Assembly and annotation of genomes

Dr. rer. nat. Diego Mauricio Riaño-Pachón

Laboratório de Biologia Computacional, Evolutiva e de Sistemas

Centro de Energia Nuclear na Agricultura

Universidade de São Paulo

diego.riano@cena.usp.br

Required software and download data

Make sure there are not error messages after running each of the following commands!

Run the following command to install the required software for this tutorial:

```
sudo apt install khmer fastqc sra-toolkit trimmomatic velvet*
python-pip libegl1-mesa

wget https://repo.anaconda.com/miniconda/Miniconda2-latest-
Linux-x86_64.sh
bash Miniconda2-latest-Linux-x86_64.sh -b -p ~/miniconda

echo "export PATH=~/miniconda/bin:$PATH" >> ~/.bashrc
source ~/.bashrc
conda install -y -c bioconda kmergenie quast
```

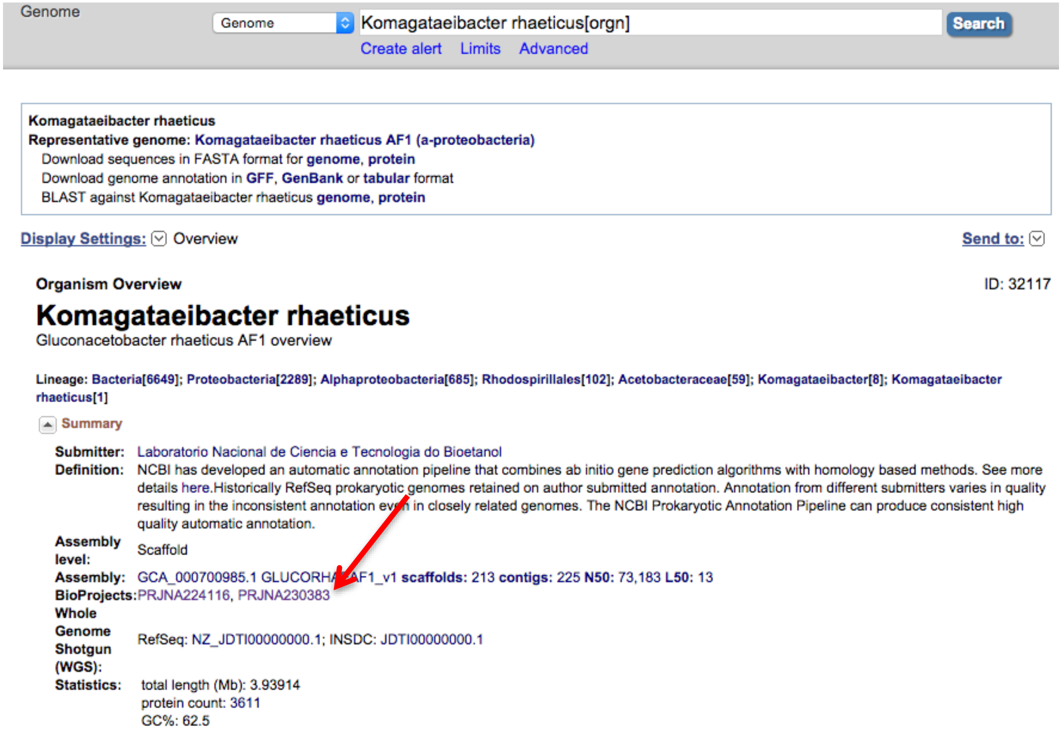
Run the following in a new terminal window:

```
wget
https://download.asperasoft.com/download/sw/connect/3.8.3/ibm-
aspera-connect-3.8.3.170430-linux-g2.12-64.tar.gz

./ibm-aspera-connect-3.8.3.170430-linux-g2.12-64.sh

prefetch -X800G --ascp-path
"~/.aspera/connect/bin/ascp|~/.aspera/connect/etc/asperaweb_id
_dsa.putty" -o SRR1156953.sra SRR1156953

fastq-dump --readids --split-files --gzip SRR1156953.sra
```

Let’s get some Next Generation Sequencing data – Getting data from the Short Read Archive (SRA)

We will start by accessing the National Center for Biotechnology Information’s Short Read Archive to download some public data. We are going to assemble the genome sequence of the bacterium *Komagataeibacter rhaeticus*, this is a bacterium isolated from Kombucha tea (https://en.wikipedia.org/wiki/Kombucha), and it is characterized by producing high levels of cellulose.

Find the Genome Project page for this species, go to http://www.ncbi.nlm.nih.gov/, select the genome database and use “*Komagataeibacter rhaeticus*” as your query term, you should get a single result, as show in the figure below. In the lower part of the webpage, please check details about this genome assembly and write that down for future reference.



Genome

Genome: Komagataeibacter rhaeticus[orgn]

Komagataeibacter rhaeticus
Representative genome: Komagataeibacter rhaeticus AF1 (a-proteobacteria)
Download sequences in FASTA format for **genome**, **protein**
Download genome annotation in **GFF**, **GenBank** or **tabular** format
BLAST against Komagataeibacter rhaeticus **genome**, **protein**

Organism Overview ID: 32117

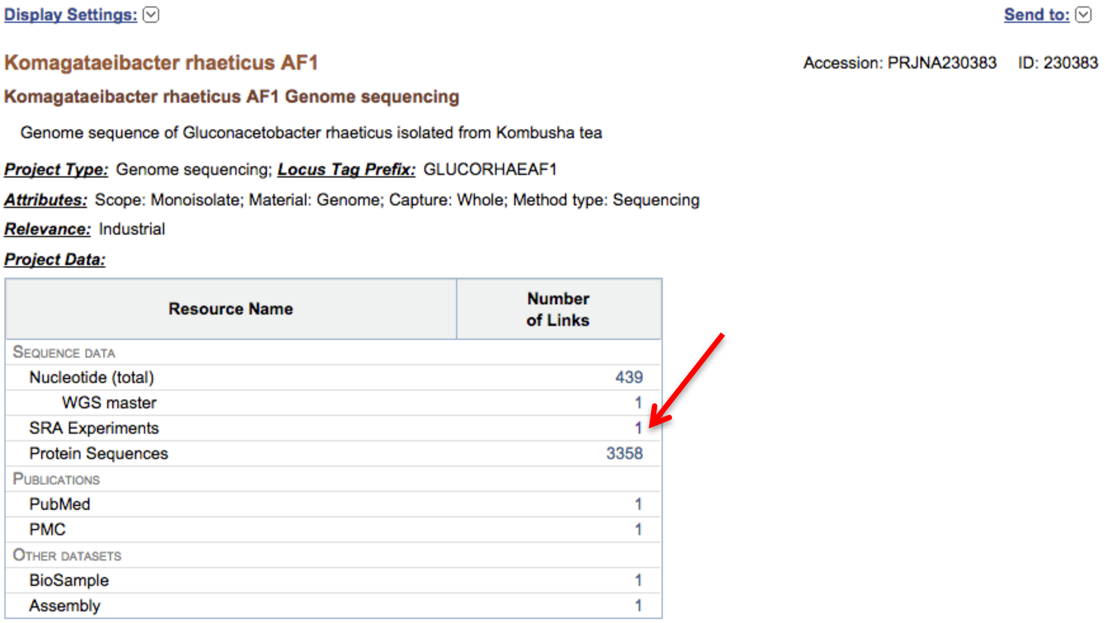
Komagataeibacter rhaeticus
Gluconacetobacter rhaeticus AF1 overview

Lineage: Bacteria[6649]; Proteobacteria[2289]; Alphaproteobacteria[685]; Rhodospirillales[102]; Acetobacteraceae[59]; Komagataeibacter[8]; Komagataeibacter rhaeticus[1]

Summary

Submitter: Laboratorio Nacional de Ciencia e Tecnologia do Bioetanol
Definition: NCBI has developed an automatic annotation pipeline that combines ab initio gene prediction algorithms with homology based methods. See more details here.Historically RefSeq prokaryotic genomes retained on author submitted annotation. Annotation from different submitters varies in quality resulting in the inconsistent annotation even in closely related genomes. The NCBI Prokaryotic Annotation Pipeline can produce consistent high quality automatic annotation.
Assembly level: Scaffold
Assembly: GCA_000700985.1 GLUCORHAF1_v1 **scaffolds:** 213 **contigs:** 225 **N50:** 73,183 **L50:** 13
BioProjects: PRJNA224116, PRJNA230383
Whole Genome Shotgun (WGS): RefSeq: NZ_JDTI00000000.1; INSDC: JDTI00000000.1
Statistics: total length (Mb): 3.93914
protein count: 3611
GC%: 62.5

Click on the BioProject link for PRJNA230383 (see red arrow in the figure above), this will leave you a more detailed page with cross-references, among many other things, to the raw data used to build the assembly, as shown in the figure below. Follow the link to the SRA (red arrow), from there we will download the sequences of the short reads that we will use to assemble this genome.



Komagataeibacter rhaeticus AF1

Accession: PRJNA230383 ID: 230383

Komagataeibacter rhaeticus AF1 Genome sequencing

Genome sequence of Gluconacetobacter rhaeticus isolated from Kombusha tea

Project Type: Genome sequencing; ***Locus Tag Prefix:*** GLUCORHAEAF1

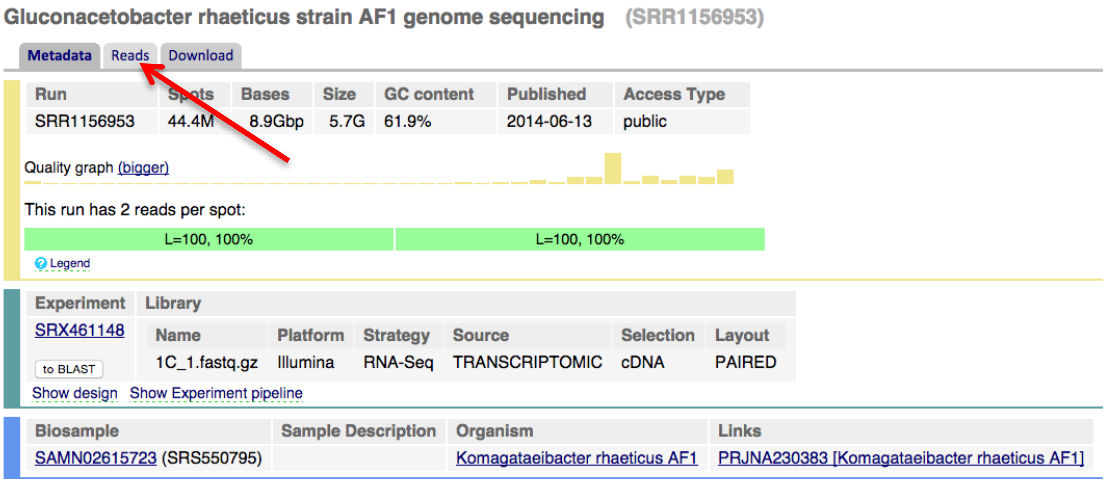
Attributes: Scope: Monoisolate; Material: Genome; Capture: Whole; Method type: Sequencing

Relevance: Industrial

Project Data:

Resource Name	Number of Links
Sequence data	
Nucleotide (total)	439
WGS master	1
SRA Experiments	1
Protein Sequences	3358
Publications	
PubMed	1
PMC	1
Other datasets	
BioSample	1
Assembly	1

For details on how to use the SRA, please check: http://www.ncbi.nlm.nih.gov/Traces/sra/
Once you are in the SRA page, follow the link with the text: **SRR1156953**, this will lead you to the page shown in the following figure. Once there click on the **Reads** tab (read arrow in the figure):



Gluconacetobacter rhaeticus strain AF1 genome sequencing (SRR1156953)

Run	Spots	Bases	Size	GC content	Published	Access Type
SRR1156953	44.4M	8.9Gbp	5.7G	61.9%	2014-06-13	public

Quality graph (bigger)

This run has 2 reads per spot:

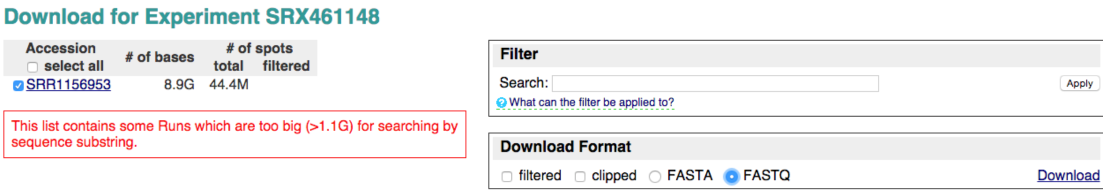
L=100, 100%	L=100, 100%

Experiment	Library					
	Name	Platform	Strategy	Source	Selection	Layout
SRX461148 to BLAST	1C_1.fastq.gz	Illumina	RNA-Seq	TRANSCRIPTOMIC	cDNA	PAIRED

Show design Show Experiment pipeline

Biosample	Sample Description	Organism	Links
SAMN02615723 (SRS550795)		Komagataeibacter rhaeticus AF1	PRJNA230383 [Komagataeibacter rhaeticus AF1]

In the reads page, look for the button "Filtered download" and click on it. This will lead you to a similar page to the one shown in the following figure:



Download for Experiment SRX461148

Accession □ select all	# of bases	# of spots total	 filtered
☑ SRR1156953	8.9G	44.4M	

This list contains some Runs which are too big (>1.1G) for searching by sequence substring.

Filter

Search: [ ] Apply

What can the filter be applied to?

Download Format

□ filtered □ clipped ○ FASTA ● FASTQ Download

In the new page, make sure that the option "**Download format**" is set to **FASTQ**, and then click on the link **Download**. This will download a several GB file from the SRA, this can take a while, go for a coffee.

Alternatively, and perhaps a lot faster, you can use the SRA toolkit to download the data from the command line. First, make sure to install the sra toolkit:

```
apt install sra-toolkit
```

Then run the following command (first it is much better to use prefetch and aspera connect to speed up the download, check page 2):

```
fastq-dump --readids  --split-files --gzip SRR1156953.sra
```

This will download the raw reads from NCBI's SRA.

What do the arguments `--readids`, `--split-file` and `--gzip` do?

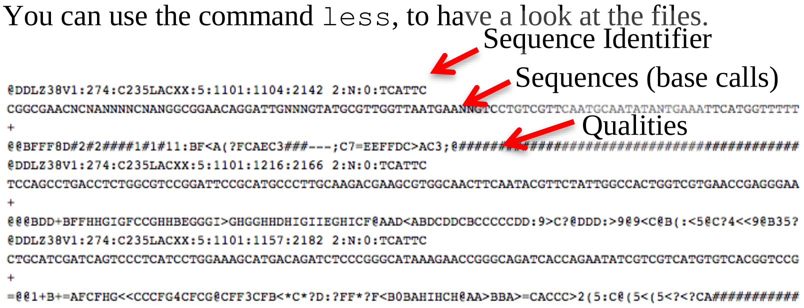
Now download the following files to your desktop, they are in the Moodle system

qc1.fq.gz
qc2.fq.gz
qc3.fq.gz
qc4.fq.gz
single_data.fastq.gz

Visualizing your files

Once the files are ready, we can have a look at the FastQ files. We will spend some time understanding how the sequence data is stored in those files.

You can use the command `less`, to have a look at the files.



```
@DDLZ38V1:274:C235LACXX:5:1101:1104:2142 2:N:0:TCATTC
CGGCGAACNCNANNNNCNANGGCGGAACAGGATTGNNNGTATGCGTTGGTTAATGAANNGTCCTGTCGTTCAATGCAATATANTGAAATTCATGGTTTTT
+
@@BFFF8D#2#2####1#1#11:BF<A(?FCAEC3###---;C7=EEFFDC>AC3;@###########################################
@DDLZ38V1:274:C235LACXX:5:1101:1216:2166 2:N:0:TCATTC
TCCAGCCTGACCTCTGGCGTCCGGATTCCGCATGCCCTTGCAAGACGAAGCGTGGCAACTTCAATACGTTCTATTGGCCACTGGTCGTGAACCGAGGGAA
+
@@@BDD+BFFHHGIGFCCGHHBEGGGI>GHGGHHDHIGIIEGHICF@AAD<ABDCDDCBCCCCCDD:9>C?@DDD:>9@9<C@B(:<5@C?4<<9@B35?
@DDLZ38V1:274:C235LACXX:5:1101:1157:2182 2:N:0:TCATTC
CTGCATCGATCAGTCCCTCATCCTGGAAAGCATGACAGATCTCCCGGGCATAAAGAACCGGGCAGATCACCAGAATATCGTCGTCATGTGTCACGGTCCG
+
=@@1+B+=AFCFHG<<CCCFG4CFCG@CFF3CFB<*C*?D:?FF*?F<B0BAHIHCH@AA>BBA>=CACCC>2(5:C@(5<(5<?<?CA###########
```

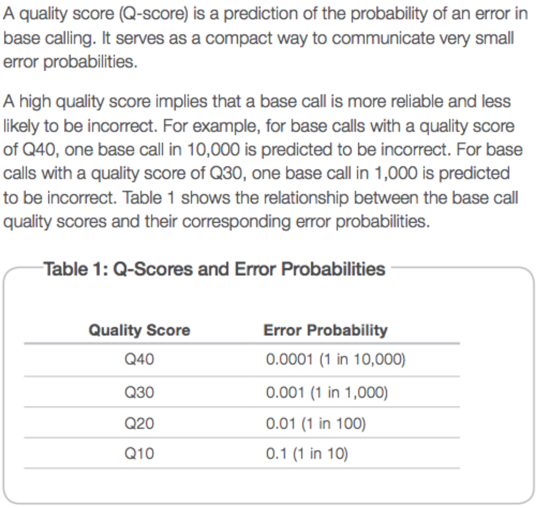
Look at the files which name starts with "SRR1156953_". Have you uploaded two different files? Why? What does the _1 and _2, in the names of the files mean?

Each read in a FastQ file has 4 lines in the file. The first line has information about the identification of the reads, which include the name of the sequencer, the Id of the flowcell, and number of the lane and the position of the read (cluster) in the sequencing surface. The second line has the actual base calls, the third line is a spacer, and the forth and last line are the qualities of the base calls from the second line. Now make sure you check both files, what are the differences and similarities among them? Notice that the order of the reads is the same in both files and many programs would assume this when using files for paired-end sequencing in FASTQ format. For more information about the FastQ format check:

- https://en.wikipedia.org/wiki/FASTQ_format
- http://nar.oxfordjournals.org/content/38/6/1767.long

What does the Q value mean?

$$Q_{sanger} = -10\log_{10} p$$



A quality score (Q-score) is a prediction of the probability of an error in base calling. It serves as a compact way to communicate very small error probabilities.

A high quality score implies that a base call is more reliable and less likely to be incorrect. For example, for base calls with a quality score of Q40, one base call in 10,000 is predicted to be incorrect. For base calls with a quality score of Q30, one base call in 1,000 is predicted to be incorrect. Table 1 shows the relationship between the base call quality scores and their corresponding error probabilities.

Table 1: Q-Scores and Error Probabilities

Quality Score	Error Probability
Q40	0.0001 (1 in 10,000)
Q30	0.001 (1 in 1,000)
Q20	0.01 (1 in 100)
Q10	0.1 (1 in 10)

http://www.illumina.com/documents/products/technotes/technote_understanding_quality_scores.pdf

Evaluating the quality of your data

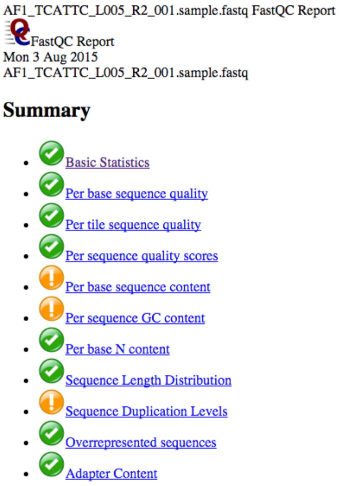
Now you have your short sequence reads in the server, we will generate some statistics that will help us evaluate the quality of the data that you have.

Make you you have installed the program FastQC:

```
apt install fastqc
```

This program will evaluate several parameters of your reads, and we will use that information to, among other things, decide about the steps that we will follow to clean the data.

FastQC will generate a a web page, where you will find a graphical display of the results. Let’s check the web page, you can open it with any browser, e.g., firefox.



AF1_TCATTC_L005_R2_001.sample.fastq FastQC Report

FastQC Report
Mon 3 Aug 2015
AF1_TCATTC_L005_R2_001.sample.fastq

Summary

- ✓ Basic Statistics
- ✓ Per base sequence quality
- ✓ Per tile sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ! Per sequence GC content
- ✓ Per base N content
- ✓ Sequence Length Distribution
- ! Sequence Duplication Levels
- ✓ Overrepresented sequences
- ✓ Adapter Content

You will see something similar to the Figure in the left. It is a list of the different assessment modules run by FastQC.

Let’s start checking the Basic statistics. This will show you the number of reads that are present in your FastQ file, the encoding of the quality scores. By the way how many different encodings are available? How would you distinguish one from another? Here you will also find the length of your reads. Why is that in Illumina Sequencing by Synthesis your reads are always of the same size? How are Ion Torrent or 454 in that regard?

Let’s just think for a moment on the amount of sequencing that you have at hand, and how that relates to your needs.

From the basic statistics module you get that you have sequenced 5.551.645 100bp reads[1]. Remember that you did paired-end sequencing, and this is just one of the file, so you must have the exact same number of reads in the second file. So in total you should have 1.110.329.000bp of raw sequencing data. How is that related to your genome sequence? Check again the Genome Project for this species as shown above. The estimated genome size for *K. rhaeticus* is approx. 4Mbs, with these two pieces if information we can use the Lander/Waterman[2,3] equation to estimate our coverage

[1] Your actual numbers can be different

[2] Lander ES, Waterman MS. Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics. 1988 Apr;2(3):231-9. PubMed PMID: 3294162.

[3] http://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf

$$C = LN/G$$

Which give us approx. 277 times the genome size, which means that in average we will sequence each base 277 times with the amount of data we have generated. Later today you will generate subsets of the reads dataset and a digitally normalized dataset, please come back and compute the estimated coverage for each of these.

You can use the same equation to estimate the amount of sequencing that you would require to achieve a certain coverage for a given organism, you should just have to solve the equation for LN, the total amount of sequencing.

Do you have a genome-sequencing project in mind? Let's make the computation of amount of sequencing required!

Now let's go back to the FastQC results for the other files, we will check the results presented in the other modules together[4].

- What types of warnings/errors were detected? What are the reasons for that?
- Which is the best/worst dataset?
- Which FastQ encoding are used?

It is a good practice to check your newly sequenced reads for contamination. Possible sources of contamination are human DNA, DNA from other bugs sequenced at the same time as yours, bugs co-isolated with yours, the technician DNA and so on. A good strategy is to get a sample of your reads, say 10% and run a BLAST[5] search (perhaps using megablast. **What is the main difference between megablast and a standard blastn search?**[6]) against the NCBI's nt database, and analyze the results with a program such as MEGAN5[7], this usually gives a very good idea whether or not there is something funny regarding contamination. A better alternative is to use a software tool such as blobtools[8] to generate taxon-annotated GC-coverage plots, as those shown below. Discuss what you observe in these two figures.

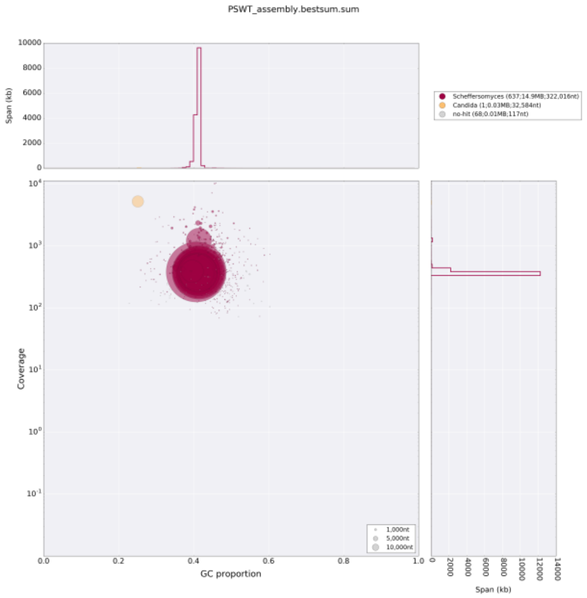
[4] You can also check the FastQC manual: http://www.bioinformatics.nl/courses/RNAseq/FastQC_Manual.pdf

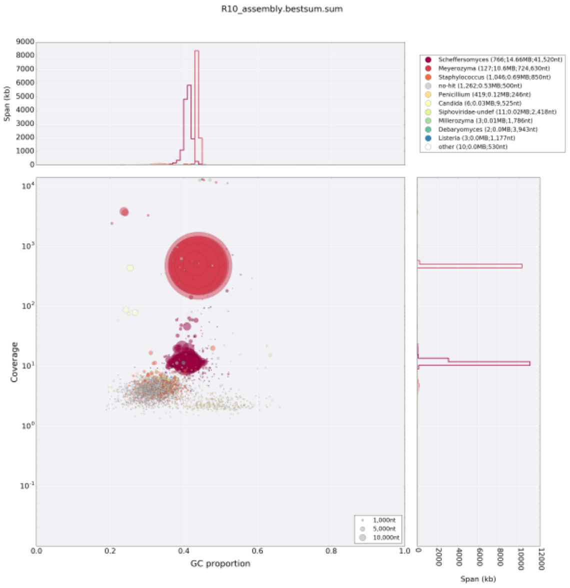
[5] http://blast.ncbi.nlm.nih.gov/Blast.cgi

[6] http://www.ncbi.nlm.nih.gov/blast/html/megablast.html

[7] http://ab.inf.uni-tuebingen.de/software/megan5/

[8] https://blobtools.readme.io/docs





During the quality analysis above, we saw that the quality of the base calls decreasd towards the 3' end of the reads, this is typical of all second generation sequencing technologies currently on the market. We usually remove the worse bases from the 3' end of the read, this is known as quality trimming. It seems that there are not remainder adaptors in the reads, anyway we will run a module to remove any, as this could be helpful for another projects. To carry out these steps we will use Trimmomatic[9], software developed in the Lab. of Björn Usadel in Germany.

Run trimmomatic on the SRR* files as:

```
java      -jar      /usr/share/java/trimmomatic-0.38.jar      PE
SRR1156953_1.fastq.gz      SRR1156953_2.fastq.gz      -baseout
SRR1156953_Clean          ILLUMINACLIP:NexteraPE-PE.fa:2:30:10
SLIDINGWINDOW:4:20 MINLEN:60
```

Go back and look for the application Trimmomatic, we will carry on with the cleaning of our sequences.

Using trimmomatic again, clean the datasets which names start with "**qc**". Select appropriate parameters for each case based on the previous analysis with FASTQC. After cleaning all your files, run FASTQC again to verify your results, you should notice improvements of the FASTQC statistics, otherwise change your cleaning parameters and try again. Discuss your strategies with your colleagues.

As we saw above, the data we have covers the genome several times over. In some cases this would actually create problems. For instance you will need a larger machine to assemble a genome with a larger amount of raw data. Recently, and developed with metagenomics in mind, new techniques have been created that try to reduce the amount of data without loosing information. We will look at read normalization technologies. The most frequently used technology is implemented in *khmer*[10] created by Dr. Titus C. Brown. Pay attention to your instruction there will be a brief explanation about how digital normalization works. For the practical session we will use a much simpler approach, taking a subset of the reads, in addition to the *khmer* package.

Again use the tool "Sub-sample sequences files", and create three datasets:

- Half the reads

[9] http://www.usadellab.org/cms/?page=trimmomatic

[10] http://khmer.readthedocs.org/

- One third of the reads
- One fifth of the reads

Discuss about the cons and pros about these two approaches.

We will use these files for the genome assembly, in the next session. We will not have time for each of you run all the assemblies. So, split into 7 groups, the task for each group are:

Group	Task
1	Subsample 50%
2	Subsample 33%
3	Subsample 20%
4	Normalize 2x
5	Normalize 5x
6	Normalize10x
7	Normalize 20x

Digital normalization (Normalize by median - Khmer)

What is a *k*-mer? It is a DNA sequence of fixed length, *k*. We will talk more about that tomorrow. We will use the tool Normalize by median, but before we have to prepare the input reads. Khmer programs expect the reads to be in interleaved format, i.e., the R1 after the R2 reads. We can use the tool `interleave-reads.py`

```
interleave-reads.py   -o   BASE_interleaved_fastq.gz [11]   --gzip
SRR1156953_Clean_1P.fastq.gz SRR1156953_Clean_2P.fastq.gz
```

Then we will use the tool Normalize by Median, to normalize to a desired depth (see table above)

```
khmer normalize-by-median
```

There are four important parameters for khmer. First it is the *k* value, it is recommended by the developers to leave this as 20. Then we have n_tables and tablesize. The product of these two should be approx. the amount of RAM memory that you could use for your job, and it is used to keep track of *k*-mers. During this course we are limited by the RAM memory available in your desktops, which should be around 4GB (you can check running the command free -g). Usually n_tables=4. Thus, this would lead us to tablesize=1e9[12]. Last, but not least, the fourth parameter, is the desired or target coverage. Generate file with the following target coverages: 2x, 5x, 10x and 20x, make sure to change the names of your output “files”. Do not forget to use your cleaned reads.

After running the digital normalization procedure, please report how many reads remain. It will also be worth it to check these files with FastQC.

Genome size estimation

You can use your own read data to estimate the genome size of your organism under study, this could be a lot simpler to go through the classical approaches, i.e., Feulgen image analysis, flow cytometry, real time PCR.

Use the same interleaved reads file from the previous step. You should explore the estimation performance using different datasets to have a feeling about how this could vary. Look for the tool kmergenie, this will estimate the genome size of your organism under study and try to predict the best *k*-mer for assembly, both using *k*-mer statistics.

[11] Instead of BASE you should use group1, group2, and so on

[12] http://khmer.readthedocs.org/en/v1.1/choosing-table-sizes.html

```
gunzip BASE_interleaved_fastq.gz
kmergenie BASE_interleaved_fastq
gzip BASE_interleaved_fastq
```

Genome assembly

We will use Velvet[13] to assemble the datasets you have generated before and study the effect of sequencing coverage on the quality of the assemblies.

By the way what is the difference between a contig and a scaffold?

Now, let's assemble the same dataset using Velvet. Velvet expects the reads in the interleaved format, so make sure to use the file `BASE_interleaved_fastq.gz`

First we will need to create a hash table, this is done by the tool velveth. The most important parameter for this tool is the length of the K-mer. We will use the best kmer suggested by kmergenie (replace the number 31 as appropriate).

```
velveth myAssemblyDir 31 -fastq.gz -interleaved -shortPaired
BASE_interleaved_fastq.gz
```

Once we have created the hash table, we can build the *de bruijn* graph and extract the contigs, this is carried out by the tool velvetg. Velvetg uses as i.e., the hash table, but is has many parameters that can be set and that will affect the assembly, please check the Velvet manual[14] and discuss with your partners the different options.

```
velvetg myAssemblyDir/ -cov_cutoff 2 -min_contig_lgth 200
```

Now we will compare the different assemblies using typical genome assembly statistics[15,16]. For this we will use the tool Quast. Compute these statistics for each of the assemblies that you have. Discuss with your partners and instructors the meaning of the different statistics.

```
quast myAssemblyDir/contigs.fa
```

Make and assembly only using single reads. How does that affect the quality of your assembly? Look for the contig with the highest coverage in your best assembly. Why is that its coverage is so high? It is many times the average coverage. You can extract the sequence for that contig using the tool `extractseq` from EMBOSS. Get it and BLAST it using the NCBI web interface. Can you figure out what it is?

[13] In an actual research project you will perhaps not use Velvet, or at least not alone. You will use several assemblers, with different parameters. For time's sake we are only using velvet, but the basic approach is similar with all assemblers.

[14] https://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf

[15] https://en.wikipedia.org/wiki/N50_statistic

[16] http://bioinformatics.oxfordjournals.org/content/29/8/1072