Bioinformatics Analysis Exercise: Introduction

In this workshop, you will use elementary bioinformatics methods to analyze two next-generation sequencing (NGS) data sets that were created to answer two different biological questions. Our ultimate goal is to transform the raw reads in the data sets and extract information that we can then interpret and use to answer the questions at hand.

On day 1, we will deal with preprocessing and exploring the two read data sets. At the beginning of any NGS data analysis you are left with raw reads, which are usually directly obtained from a sequencing facility. Before using them in more specific analysis, it is important to verify and ensure their quality. This encompasses inspecting the raw data to identify or correct unusual patterns or biases related to the sequencing process itself. This step can also give you early hints on whether something has gone bad in the earlier steps of your experiment.

On day 2, we will deal with the early downstream part of bioinformatics analysis for the two data sets preprocessed on day 1. The particulars of downstream analysis naturally depend entirely on the biological question that you are trying to answer, therefore the second day will consist of two parts, one for each data set.

The first experiment that we will analyze is about detecting genetic variations in a sample of E. coli DNA. Here, we will extract the genetic variations present in the sequenced strain in comparison to a reference strain using the aligned read data from day 1. The final output of our analysis will then consist of a list of putative genetic variations present in our analyzed strain.

The second experiment that we chose for this workshop is about gene expression in S. cerevisiae and consists of RNA-Seq reads. We will further analyze the aligned read data from day 1, ultimately getting to a read count for each known S. cerevisiae gene. In RNA-Seq the read counts for each gene are essentially measures for expression levels, even though they still have to be corrected for statistical biases before being interpreted as such.

Day 1

Today, we will acquire, preprocess and examine our data sets. The first steps we will take are mostly related to ensuring the quality of the reads. They are therefore very similar for the two data sets, since they depend less on the biological question but mostly on the technical aspects in relation to the sequencing process itself. An essential last step before further analysis is mapping the reads, which can be described as putting them in relation to a reference genome through sequence alignment. These steps are needed for both our differential expression and variation calling data sets.

Part 1: Installation of Tools and Experiments Overview

For today's exercises, you will need to use the following programs which are not available by default on Linux/UNIX systems.

General Tips for installing the programs

- While you can place the downloaded software anywhere, we recommend to:
    - Put the programs in a new subdirectory named "software", in your home directory to separate them from your data and the rest of your files
    - Rename long, complicated names of program directories to short, simple names so you do not have to type so much later when using the programs and can find them more easily
- **Some** of the programs are available in binary format. This means that you can just download and unpack them - and you will be ready to go.
- If binary files are provided, be sure to download the ones fitting your system architecture. To find out your system architecture, enter "uname -a" in a terminal.
- **Most programs** on Linux are provided to you as source code. In order to use them, you have to compile this source code into binary format first. These programs typically include instructions on how to go about compilation.
- GitHub is a common platform where programs are made available in source code. To download a project from GitHub you can use the command "git clone <URL>" or download it directly as a .zip file from github.com
- Java programs (.jar files) can be started with: "java -jar <.jar File> arguments"
- Test whether each program works by calling it with "--help" as parameter after installation.

Programs that you will need today

- **Download: fastq-dump**, part of the **sratools** package. For downloading reads from the SRA. http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=show&f=software&m=software&s=software
- **Download: FASTQC**. Quality control for read data http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

- **Download: Trimmomatic**. Filtering of read data
  http://www.usadellab.org/cms/?page=trimmomatic

- **Download+Compile: NextGenMap**. Mapping of genomic reads
  https://github.com/Cibiv/NextGenMap

  (See the INSTALL file for installation instructions)

- **Download: HISAT**. Mapping of transcriptomic reads
  http://www.ccb.jhu.edu/software/hisat/index.shtml

- **Download+Compile: samtools**. Working with files containing read alignments
  https://github.com/samtools/samtools

  (See the INSTALL file for installation instructions)

- **Download: IGV**. Visualizing read alignments
  https://www.broadinstitute.org/software/igv/log-in

Experiments Overview

In the following sections, you will apply bioinformatics analysis to the following two experiments.

	Experiment 1	**Experiment 2**
Organism	Escheria coli	Saccharomyces cerevisiae
Experiment Type	Genomic Sequencing	RNA-Seq
Main Goal	Detect genomic variations	Quantify gene expression
Experiment SRA ID	SRX158902	SRX1586343

Important: The following parts 2 to 4, inclusively, should be executed separately for each experiment!

Part 2: Obtaining the NGS read data set

First, we will acquire and make ourselves familiar with the read data from the experiments introduced before. While for your own experiments, you will likely retrieve reads from the sequencing facility - here we will instead download them from the Sequence Read Archive (SRA), a database containing metadata and reads for many NGS data sets.

1. Use a web browser to search for the experiment in the SRA using its SRA Accession. *SRA Webpage: http://www.ncbi.nlm.nih.gov/sra/*

2. How many sequencing runs are there in the experiment?

   For the next steps, select one of the sequencing run(s).

3. How many bases were sequenced in the run?

4. Does the run contain single-end or paired-end reads?

5. What sequencing technology was used in the run?

6. Copy the SRA ID of the run and use the command “fastq-dump” to download the reads in fastq format. For this workshop, we will limit the number of reads downloaded for each experiment to 1 million.

   The command “fastq-dump --split-3 -X 1000000 <SRA Run Accession>” will download the first million reads in the data set from the SRA.

7. What does the “--split-3” option do and why do you need it?

8. To make yourself familiar with the fastq format, explore any of the fastq files using a command line tool of your choice (e.g. less). How many lines does a single read occupy?

   It is not the best idea to open (very) large files in graphical text editors as they will likely become unresponsive. Command line tools are generally much more suitable for working with large amounts of data.

9. How does the .fasta format differ from the .fastq format?

10. If the run is paired-end, how are the read pairs stored in the fastq files?

11. How many individual reads (and how many read pairs) are there in the sample?

12. What is the quality value of the 4th base in the 1st read (of both pairs, if paired-end)?

13. What is the sequence of the 3210th read in the sample (for paired-end: use the 3210th of the first reads in the pair)?

14. Check the total size of the fastq files using “ls -lah”. Then compress all the fastq files with “gzip”. How much smaller are the files after compression?

    It is a good idea to archive very large files to save disk space, but some analysis steps can only be done with uncompressed files. Many tools that require .fastq as input will also accept gzipped .fastq.gz files.

Bonus exercises:

- Find a way to convert the downloaded reads from .fastq to .fasta format.

- Building on exercise 12 from above, what is the corresponding probability that the base was incorrectly called in the sequencing process, given this quality value?

  To answer this question, you will have to look up how base quality values are encoded in fastq files.

Part 3: Quality Control for Raw Reads

Before starting out with the main part of our analysis, we want to ensure that the data set is free from technical artifacts that would likely introduce strange biases in the results.

1. **Produce a quality summary report for the fastq files using FastQC.**

   To run fastqc, use the command “fastqc <fastq files>”. The reports will be output in zipped format, after unpacking you can view them in a web browser of your choice.

   In the course of the next tasks, we will interpret the quality control report and what they mean for our analysis. We encourage you to, besides doing the exercises below, browse the FastQC report according to your interests.

   For more information on the interpretation of specific sections in the report, you can consult the FastQC documentation at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/.

   a. How long are the reads? Interpret the sequence length distribution.

   b. Interpret the per base sequence quality.

   c. What is the average and per-sequence GC content, and does this fit our expectations for the sample?

   d. What are possible reasons for deviations from the expected GC content?

   e. Interpret the per base sequence content. Consider what you would expect to see under ideal circumstances for this experiment, and how your observation deviates from this.

      Note: Per base sequence content refers to how often each of the 4 different nucleotide types appears at each position, over all reads.

   f. Which sequences are overrepresented in the reads? Could this be due to technical artifacts, if yes, what kind?

   g. Were any standard adapters found in the reads?

   h. Investigate the sequence duplication level. Why do we want to avoid sequence duplication? What are possible reasons for a high sequence duplication level?

2. **Deal with the identified technical artifacts, if any**

   If you identified adapter sequences or low-quality regions in the reads using FastQC, these problems should be corrected before further analysis.

   Trimmomatic is a tool that automatically removes Illumina adapter sequences and low-quality ends of reads. The details on what further steps the program can applie can be found at http://www.usadellab.org/cms/?page=trimmomatic.

   We will use the following command to run trimmomatic. Before using it, you have to replace the placeholders with the right values for each experiment.

   java -Xmx8g -jar <Path To Trimmomatic.jar> PE <Path to Reads 1.fastq> <Path to Reads 2.fastq> <Trimmed Paired Output 1.fastq> <Trimmed Unpaired Output 1.fastq> <Trimmed Paired Output 2.fastq> <Trimmed Unpaired Output 2.fastq> ILLUMINACLIP:<Adapters Fasta Filename>:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36

   Use “TruSeq3-PE-2.fa” as adapter list for filtering. (for the placeholder <Adapters Fasta Filename>). The file is located in the “adapters” directory, a subdirectory of the trimmomatic directory you downloaded and unpacked. Be sure to refer to the adapter list file by its correct relative path based on the directory where you execute trimmomatic.

   a. What filtering steps will the above command apply to our reads?

      To understand the filtering steps in detail, consult the trimmomatic documentation.

   b. If adapter sequences, or low quality ends were identified by FastQC, process the reads with trimmomatic by using the template command from above. Store the paired trimmed reads as “<original_name_1/2>_trimmed.fastq”. Discard the unpaired reads.

      If you used trimmomatic for this data set, use the trimmomatic output fastqc files or all later analysis, including read mapping.

   c. If you have chosen to apply trimmomatic to improve the quality of your reads in this experiment, execute / answer the following exercises:

      i. Repeat the FastQC analysis in Part 3, step 1 for the trimmed reads.

      ii. Were the issues resolved by trimmomatic?

      iii. How many reads did you lose due to filtering?

iv. What issues have persisted, and how could they influence the later stages of analysis?

v. If overrepresented sequences are still present, perform a BLAST search to identify their origin.

vi. Why is it possibly not a good idea to manually perform additional filtering on the data, for example to correct strange patterns or filter out overrepresented sequences that you do not fully understand?

Part 4: Mapping the Reads

After examining the reads for possible technical problems and correcting them as good as possible while being careful not to introduce new biases ourselves, it is time to *map* them. Mapping refers to alignment of reads to a reference sequence. This is an essential step for both the variation calling and RNA-Seq experiments: For variation calling, we use the alignments to infer genetic variations by looking for identical mismatches occurring throughout multiple alignments at a single position in the genome. For RNA-Seq, we can combine the alignments to the genome with annotations to find out which read mapped to which gene.

For mapping, we first need to acquire a reference genome appropriate to our purpose. Here, we will simply use generic assemblies for our two organisms of interest.

Here, analysis will also start to diverge slightly between the two data sets. Many different read mapping programs exist, each being optimized for different applications. We will therefore use a different read mapper for the genomic and transcriptomic reads.

To help you choose the right read mapper for your real experiments, you can use *Teaser (available at* http://teaser.cibiv.univie.ac.at/).

We will use NextGenMap, a genomic mapper, for mapping the E. coli and HISAT, a RNA-seq read mapper, for mapping the S. cerevisiae data.

Based on your preference, you can do the following steps 1, 2 and 3 all at once, or separately for both data sets.

1. Download the reference genomes.

   (a) E. coli genomic data set. Download the Escheria coli genome from "http://www.cibiv.at/~moritz/workshop/E_coli_dbh.fasta".

   (b) S. cerevisiae RNA data set. Navigate the Ensembl database to download the compressed S. cerevisiae reference genome and then decompress it. Rename the decompressed file to "S_cerevisiae.fasta".

   Ensembl webpage: "http://www.ensembl.org/index.html"

   The Ensembl repository for an organism contains multiple compressed .fasta files, usually corresponding to individual chromosomes. For this analysis download the "toplevel" sequence containing all chromosomes.

2. Build the mapper search indices for the reference genomes

   Mappers generally require that you to index (prepare for efficient search) reference genomes before mapping reads to them for the first time.

   (a) E. coli genomic data set. NextGenMap automatically generates the reference index if it does not exist already. Therefore, you do not need to do anything for the E. coli data set in this step.

   (b) S. cerevisiae RNA data set. Use “hisat-build” to index the reference genome for mapping reads to it with HISAT. Use “S_cerevisiae” as name for the index.

3. Map the reads

   (a) E. coli genomic data set. Map the genomic reads to the E. coli reference using the NextGenMap read mapper. Use 4 threads. Name the output file “ecoli_mapped.sam”.

   Use the command “ngm” to run NextGenMap. Usage instructions can be viewed using the “--help” option.

   (b) S. cerevisiae RNA data set. Map the RNA-seq reads to the yeast reference using the HISAT spliced read mapper. Use 4 threads. Name the output file “yeast_mapped.sam”.

   Use the command “hisat-build” to create the index for the reference genome. Use the command “hisat” to map the reads to the index, specifying the parameters as required. Usage instructions for both commands can be viewed using the “--help” option.

4. Familiarize yourself with the SAM format by browsing the mapper output files using “less -S”.

   The SAM format specification can be found at https://samtools.github.io/hts-specs/SAMv1.pdf

5. Was the last read in the SAM output mapped, if yes, to which chromosome and which position on the chromosome?

6. What is the edit distance to the reference of the first read in the SAM file that was mapped?

7. Use “grep” to output all read alignments in the SAM file that have an edit distance of zero.

8. Convert the SAM alignment output to BAM format, a more efficient format to store the alignments.

   a. Use the command “samtools view” to convert the SAM file to a compressed binary BAM file with the name “<yeast/ecoli>_mapped.bam”. If they support

them, programs tend to be able to process BAM files more efficiently than SAM in terms of runtime.

Use the "--help" option with "samtools view" to find out how to solve this exercise.

b. Use the command "samtools sort" to sort the BAM file, naming the output "<yeast/ecoli>_mapped_sorted.bam".

Use the "--help" option with "samtools sort" to find out how to solve this exercise.

c. Use the command "samtools index" to index the sorted BAM file, which will be needed for displaying the data with a variety of programs, including IGV.

d. Use the command "samtools flagstat" on the sorted BAM file to output basic mapping statistics and interpret them.

   i. How many reads were mapped on the forward strand?

   ii. How many reads were mapped on the reverse strand?

   iii. How many reads were not mapped?

9. Filter the alignments

   a. Filter the sorted BAM files you just created containing the alignments: Discard all reads that have mapping qualities smaller than 5. This is a recommended step in order to remove weakly supported alignments from the results.

   Name the filtered output like "<yeast/ecoli>_mapped_sorted_filtered.bam". This file will be already sorted, since the unfiltered input was also sorted! However, you must still index it, the same way as in step 8.c.

   "samtools view" has an option "-q" can be used to specify the mapping quality threshold.

   b. How do the mapping statistics ("samtools flagstat") differ between the filtered and unfiltered BAM files? How many reads were removed due to the applied filtering?

   c. Why, in general, could an alignment be assigned a low mapping quality?

   d. What possible values are there for the mapping quality of a read?

Bonus exercises:

- Write a (bash) script that automatizes the conversion of SAM to BAM, filtering for mapping quality and indexing. The mapping quality threshold should be 5. The script

must accept two parameters: First, the input SAM file. Second, the name of the final output BAM file. Also, your script should automatically delete the intermediate files.

- Write a (bash) script that shows, for each possible mapping quality value, the number of reads in a SAM file with that mapping quality value.

Part 5: Explore the E. coli Experiment Alignment Data

Visually inspecting read alignments is an essential way of exploring your NGS data and can reveal a lot of information. For this task, we will use the IGV (Integrative Genomics Viewer).

1. Open IGV and load the E.coli reference genome that you downloaded in the previous step.

2. Load the filtered **and** unfiltered E. coli mapping data.

3. Rename the newly added lanes containing filtered and unfiltered mapping data to give them a meaningful description.

4. Navigate to reference sequence “gi|170079663|ref|NC_010473.1|”, position 2,470,923-2,472,649. How does this region differ between filtered and unfiltered data?

5. Navigate to reference sequence “gi|170079663|ref|NC_010473.1|” position 2,474,542-2,474,648. Zoom in until you can discern the alignments.
    a. Can you identify mismatches between the sample and the reference?

    b. What can we infer when an identical mismatch occurs throughout many reads at the same position in the genome?

    c. Right-click to color alignments by “insert size and pair orientation”. Which alignments are marked, and what does this tell us about the alignment?

    d. Given the data that you are seeing, how would you go about calling differences between our sample organism and the reference organism, using these alignments?

    e. How you would go about assigning confidence scores to the called differences?

6. Navigate to reference sequence “gi|170079663|ref|NC_010473.1|”, position 2,057,551-2,061,005.
    a. What strange patterns can you identify in this region?
    b. Can you think of possible biological explanations of what you are observing?

7. Download annotations from <Q> and load them into IGV. What do the annotations reveal about the region?

8. Save the session and close IGV.

9. Open IGV again and load the session you just saved again.

10. What meaning does it have if nucleotides are small / capital letters in the sequence lane at the bottom?

11. Copy the reference sequence that is annotated as gene <R>.

Part 6: Explore the Yeast Experiment Alignment Data

1. Start a new IGV session and load the unfiltered yeast alignment data.

2. Download annotations in GTF format for Sacharomyces cerevisiae from "ftp://ftp.ensembl.org/pub/release-84/". Unzip them and load them into IGV. Right click on the left side of the annotation track and select the option to expand the track, in order to show all annotations.

3. Navigate to sequence "IV", position 575,322-583,160

    a. What do you see in the alignments in this region, and how do your observations relate to your expectations for the experiment type?

    b. Investigate the annotations in the region of interest. What do they tell you about the transcript?

4. Navigate to sequence "IV", position 525,651-541,328.

    a. What do you see in the alignments in this region, and how do your observations relate to your expectations for the experiment type?

    b. Investigate the annotations in the region of interest. What do they tell you about the transcript?

5. Can you find other splice sites?

6. Can you identify splicing events that were called by the mapper but that you do not trust?