1016/j.ygeno.2010.03.001
6. Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, Tak-Wah Lam, MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct *de Bruijn* graph, *Bioinformatics*, Volume 31, Issue 10, 15 May 2015, Pages 1674–1676, https:// doi.org/10.1093/bioinformatics/btv033
7. Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, Jingbo Tang, Gengxiong Wu, Hao Zhang, Yujian Shi, Yong Liu, Chang Yu, Bo Wang, Yao Lu, Changlei Han, David W. Cheung, Siu-Ming Yiu, Shaoliang Peng, Zhu Xiaoqian, Guangming Liu, Xiangke Liao, Yingrui Li, Huanming Yang, Jian Wang, Tak-Wah Lam, Jun Wang, SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler, GigaScience, Volume 4, Issue 1, December 2015, s13742–015–0069–2, https://doi.org/10.1186/s13742-015-0069-2
8. Arbor, A, Gene Codes Corporation (2008), Sequencher - Tutorial for Windows and Macintosh Trimming Sequence [PDF file] (n.d.). Retrieved May 13, 2019, doi: http:// www.genecodes.com/sites/default/files/documents/ Tutorials/Trimming%20Sequence.pdf
9. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q, Chen Z, Mauceli E, Hacohen N, Gnirke A, Rhind N, di Palma F, Birren BW, Nusbaum C, Lindblad-Toh K, Friedman N, Regev A. Full-length transcriptome assembly from RNA-seq data without a reference genome. Nat Biotechnol. 2011 May 15;29(7):644-52. doi: 10.1038/nbt.1883. PubMed PMID: 21572440.

10. Yoon, S., Kim, D., Kang, K. *et al.* TraRECo: a greedy approach based de novo transcriptome assembler with read error correction using consensus matrix. *BMC Genomics* **19,** 653 (2018). https://doi.org/10.1186/s12864-018-5034-x
11. Pop, M., & Salzberg, S. L. (2008). Bioinformatics challenges of new sequencing technology. *Trends in Genetics*, *24*(3), 142–149. doi: 10.1016/j.tig.2007.12.006
12. Masoudi-Nejad, A., Narimani, Z., & Hosseinkhan, N. (2013). Next generation sequencing and sequence assembly: methodologies and algorithms. New York: Springer.
13. Warren, R. L., Sutton, G. G., Jones, S. J. M., & Holt, R. A. (2006). Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, *23*(4), 500–501. doi: 10.1093/bioinformatics/btl629
14. Dohm, J. C., Lottaz, C., Borodina, T., & Himmelbauer, H. (2007). SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, *17*(11), 1697–1706. doi: 10.1101/gr.6435207
15. Jeck W.R., Reinhardt J.A., Baltrus D.A., Hickenbotham M.T., Magrini V., Mardis E.R., Dangl J.L., Jones C.D., Extending assembly of short DNA sequences to handle error, Bioinformatics, Volume 23, Issue 21, 1 November 2007, Pages 2942–2944, https://doi.org/10.1093/bioinformatics/btm451
16. Haider, B., Ahn, T.-H., Bushnell, B., Chai, J., Copeland, A., & Pan, C. (2014). Omega: an Overlap-graph de novo Assembler for Metagenomics. *Bioinformatics*, *30*(19), 2717–2722. doi: 10.1093/bioinformatics/btu395
17. Denisov, G., Walenz, B., Halpern, A. L., Miller, J., Axelrod, N., Levy, S., & Sutton, G. (2008). Consensus generation and variant detection by Celera Assembler. *Bioinformatics*, *24*(8), 1035–1040. doi: 10.1093/bioinformatics/btn074
18. Adams, J. (2008) Complex genomes: Shotgun sequencing. Nature Education 1(1):186
19. Elloumi, M., & Zomaya, A. Y. (2014). Biological knowledge discovery handbook: preprocessing, mining, and postprocessing of biological data. Hoboken, NJ: John Wiley & Sons, Inc.
20. Gurevich, A., Saveliev, V., Vyahhi, N., & Tesler, G. (2013). QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, *29*(8), 1072–1075. doi: 10.1093/bioinformatics/btt086
21. Buchfink B, Xie C, Huson DH, "Fast and sensitive protein alignment using DIAMOND", *Nature Methods* **12**, 59-60 (2015). doi:10.1038/nmeth.3176

22. Wheeler,D.L., Benson,D.A., Bryant,S., Canese,K., Church,D.M., Edgar,R., Federhen,S., Helmberg,W., Kenton,D., Khovayko,O. *et al* . (2005) Database resources of the National Center for Biotechnology Information: Update. *Nucleic Acid Res* , 33 , D39 –D45.
23. Schuler,G.D., Epstein,J.A., Ohkawa,H. and Kans,J.A. (1996) Entrez: molecular biology database and retrieval system. *Methods Enzymol.* , 266 , 141 –162.
24. Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, *30*(15), 2114–2120. doi: 10.1093/bioinformatics/btu170
25. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., … Pevzner, P. A. (2012). SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology*, *19*(5), 455–477. doi: 10.1089/cmb.2012.0021
26. Nurk S. *et al.* (2013) Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads. In: Deng M., Jiang R., Sun F., Zhang X. (eds) Research in Computational Molecular Biology. RECOMB. Lecture Notes in Computer Science, vol 7821. Springer, Berlin, Heidelberg
27. Li, D., Liu, C.-M., Luo, R., Sadakane, K., & Lam, T.-W. (2015). MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, *31*(10), 1674–1676. doi: 10.1093/bioinformatics/btv033
28. Mikheenko, A., Prjibelski, A., Saveliev, V., Antipov, D., & Gurevich, A. (2018). Versatile genome assembly evaluation with QUAST-LG. *Bioinformatics*, *34*(13), i142–i150. doi: 10.1093/bioinformatics/bty266
29. Trim reads for several criteria(n.d.). Retrieved May 13, 2019, doi: https://chipster.csc.fi/manual/prinseq-trimmer.html
30. Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

31. Supplement Content

A. Additional figures and tables for samples B,W

B. Script for additional work with blasted files

C. Supplement file containing FASTQC reports, summary of FLASH and PRINSEQ results, ORG_DB_STATS.py as well as example files resulting from this program for threshold of 0.5% together with overall summary of these outcomes in PDF format for files B and W.

A. Additional figures and tables

Table A-1. FASTQC Report for problematic categories in case of B and W

Categories	Warning	Failure
Per tile sequence quality	• B_R2	
Per base sequence content	• B_R2	• B_R1 • W_R1 • W_R2
Per sequence GC content	• B_R1 • B_R2	• W_R1 • W_R2
Sequence Duplication Levels		• B_R1 • B_R2 • W_R1 • W_R2
Overrepresented sequences	• B_R1 • B_R2	• W_R1 • W_R2
Adapter Content	• W_R1 • W_R2	

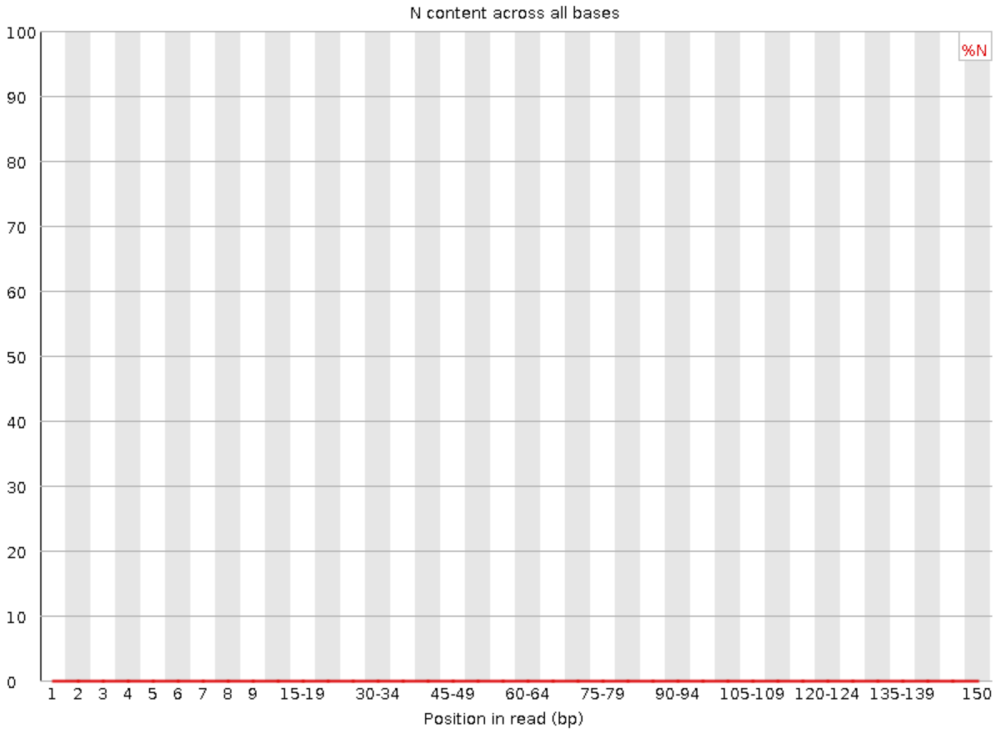

Figure A-1. Per base N content of B

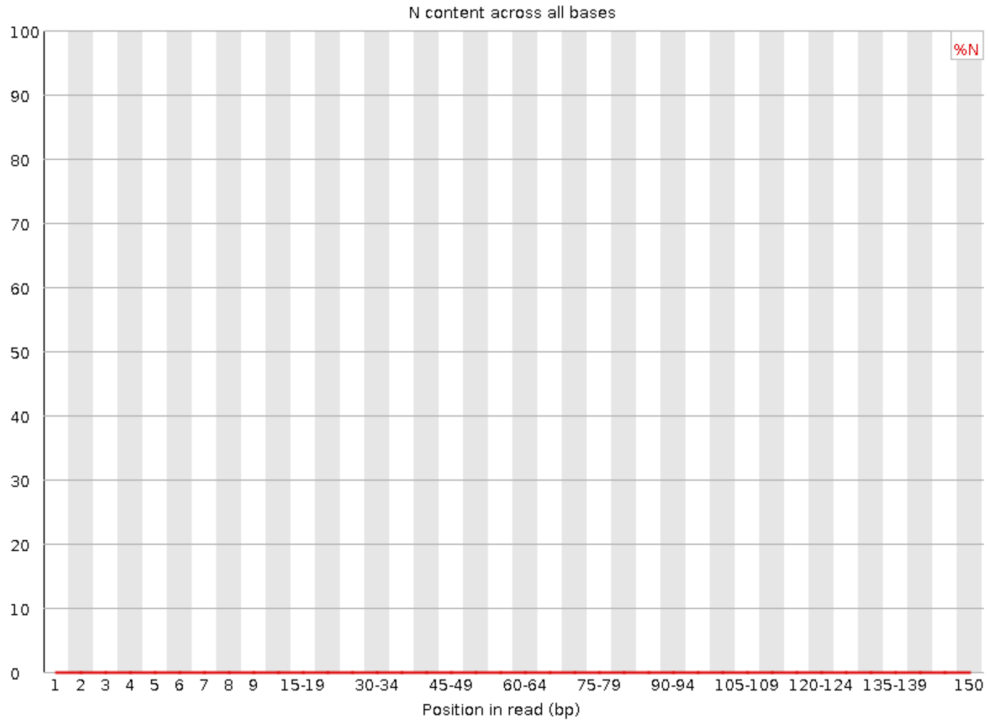

Figure A-2. Per base N content of W

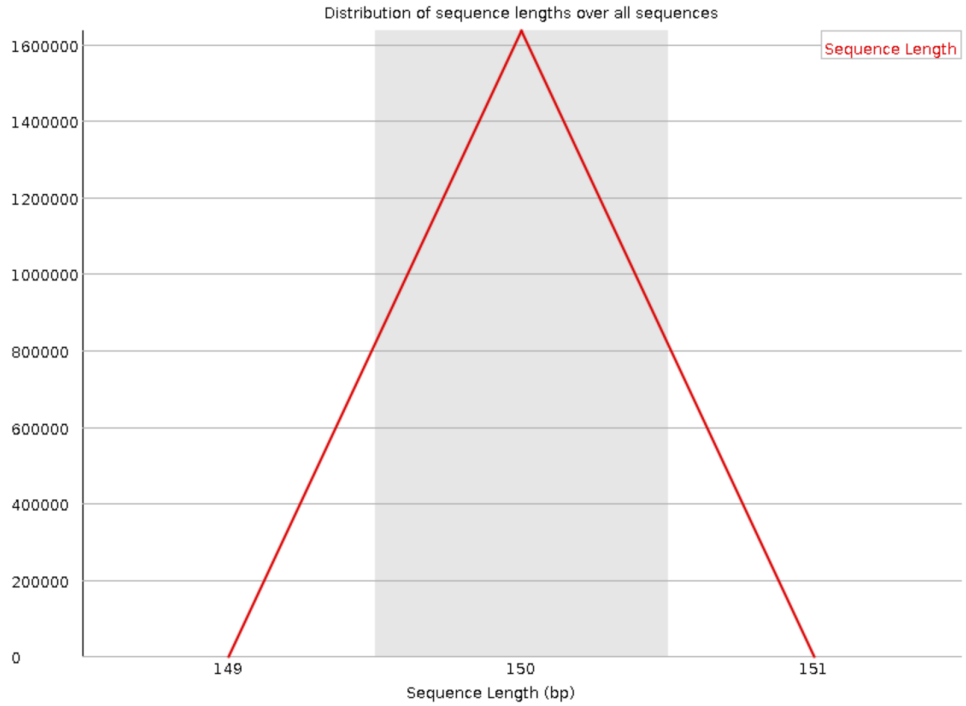

Figure A-3. Sequence Length Distribution of W

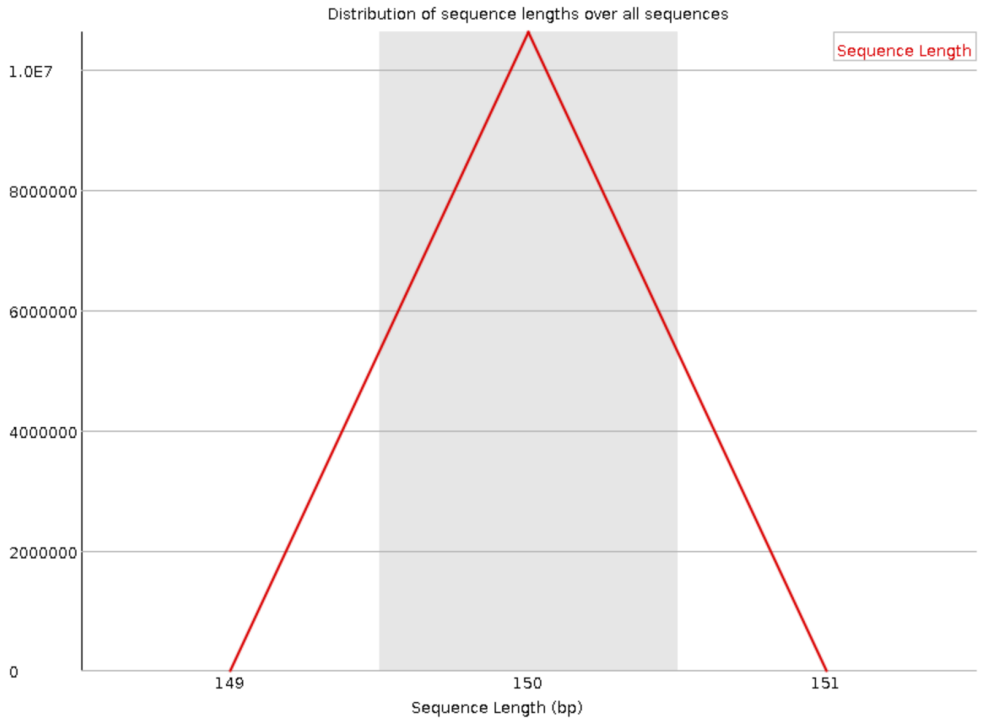

Figure A-4. Sequence Length Distribution of B

```
[FLASH] Input files:
[FLASH]     16NQ040_B_ACTGAT_L004_R1_001.fastq
[FLASH]     16NQ040_B_ACTGAT_L004_R2_001.fastq
[FLASH]
[FLASH] Output files:
[FLASH]     ./out.extendedFrags.fastq
[FLASH]     ./out.notCombined_1.fastq
[FLASH]     ./out.notCombined_2.fastq
[FLASH]     ./out.hist
[FLASH]     ./out.histogram
[FLASH]
[FLASH] Parameters:
[FLASH]     Min overlap:          10
[FLASH]     Max overlap:          65
[FLASH]     Max mismatch density:  0.250000
[FLASH]     Allow "outie" pairs:   false
[FLASH]     Cap mismatch quals:    false
[FLASH]     Combiner threads:      16
[FLASH]     Input format:         FASTQ, phred_offset=33
[FLASH]     Output format:        FASTQ, phred_offset=33

[FLASH] Read combination statistics:
[FLASH]     Total pairs:      10625763
[FLASH]     Combined pairs:   3049899
[FLASH]     Uncombined pairs: 7575864
[FLASH]     Percent combined: 28.70%
```

Figure A-5. FLASH results for B

```
[FLASH] Input files:
[FLASH]     16NQ040_W_ATGAGC_L004_R1_001.fastq
[FLASH]     16NQ040_W_ATGAGC_L004_R2_001.fastq
[FLASH]
[FLASH] Output files:
[FLASH]     ./out.extendedFrags.fastq
[FLASH]     ./out.notCombined_1.fastq
[FLASH]     ./out.notCombined_2.fastq
[FLASH]     ./out.hist
[FLASH]     ./out.histogram
[FLASH]
[FLASH] Parameters:
[FLASH]     Min overlap:          10
[FLASH]     Max overlap:          65
[FLASH]     Max mismatch density:  0.250000
[FLASH]     Allow "outie" pairs:   false
[FLASH]     Cap mismatch quals:    false
[FLASH]     Combiner threads:      16
[FLASH]     Input format:         FASTQ, phred_offset=33
[FLASH]     Output format:        FASTQ, phred_offset=33

[FLASH] Read combination statistics:
[FLASH]     Total pairs:      1635208
[FLASH]     Combined pairs:   918724
[FLASH]     Uncombined pairs: 716484
```

Figure A-6. FLASH results for W

16NQ040_B_ACTGAT_L004_R1-R2

Input and filter stats:
Input sequences: 3,049,899
Input bases: 750,512,348
Input mean length: 246.08
Good sequences: 2,691,317 (88.24%)
Good bases: 686,024,150
Good mean length: 254.90
Bad sequences: 358,582 (11.76%)
Bad bases: 64,458,539
Bad mean length: 179.76
Sequences filtered by specified parameters:
trim_tail_left: 405
min_len: 355053
min_qual_mean: 2668
min_qual_mean: 1643

Figure A-7. Prinseq analysis for B

16NQ040_W_ATGAGC_L004_R1-R2

Input and filter stats:
Input sequences: 918,724
Input bases: 170,368,034
Input mean length: 185.44
Good sequences: 209,444 (22.80%)
Good bases: 51,707,212
Good mean length: 246.88
Bad sequences: 709,280 (77.20%)
Bad bases: 118,344,667
Bad mean length: 166.85
Sequences filtered by specified parameters:
trim_tail_left: 133709
trim_tail_right: 3
min_len: 573883
min_qual_mean: 1643

Figure A-8. Prinseq analysis for W

```
TrimmomaticSE:
Started with arguments:
FLASh/16NQ040_B_ACTGAT_L004_R1-R2/out.extendedFrags.fastq trimresults/
TrimSW25 SLIDINGWINDOW:50:25
Automatically using 1 threads
Quality encoding detected as phred33
Input Reads: 3049899
Surviving: 3049296 (99.98%)
Dropped: 603 (0.02%)
```

Figure A-9. TRIMMOMATIC analysis for B

```
TrimmomaticSE:
Started with arguments:
 FLASh/16NQ040_W_ATGAGC_L004_R1-R2/out.extendedFrags.fastq trimresults/
TrimSW26 SLIDINGWINDOW:50:25
Automatically using 1 threads
Quality encoding detected as phred33
Input Reads: 918724
Surviving: 918373 (99.96%)
Dropped: 351 (0.04%)
```

Figure A-10. TRIMMOMATIC analysis for W

Table A-2. Quast results for B after FLASH

B after FLASH			
	SPAdes	Megahit	Velvet
# contigs	1632	3033	760
# contigs (> = 0 bp)	2318	7688	104 798
# contigs (> = 1000 bp)	333	531	52
# contigs (> = 5000 bp)	2	4	0
Largest contig	6393	7027	1994
Total length	1 336 215	2 444 554	512 455
Total length (> = 0 bp)	1 648 987	4 226 610	12 240 023
Total length (> = 1000 bp)	497 656	823 034	63 602
Total length (> = 5000 bp)	12 081	25 627	0
N50	787	765	650
N75	605	598	561
L50	528	996	305
L75	1019	1911	519
GC%	54,82	56,71	57,75

Table A-3. Quast results for W after FLASH

W after FLASH			
	SPAdes	Megahit	Velvet
# contigs	421	763	25
# contigs (> = 0 bp)	623	2703	72 661
# contigs (> = 1000 bp)	98	58	1
# contigs (> = 5000 bp)	0	0	0
Largest contig	3209	2753	1128
Total length	346 288	525 007	15 543
Total length (> = 0 bp)	437 685	1 269 348	8 802 303
Total length (> = 1000 bp)	133 733	74 247	1128
Total length (> = 5000 bp)	0	0	0
N50	852	657	578
N75	612	564	536
L50	142	299	11
L75	264	516	18
GC%	51,91	49,36	55,5

Table A-4. Quast results for B after PRINSEQ

B after PRINSEQ			
	SPAdes	Megahit	Velvet
# contigs	1596	2867	701
# contigs (> = 0 bp)	2253	7224	97 751
# contigs (> = 1000 bp)	316	469	48
# contigs (> = 5000 bp)	2	3	0
Largest contig	6393	6670	2426
Total length	1 309 324	2 300 617	470 030
Total length (> = 0 bp)	1 609 581	3 975 216	11 383 574
Total length (> = 1000 bp)	475 998	739 789	57 796
Total length (> = 5000 bp)	12 032	18 985	0
N50	801	755	647
N75	607	599	560
L50	518	950	282
L75	996	1813	479
GC%	54,8	56,62	57,62

Table A-5. Quast results for W after PRINSEQ

W after PRINSEQ			
	SPAdes	Megahit	Velvet
# contigs	401	758	23
# contigs (> = 0 bp)	602	2611	56 892
# contigs (> = 1000 bp)	75	64	0
# contigs (> = 5000 bp)	0	0	0
Largest contig	3589	2276	734
Total length	325 527	523 457	13 581
Total length (> = 0 bp)	417 106	1 240 960	7 117 707
Total length (> = 1000 bp)	110 150	84 935	0
Total length (> = 5000 bp)	0	0	0
N50	817	661	587
N75	608	564	540
L50	134	295	11
L75	253	511	17
GC%	51,96	49,49	53,46

Table A-6. Quast results for B after TRIMMOMATIC

B after TRIMMOMATIC			
	SPAdes	Megahit	Velvet
# contigs	1601	2940	767
# contigs (> = 0 bp)	2264	7516	80 676
# contigs (> = 1000 bp)	332	518	47
# contigs (> = 5000 bp)	2	4	0
Largest contig	6393	6687	3339
Total length	1 320 938	2 379 459	523 732
Total length (> = 0 bp)	1 623 419	4 140 136	10 203 337
Total length (> = 1000 bp)	499 043	804 044	61 273
Total length (> = 5000 bp)	12 080	24 633	0
N50	795	770	654
N75	607	600	568
L50	515	964	306
L75	996	1849	521
GC%	54,8	56,83	57,66

Table A-7. Quast results for W after TRIMMOMATIC

W after TRIMMOMATIC			
	SPAdes	Megahit	Velvet
# contigs	415	770	24
# contigs (> = 0 bp)	622	2765	57 145
# contigs (> = 1000 bp)	92	56	1
# contigs (> = 5000 bp)	0	0	0
Largest contig	3182	3364	1128
Total length	341 690	525 885	14 906
Total length (> = 0 bp)	435 553	1 293 165	7 277 513
Total length (> = 1000 bp)	129 131	75 290	1128
Total length (> = 5000 bp)	0	0	0
N50	835	650	576
N75	608	561	540
L50	138	303	11
L75	260	522	17
GC%	52,18	49,6	55,37

B. Script for additional work with blasted files

File name: ORG_DB_STATS.py
Language: Python
Description: The script evaluates the input file and presents the user with either .csv file with organism names, their occurrences and the corresponding overall percentage or a .csv file similar to the previously mentioned one, but limited by a threshold selected by the user as well as a plain text file containing organism names.
Input file: Tabulated output of Diamond blast
Output file: The results are presented in a .csv file and plain text file.

The full script is enclosed in the supplement file.