‘Applied Genome Research’

Table of contents

1 Basic commands and comments

All bioinformatical work was carried out in the lqxterm, the so-called console. This is more efficient than xterm, since there is more space available.

- Xterm uses only two servers, while lqxterm can use several servers
- To open the lqxterm the word lqxterm is entered in the xterm
- statler: an example of the server's name
- The default directory is the homes (homes/username)

1.1 Basic commands

Table 1: Symbols or commands and their meanings.

Symbol/command	Meaning
$	Displays the command line, but not with is entered in the console
/	Separates the different layers of folders of e.g. a file path File path example: /folder1/ folder2/ folder3/ file1
./	Represents the current directory
/..	Get up a directory
cd	'change dictionary', if you want to enter a new folder/directory A space must follow to cd, e.g.: cd /folder1/ folder2/ folder3/ file1
cd	Type only cd and you will arrive back in the "homes"
cd ..	Get up one folder hierarchy
Tab	Auto-completion of commands / directory name
F5	Updating the directory
Kwrite	A program that was used to open large files
Strg + C	Terminates all processes that run in the console
Strg + D	Cancels a process
Strg + T	Opens a new tab
Arrow keys (up and down)	Using the keys, previously entered commands can be called back to save time
ls	Displays the files in a folder

ls -l	Displays history with time and generated files
--help or -help Or enter only the programme's name	Invokes the help and gives information about how to use a program. Important, because the arguments are different for every program. Examples: python - help
mkdir	Generate a new file via the console
|	So-called pipe, used to compress the output directly via the command gzip and then save it in a new folder
head	To display the first lines of a file

Table 2: File extensions and their meanings.

File extensions	**Meaning**
.txt	Text file
.html	Hyper Text Markup Language, HTML-file
.abi oder .ab1	Trace-file
.quala	QUALA
metrics	Matlab-scripts, e.g. metrics
.jar	Java-file
.py	Pythonscript
.pl	Pearlscript
.fastq oder .fq	FastQ-file
.fa oder .fasta	FASTA-file
.zip	Compressed (Zip-) file
.gz	Compressed (gzip) file
.SAM	Sequence Alignment/Map format, .SAM-file
.BAM	Binary Alignment/Map format, version of .SAM-file, .BAM-file
.idx	Indexed file containing FASTA index
.bai	Indexed file containing BAM index
.vcf	Variant calling format
.GFF	GFF-file (General Feature Format)

2 Course summary

For the first week of the course, sequencing data (paired-end, Illumina) of *Arabisdopsis thaliana* Niederzenz 1 (Nd-1) were used. First, common methods such as DNA isolation and the sequencing technologies used to generate reads were discussed. FastQC was used to assess the quality of the reads and finally a trimming with Trimmomatic was carried out as part of the quality improvement process.

Afterwards, the Nd-1 reads were assembled to contigs via SOAPdenovo2. The quality of the assembly was analysed using different methods and statistics. Next, scaffolding was performed with SSPACE and compared to the SOAPdenovo2 scaffolding.

AUGUSTUS was applied for the structural annotation of the generated scaffolds. After the mRNA, CDS, CDS exon, and peptide sequences of the annotated genes were extracted, a sequence comparison of the Nd-1 scaffolds with the Col-0 reference sequence was performed using BLAST. In this way, the assembly was checked for its completeness. Next, Reciprocal Best Hits (RBHs) against Col-0 were identified. Finally, the functional annotation of the best hits in Col-0 was transferred to the corresponding Nd-1 gene.

BWA MEM was used for the mapping of Nd-1 sequencing reads against the Col-0 reference sequence. The resulting files were prepared for the use of the "Genome Analysis Tool Kit" (GATK). Removal of PCR duplicates and indexing were some of these steps. Next, variant calling with GATK was performed. For this purpose, an read realignment around putative indels was performed to increase the accuracy in indel identification. A local *de novo* assembly was used to resolve critical indel candidates. Variations between Col-0 and Nd-1 were identified using the HaplotypeCaller. The variations were extracted and filtered. Finally, the variation frequency was determined for the resequenced interval. The functional effects of the variations were determined and evaluated with SnpEff.

In the second week, RNA-Seq data from Col-0 wild type (WT) and a 3xmyb mutant was used. The aim was to identify differentially expressed genes (DEGs). Once again, common methods such as RNA isolation and gene expression quantification technologies were discussed, because they are essential to carry out a *de novo* transcriptome or reference-based assembly. The quality of the RNA-Seq data provided was first analysed using FastQC. Reads were trimmed prior to transcriptome assembly. STAR was deployed to map the Col-0 WT and 3xmyb reads to the reference sequence. Afterwards, featureCounts was used to count the mapped reads per gene (performed for both accessions).

Finally, a statistical analysis was carried out with DESeq2 using the mapped and counted reads.

Additionally, random replicates were created by a Python script, as they were a prerequisite for using DESeq2. DEGs identified via DESeq2 were functionally annotated. Finally, experimental methods for validation of bioinformatically generated results were discussed.

The following protocol is separated into chapters, which are subdivided in a theoretical and practical part.

3 Generation of reads

3.1 Theory

- Types of eukaryotic DNA (g/mt/cp/p DNA)
- Critical steps of DNA isolation for a follow up efficient sequencing
- DNA isolation techniques for plant DNA and plasmid DNA
- Quality controls of isolated DNA
- Sequencing technologies (First, second, third Generation)
    - Especially Illumina sequencing, Sanger, and Roche 454 sequencing methodology
        - Illumina
            - Multiplexing
            - Types: single-end, paired-end, multiplex
            - Read length from 32nt to 300nt possible
- File formats of sequencing results
    - TRACE, FASTA/QUALA, FASTQ, SAM/BAM, HTML
- Process of sequencing data including
    - Quality controls
    - Trimming
    - Backup and uploading of the results (Sequence Read Archive)
- FastQC (fast quality control)
    - The different properties of the sequencing data were examined and the results of the individual categories are evaluated using a colour code (good = green, ok = orange, bad = red)

3.2 Practical application

For read generation the sequence length distribution of TAIR10.fa (Col-0 reference sequence) was analysed. This was an already reduced dataset to facilitate the work process and computing times. To practice how to display the help of different programmes, the help of SOAPdenovo2 and Trimmomatic were invoked.

Example command to call the help of FastQC:

$ fastqc-help

Now FastQC is used to check the quality of the sequencing data.

Example command to start FastQC on our data:

$ fastqc Nd1_fw.fastq Nd1_rv.fastq Nd1_fw.fastq Nd1_rv.fastq

Output: Nd1_rv_fastqc.zip, Nd1_rv_fastqc.html, Nd1_fw_fastqc.zip, Nd1_fw_fastqc.html

Various properties were analysed via FastQC:

- **Basic Statistics**
    - Provides general information:
        - File name and type
        - Encoding
            - Decides how the quality score is calculated
                - ASCII Code represent the quality of each base and by means of Illumina 1.9, which works with phred 33, the phred score of each base can be calculated.
                - The ASCII code “”stands for 34, then 33 (because of phred 33) would be deducted to get a phred score of 1 for the base.
            - Fastq determines which version is available (phred 33 or 64)
                - Total number of sequences analysed
            - Number of low-quality sequences
                - These are removed and not heeded during analysis
            - Sequence length

- gives either the value of the longest and shortest sequence, or only one if all reads are the same length
- GC content in percentage of all bases in all sequences

  - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fastq

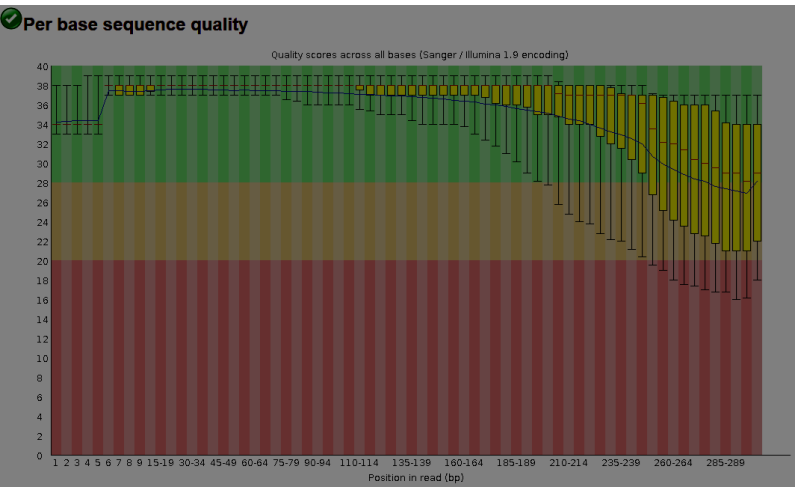
Basic Statistics

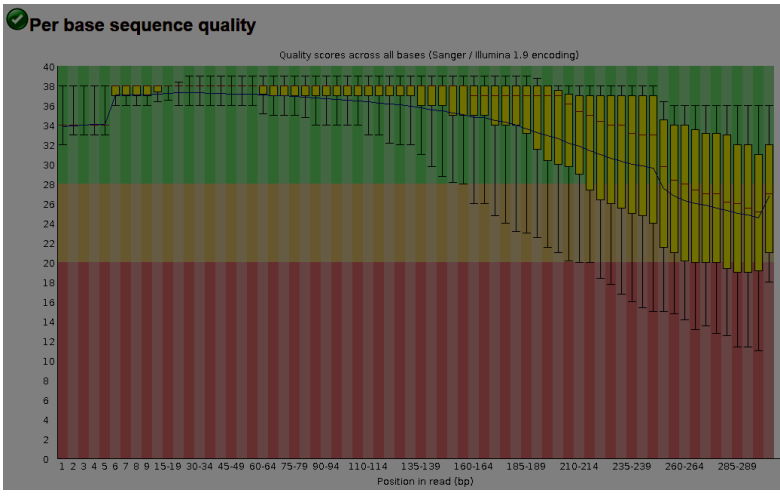
Measure	Value
Filename	Nd1_fw.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	233092
Sequences flagged as poor quality	0
Sequence length	22-300
%GC	35

Basic Statistics

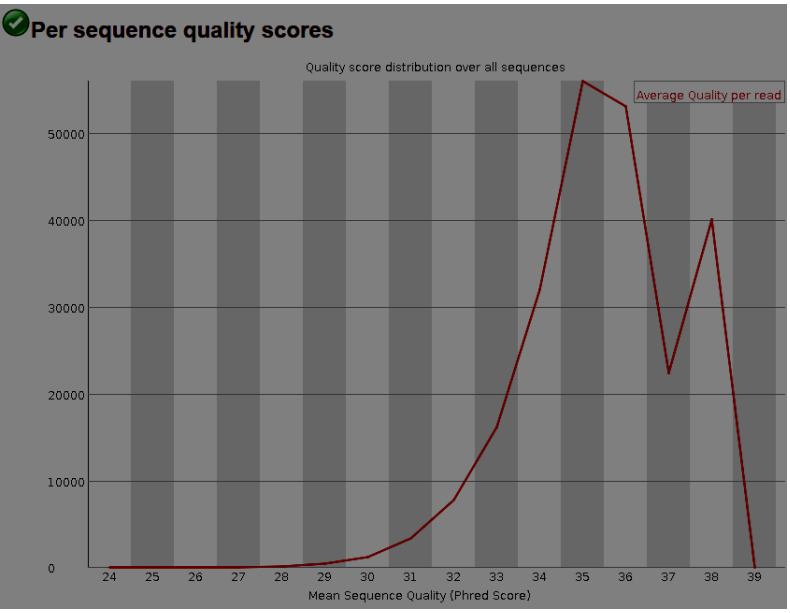
Measure	Value
Filename	Nd1_rv.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	233092
Sequences flagged as poor quality	0
Sequence length	22-300
%GC	35

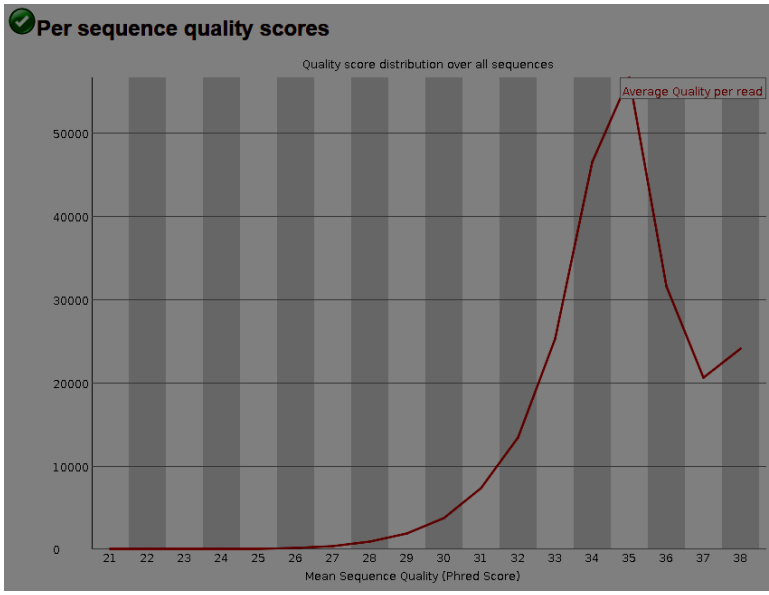
- **Per base sequence quality**
  - Displays the position-specific respective quality of each base
  - The quality is indicated by the phred score. The maximum value is 40 and means that one base out of 10,000 is “wrong” (e.g. sequencing errors) or in other words 99.99% accuracy
  - A low quality is typical at the beginning and end of the reads, because of e.g. artifacts caused by the Illumina software
    - The sequences can still be good
  - The average phred score per position of our data varied between 16-38 for the fw _ reads and 12-38 for the rv _ reads and showed the typical slump at the beginning and ending of the reads
  - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



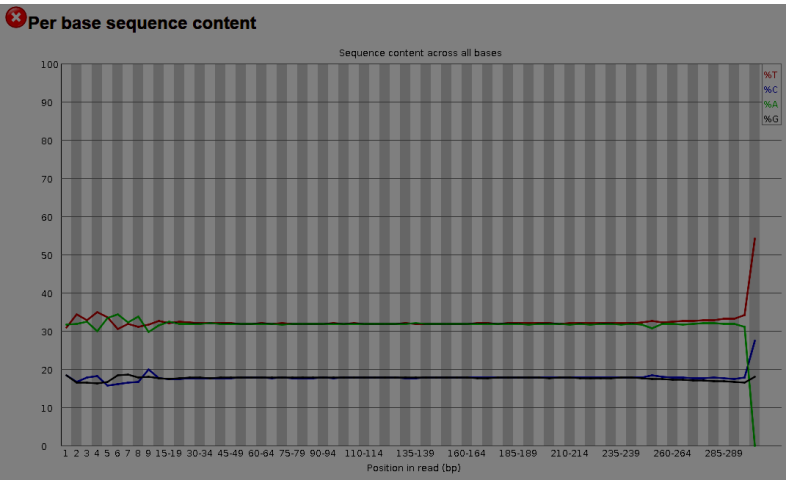


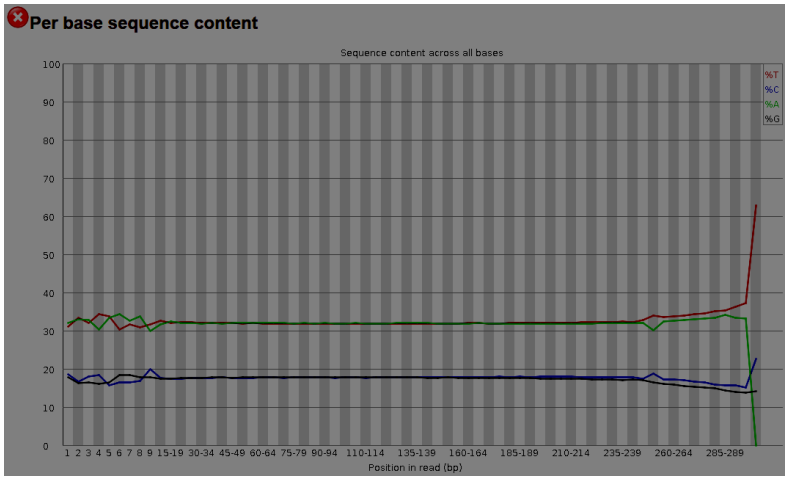
- **Per tile sequence quality**
  - A tile is a region of the flow cell. It is the cluster of a sequence and thus provides a read.
  - By means of tile sequence quality you can determine the quality per tile and if necessary remove sequences with low quality.
    - Low quality sequences can derive from e.g. a dust particle on the flow cell, which disturbs the fluorescence signal.
- **Per sequence quality score**
  - Shows the average sequence quality
    - In general, sequences with a minimum phred score of 20 are used for further analyses. Every sequence below this threshold is removed.
  - For one data set you would expect in general one peak
    - We observed two peaks for the fw_reads (Phred score of 35 and 38) and one peak for the rv_reads (average phred score at 35). The two peaks of the fw_reads can be explained through the two data sets with different qualities.
    - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



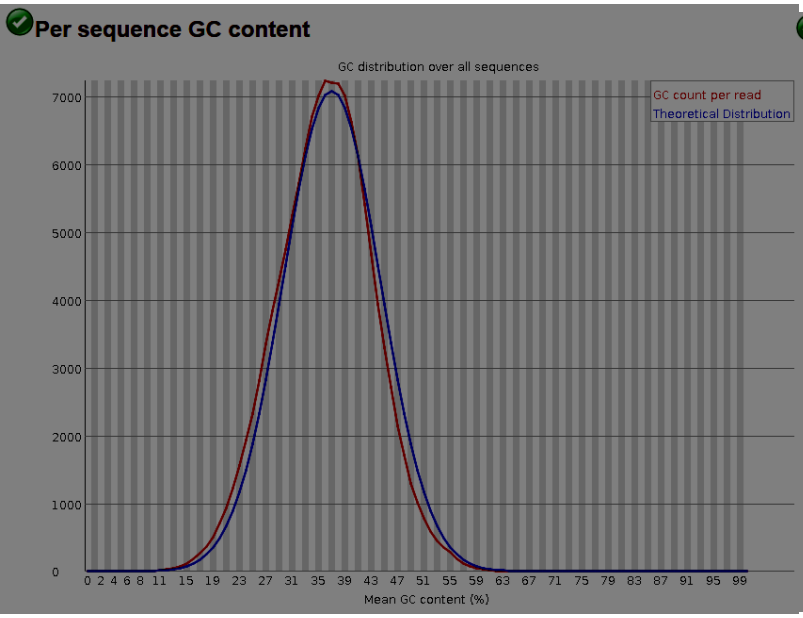


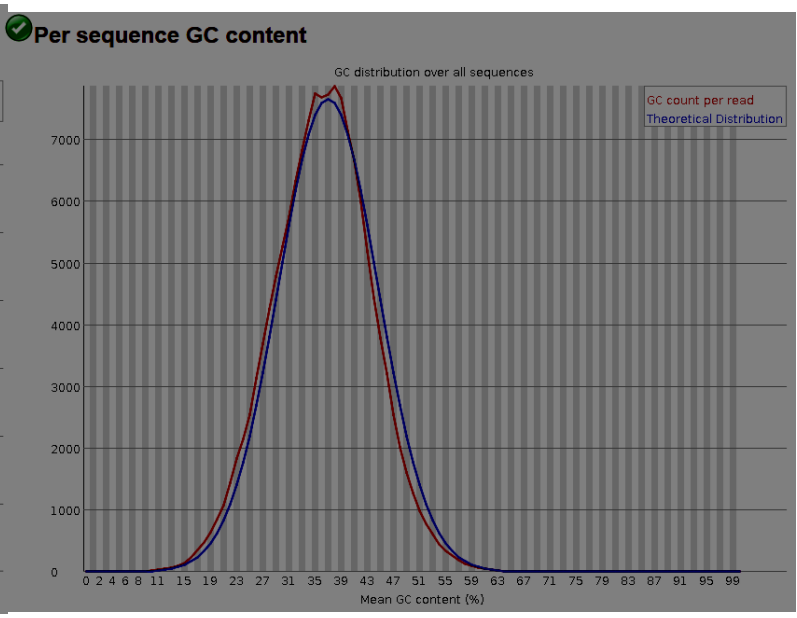
- **Per base sequence content**
  - Displays percentage of guanine, adenine, cytosine, and thymine at each position for all bases of all reads
    - Values of complementary bases, should match (A = T, G = C)
    - Differences in the sequence ends are caused by the fragmentation of DNA
    - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



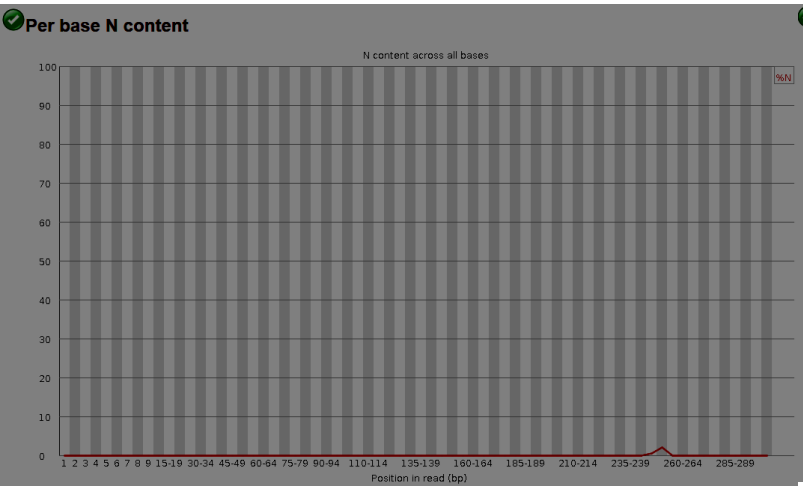


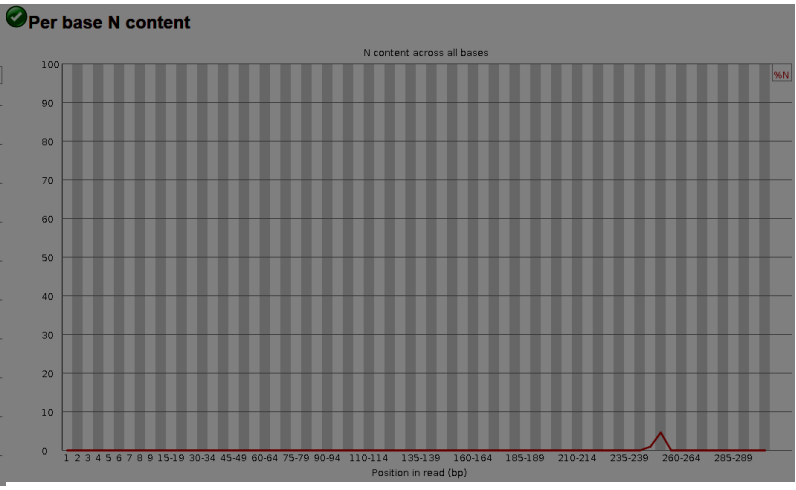
- **Per sequence GC content**
  - Average GC content [%] of all reads and theoretical distribution of GC content [%] are diplayed
  - Because of the trimming, we got two peaks and artificial regions are removed
  - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



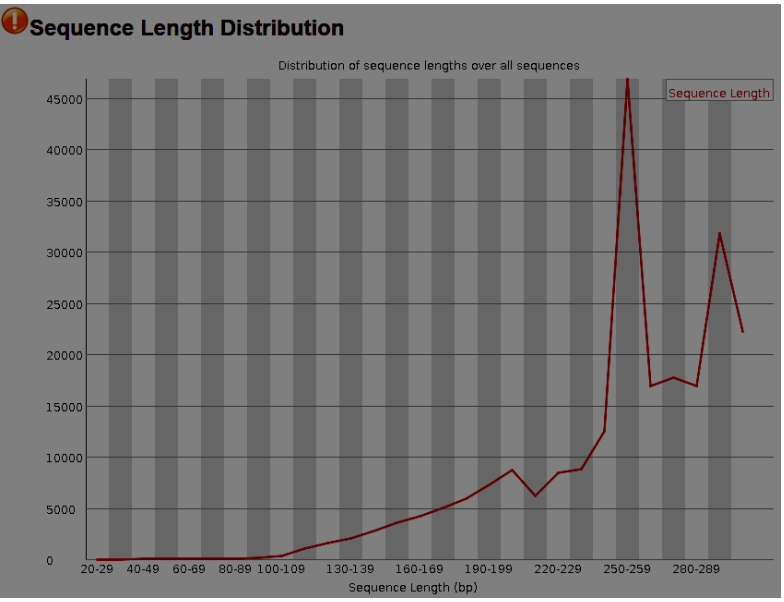


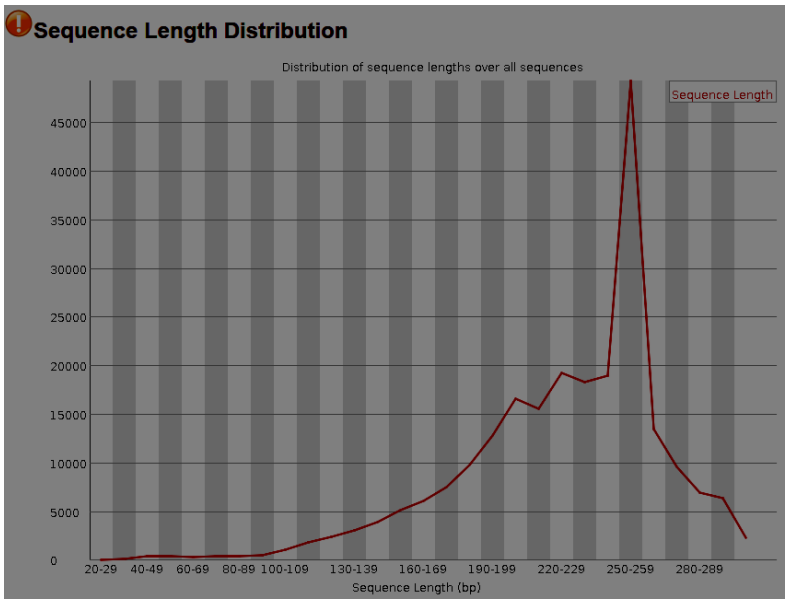
- **Per base N content**
  - Position-specific and percentage of the Ns (Symbol for sequencing error) in the reads
  - Ns should be removed by trimming
  - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq





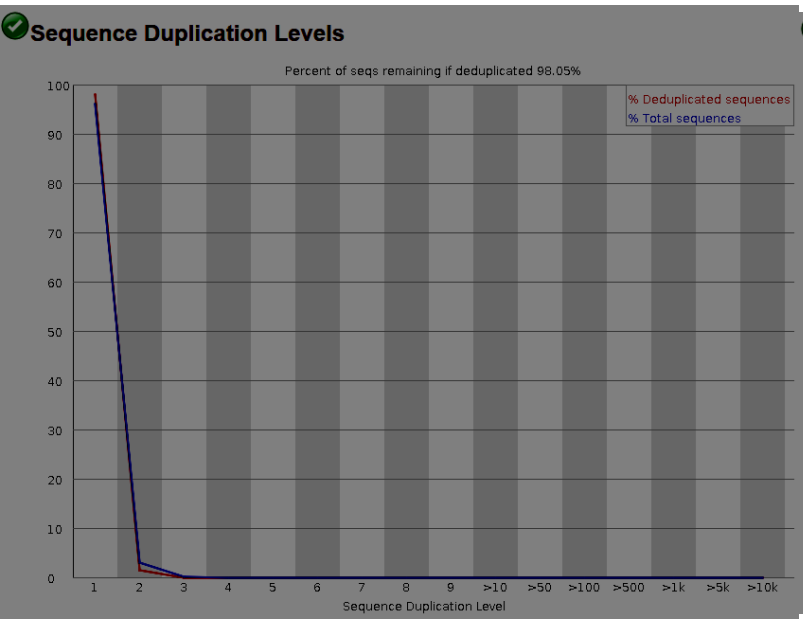
- **Sequence Length Distribution**
  - Distribution of sequence length across all sequences
    - Before trimming, all Illumina reads should be the same length (e.g. 300 bp)
    - We received two peaks at the fw _ reads (1.250 bp, 2.300 bp). This is because we used two data sets. A peak turned out for the rv _ reads (1.250 bp) However, we get an orange, i.e. mean rating from FastQC here, as the program does not expect such heterogeneity (usually all reads are the same length).
    - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq

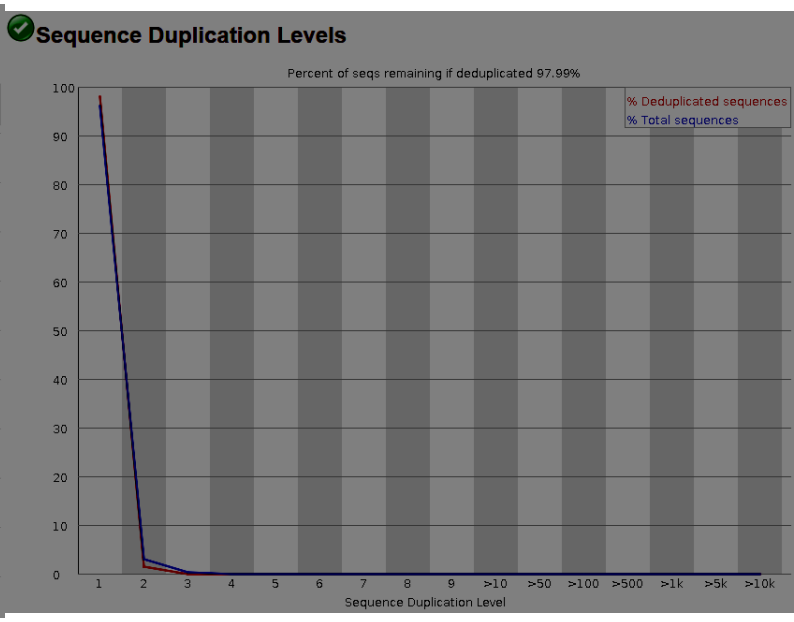




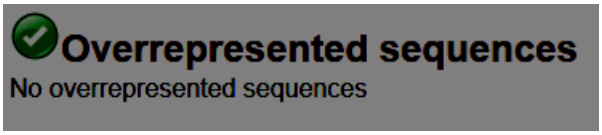
- **Sequence Duplication Levels**
  - Duplicated sequences are sequences that start and end at the same point. They have a 5 ' part and a 3 ' part, which are related (homologously). For example, an area is covered by one sequence several times.

        - Technical reasons for duplicated sequences: Such sequences are generated by PCR amplification of DNA fragments before sequencing. DNA polymerase can replicate certain sequences better than others and therefore produces duplications (preferably GC rich regions).
    - Provides information on whether some fragments have been sequenced more frequently than others. This is the case when multiple clusters form from one sequence by accident.
    - Not relevant to the assembly because only one version of the sequence is used
    - However, it is important for read mapping, as it can explain "false" variations
    - As a rule, sequences occur only once. Also possible twice if e.g. two exactly the same fragments were sequenced by chance (NOR = nucleolus organising region).
    - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



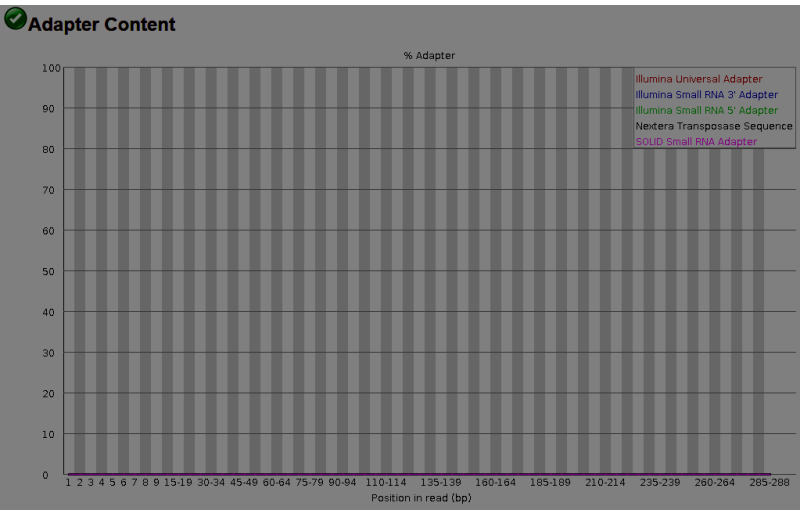


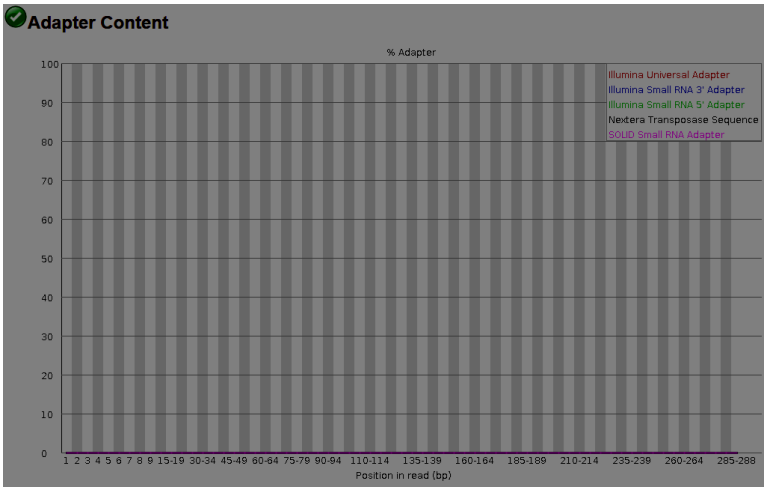
- **Overexpressed sequences**
    - Can prefigure remaining adapter sequences
    - Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



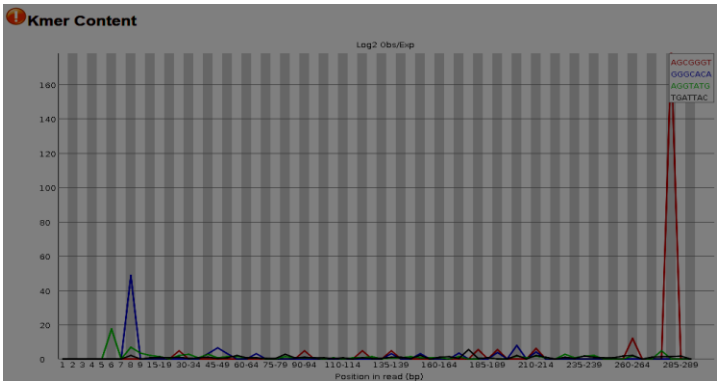
- **Adapter content**
- Position-specific content of adapter sequences [%]

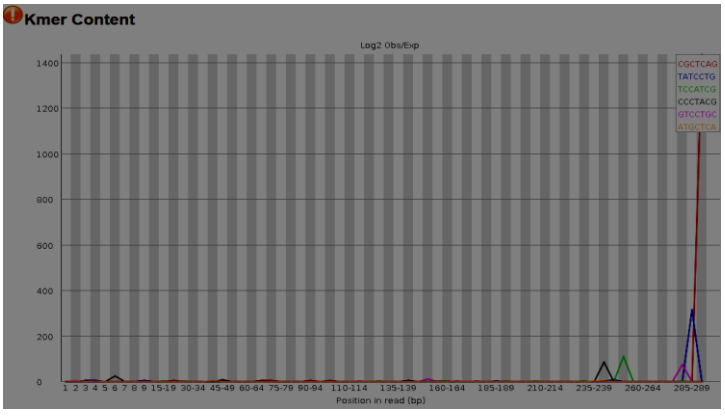
- Differentiation between universal, small RNA adapters possible, as well as transposase sequences
- Adapter sequences are trimmed before assembly. This prevents the reads from linking them incorrectly to contigs.
- No adapter sequences have been found in our data, as the Illumina software removes them and the already done trimming with trimmomatic does this as well
- Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq



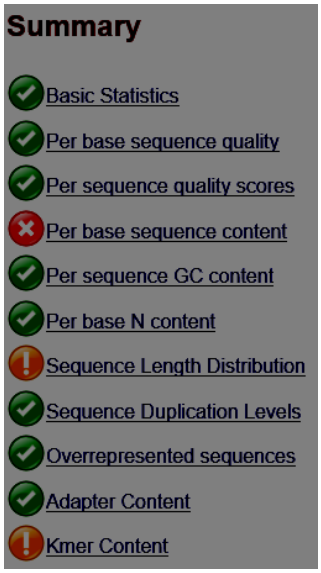


- **Kmer content**
- Peaks display overrepresented kmere position-specific
- It also shows how common the kmer is and how high its absolute overrepresentation is, as well as the corresponding p value
- The overrepresented kmer can also indicate adapter sequences, e.g. when the kmer occurs more strongly at the beginning of the sequence
- Example: 1) Nd1_fw.fastq 2) Nd1_rv.fatsq





- Summary was the same for each data set:



After the data was analyzed with FastQC, the data was trimmed with trimmomatic. Trimmomatic is a tool for flexibly trimming of Illumina NGS data. If the data had already been found to be good after FastQC (all green), it would not necessarily have been necessary to trim again. After trimming the data, the quality of the data could be checked again with FastQC. For trimming with trimmomatic, the following properties were chosen and these steps were performed: Features of flexible trimming and the structure of the command to run trimmomatic:

- **java-jar trimmomatic 0.35.jar**
  - Program call, for which you may need to specify the absolute path of the directory in which the program is located
- **PE or SE**
  - Trimming mode: paired or single end
  - For PE you hand over two input files and get four output files
  - A credit input file is handed over for SE and an output file is obtained
- **phred33 or-phred64**
  - Passes phred score with the should be trimmed
  - According to the SLIDINGWINDOW command: For example, it takes three bases and calculates the average score. When the result is below a certain cutoff (can be selected) the sequence is removed.
- **input_fw.fq** (output files to be trimmed)
- **input_fw.fq** (output files to be trimmed)

- **out_fw.paired.fq** (created after trimming)
- **out_fw.unpaired.fq** (created after trimming)
- **out_rv.paired.fq** (created after trimming)
- **out_rv.unpaired.fq** (created after trimming)
    - The order is important: The first output file created contains the fw paired reads, independent of the output file, therefore the file should be named more sensibly.
- **ILLUMINACLIP**
    - Passes fasta file with adapter sequences and indicates how they should be cut off, e.g. **Illumina_adapters.v4.fa:2:30:10**
    - 2: Number of mismatches that are still tolerated to find the index
    - 30: Larger areas (up to 30 bp), duplicated indexes, are also found
    - 10: Length of alignment in nt. In this case, the alignment should overlap at least 10 bp, then the index is removed.
    - The indexes are specific sequences in order to be able to uniquely identify each read. They are also used in multiplexing.
- **LEADING**
    - Indicates how many bases to cut off at the beginning (default is 3)
- **TRAILING**
    - Indicates how many bases you want to cut off at the end (default is 3)
- **SLIDINGWINDOW (e.g. SLIDINGWINDOW:4:15:10)**
    - 4:15:10: Sequence is analyzed in four bp steps: Should the average quality be less than 15 (phred score), the sequence will be trimmed. A low phred score is expected at the ends. 10: Entry is no longer necessary. In the past, this may have asked for the possibility that in the end 10 bp were basically cut off.
- **MINLEN**
    - The minimum length of the reads that should still be given after trimming, in our case no read may be shorter than 150 bp.
- **TOPHRED33 or TOPHRED64**
    - Converts the read quality to a phred33 or phred64 scale (We did not used it)

Generally speaking, inputs are executed in the order you enter them. So first the program detects whether it's PE or SE, then the adapter follows clipping and so on. Always include file format, e.g. add fq. It is important that the files are called appropriatly.

Example command to analyze files with trimmomatics:

$ java -jar trimmomatic-0.35.jar PE -phred33 **Nd1_fw.fastq Nd1_rv.fastq nddata_fwpaired.fq nddata_fwunpaired.fq nddata_rvpaired.fq nddata_rvunpaired.fq ILLUMINACLIP**:**Illumina_adapters.v4.fa**:2:30:10 **LEADING**:0 **TRAILING**:5 **SLIDINGWINDOW**:4:15:10 **MINLEN**:150

Output in the console shows how many percent of the original data has been preserved: Normally, 95-90% of the data is preserved (at Illumina).
Due to the large storage capacity required for Fastq files, the files obtained in this way should be compressed by gzip. This makes efficient data storage possible. NGS data is basically compressed in such a way since many tools support the compressed file format (.gz).

Example command to compress files with gzip:

$ gzip/vol/agrcourse/members/hschilbert/nddata_fwpaired.fq
$ gzip/vol/agrcourse/members/hschilbert/nddata_fwunpaired.fq
$ gzip/vol/agrcourse/members/hschilbert/nddata_rvpaired.fq
$ gzip/vol/agrcourse/members/hschilbert/nddata_rvunpaired.fq

Example command to decompress files with gunzip:

$ gunzip/vol/agrcourse/members/hschilbert/nddata_fwpaired.gz

Course tasks/questions:

1) How does compressed and uncompressed file differ in size?

Table 3: Filesizes of uncompressed and compressed files.

File	Uncompressed	Compressed	Difference
fwpaired	116,4 MB	36,7 MB	79,7 MB
fwunpaired	7,8 MB	2,7 MB	5,1 MB
rvpaired	106,3 MB	36 MB	70,3 MB
rvunpaired	4,3 MB	1,5 MB	2,8

2) How can you extract the files again?

By means of gunzip FILENAME (see above)

3) How can you make your data accessable fort the comunnity?

By uploading the data to NCBI's Sequence Read Archive (SRA). Assets an entry with meta information and then submit the corresponding files.

In addition, you could load the files in Filezilla and make them "locally." Filezilla is also used to transfer the data to SRA.

With the help of md5sum, the data are checked for its completeness as well as reliability of the content. The execution of the md5sum command gives the file this number. If the specific sum of the file changes, it has been modified (possibly by third parties!).

Example command to determine the md5sum of the compressed files:

$ md5sum nddata_fwpaired.fq.gz ce274ad85651de531b056060e5ad9535 nddata_fwpaired.fq.gz
$ md5sum nddata_rvpaired.fq.gz 0a1d09ba13cc275fd715ff377c2667b0 nddata_rvpaired.fq.gz

Course tasks/questions:

1) Name the md5sums of the trimmed paired reads fastq files.

For nddata_fwpaired.fq.gz => ce274ad85651de531b056060e5ad9535
For nddata_rvpaired.fq.gz => 0a1d09ba13cc275fd715ff377c2667b0

Then fastq dump is used to download data from the SRA. Many large temporary files are generated. Therefore, a directory with sufficient storage capacity was selected. By specifying an SRR_ID (SRR3390908), a specific file/entry can be downloaded. The paired-end reads are stored in two files, compressed into gzip and stored in the directory that was previously specified. The SRA data can be

transferred with Filezilla.

Example command to download data from the SRA:

$ fastq-dump--split-files--gzip SRR3390908

Output files are stored in the desired dictionary and were called e.g. SRR3340908_1.fastq.gz and SRR3340908_2.fastq.gz

A paired-end read is usually stored in a file (fw and rv hang on to each other). By split, the paired reads are then divided into fw and rv reads files. Therefore, you will get two output files, one with fw one with rv. This file is numbered so that it is still possible to assign the pairs (e.g. SRRxxx 0001/1 fw or 0001/2 rv).

Course tasks/questions:

1) What information can you get about the ID SRR3340908?

Publication date, filesize, sequence type and production of data, fw and rv read length

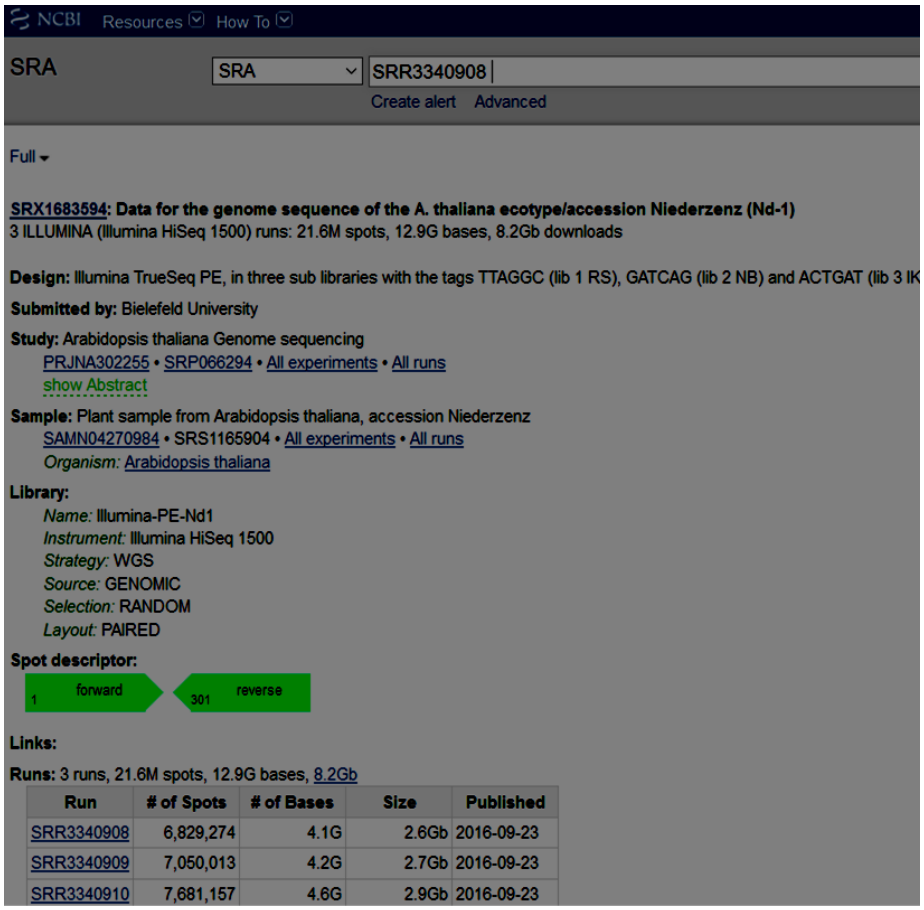


Figure 1: Structure of a SRA entry.

Diplayed is the structure of a SRA-entry for the ID SRR3340908.

2) How can you get more information?

With the button related information, e.g. BioSample or BioProjects. You can see then which and how many data sets were used.

3) Are there more entries, which belong tot he same project?

Yes. You can find them here: *Arabisdopsis thaliana* Genome sequencing
https://www.ncbi.nlm.nih.gov/bioproject/PRJNA302255
https://www.ncbi.nlm.nih.gov/sra?linkname=bioproject_sra_all&from_uid=302255
There are more data sets which belong to the same genome. To generate a genome sequence 8 projects (“SRA Experiments“) were used.

4 Assembly und Scaffolding

4.1 Theory

- **Assembly**
  - Generating overlapping reads to contigs with the goal of producing a consensus sequence/full genome sequence (or reducing the number of contigs as much as possible)
    - Contigs can be connected to scaffolds
      - Scaffolds have information about the distance between the two connected contigs. This is filled with Ns (= gap), i.e. one does not know the sequence between the two contigs.
      - The formation of scaffolds usually takes place with paired-end (or mate pair) data (with single end not possible!)
  - Necessary because the read length is limited and the full genome cannot be sequenced in one run
  - Algorithms used to create the assembler:
    - For both methods, the reads are initially broken down into kmers.
    - OLP = overlap layout consensus
      - Uses small overlaps of the kmers and connects without contradictions
        - De Bruijn Graf
      - More efficient method: Sequence is located between the nodes and therefore easier to solve. Efficiency depends on read length and quality.
  - **Assembly problems**
    - Clarity of the related paired-end reads is not always given
    - For example for inversions, centromere regions, and repeats
    - When several reads are located before/behind repeat regions, which repeat themselves in the genome, no unique paired-end read pair can be generated. Repeats are recognizable by higher coverage.
    - Regions near the centromere are difficult to assemble. Here the assembly breaks, as no connections can be formed over the large centromere region. Therefore, many scaffolds are produced.
      - Regions with very low/no coverage possible
        - This is partly due to the fact that only one chromosome arm can be assembled, since usually one cannot assemble beyond the centromere

        - Sometimes specific mapping of reads is not possible, e.g. if the sequence occurs several times.
    - In the case of these problems, you can either change the settings of the assembler or break up the sequences (e.g. no adequate coverage is available)
    - Reads are mapped against a reference (genome)
- **Can use SOAPdenovo2 – An assembler for heterozygous organisms**
    - Generates reads from contigs
        - Different settings possible: < 63 bp kmers, < 127 bp kmers
    - Input files should not be compressed
    - Generates also scaffolds, but we used an extra programme for this (SSPACE)
    - Choice, which assembler to use and other settings (such as kmer size) are dependent on the individual genome
    - As a rule: Assemblies are the critical step and computatial expensive

- **Assembly evaluation**
- Different values shed light on the quality of an assembly: For example the N50
    - N50: The contigs determined are added up and the N50 is exactly the length of the contig, where 50% of the total assembly size is exceeded. The greater that value, the better. Contigs are sorted descending.
    - N50 is no longer meaningful at some point, because the length of the contigs is limited: A contig can not be longer than a chromosome arm (possible with pacbio sequencing, but not with Illumina!)
    - N25, N75 and N90 = > N50 is established
- Read mapping for full quantification
    - Percentage of mapped reads reflects the quality of the assembly
        - Because of sequencing errors and artifacts, 100% are never reached
    - Detects areas with very high-security low coverage
    - Rearrengements in the assembly can be identified by mapping paired-end and mate pair reads

- **BUSCO Analysis**
- Generally preserved genes ("BUSCO genes": Genes that occur in different species and are homologous, about 700 pieces) that occur in the assembly are counted
    - In a good assembly, you would expect to find a variety of these genes.

        - We did not do this, because it would not make sense since we could not find all the genes because we worked with a greatly reduced dataset
- **Scaffolding theory – how can you connect contigs without losing sequence?**
    - Increases the continuity of the assembly
    - Two contigs are connected to a scaffold
- **SSPACE – scaffolding program**
    - performs scaffolding with our data

4.2 Practical application

The paired-end reads, which have already been tested and processed, are connected to contigs (and scaffolds) using SOAPdenovo2.

The paired fw and rv files (nddata_fwpaired.fq.gz and nddata_rvpaired.fq.gz) are decompressed with gunzip (command see above) and then passed to SOAPdenovo2:

Before starting SOAPdenovo2, the example_config.txt file must be edited and our paired-end data paths for q1 (forward reads) and q2 (reverse reads) must be specified. The example_config.txt file contains the following assembly parameters (these can be edited depending on the assembly):

```
#maximal read length
max_rd_len=300
[LIB] (=library)
#average insert size (distance between paired reads)
avg_ins=750
#if sequence needs to be reversed
reverse_seq=0
#in which part(s) the reads are used
asm_flags=3
#use only first 300 bps of each read
rd_len_cutoff=300
#in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for short insert size)
pair_num_cutoff=5
#minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)
map_len=50
#a pair of fastq file, read 1 file should always be followed by read 2 file
```

q1= nddata_fwpaired.fq
q2= nddata_rvpaired.fq

- Insert size: Depending on the respective library, about 600-800 bp
- asm_flag: Indicates which data to use: At 3, the data is used for both the assembly and for scaffolding. (1 = scaffolding only, 2 = assembly only). Thus, this value indirectly shows whether you trust a library.
- rank: Similar to flags, if you have several data sets then the first thing you can do is work with the one you trust the most and then, for example, use a smaller dataset to fill in the gap. This one receives a subordinated rank.
- pair_num_cutoff: How many paired-end pairs must exist for two contigs to generate a scaffold out of them (should be at least 5).

Then SOAPdenovo2 was performed at 63 k or 127 k and the results of the two assemblies were compared: Will the assembly be better in one case?

Example command to use SOAPdenovo2 to generate contigs (and scaffolds) from reads:

For 63 k:
$ SOAPdenovo-63mer all -s **example_config.txt** -K 63 -R -p 4 -o **first_test_assembly** 2>**assembly.log** 1>**assembly.err**

For 127 k:
$ SOAPdenovo-127mer all -s **example_config.txt** -K 127 -R -p 4 -o **secondfirst_test** 2>**assembly127.log** 1>**assembly127.err**

- -s: CONFIG_FILE
- -K: 63 or 127 (kmer size)
- -R: removes repeats, thus improves the assembly
- -p: Number of CPUs
- -o: output file path and name
- 1>: error log file (was empty in our case)
- 2>: output log file, documentary file
- Error and output files are used because the results would be otherwise printed in the console => unclear

After the assemblies have been created their quality is checked using, among other things, the N50 value with the help of a python script contig_stats.py:

Usual construction of a python command: Enter python and then the name of the python script followed by the input file. The output is created in the same directory as that of the input file.

Example command to use the python script contig_stats.py to determine the quality of the assembler:

For 63 k:
$ python contig_stats.py --input **first_test_assembly.contig**

For 127 k:
$ python contig_stats.py --input **secondfirst_test.contig**

The input file: first_test_assembly.contig, contains the assembled contigs
Name of the assembly for 63 k: first_test_assembly_trimmed
Name of the assembly for 127 k: secondfirst_test.trimmed

Output Statistic-file of the 63 k Assembly: **first_test_assembly_trimmed_stats.txt**

Output Statistic-file of the 127 k assembly: **secondfirst_test_trimmed_stats.txt**

By means of the stats.txt-files the assemblies were compared:

Table 4: Characteristics of the 63k and 127k assemblies.

The following table lists the contig and scaffold number, the average contig- and scaffold size, the size of the smallest and largest contig or scaffold, the total number of base pairs of the assembly with and without Ns, the GC content in percentage, as well as the N25-/N50-/N75-, and N90 value of the 63 kmer and the 127 kmer-assembly after analysis with SOAPdenovo2.

Characteristic	**63 k assembly**	**127 k assembly**
number of contigs	**679**	**223**
average contig length	4268	13263
minimal contig length	500	591
maximal contig length	24865	109064
total number of bases	2898318	2957821
total number of bases without Ns	2898318	2957821
GC content [%]	**36**	**36**
N25	11436	47421
N50	**6792**	**24972**
N75	3750	12948
N90	2031	6753

The comparison of the 63 kmer and 127 kmer size shows that the 127 kmer approach results in a significantly better assembly (Table 4). This can be seen in both the lower contig number and the higher N50 value. Presumably, the assembly would get even better if you increased the kmer size even more. However, this method only improves the assembly until the optimum is reached. The kmer size could be increased as sequencing technologies get better (pacbio) or reads get longer. The kmer size should be varied at a distance of 30 and the resulting assemblies should be compared. In the rules, the half to toe 3 of the read length is the optimal kmer size.
In addition, the GC content of the assemblies is interesting, as it can be compared with the GC content of the reference (e.g. *A. thaliana* has 36%) and can be checked whether it meets expectations.
If the GC content would contradict the expectations, artifacts or bacterial sequences may have entered the assembly. To identify such DNA, all generated contigs could be blasted. The contigs

you do not get hits for would be investigated more closely with blastn to determine if they are new genes. This could be the case if no fungi or bacteria etc. hits would occur. Otherwise, this step would identify exactly this foreign DNA.

Now scaffolding is carried out using SSPACE for the 63 kmer and 127 kmer data set, e.g. with the assembled contigs. The sspace_librarie.txt file contains the setting information for SSPACE. Below is the content of this file (we had to modify this file with our data path):

Example command to use bowtie to determine if all trimmed reads will find their pairs again:

$ run5_PE **bowtie nddata_fwpaired.fq nddata_rvpaired.fq** 750 0.2 FR

- 750: average read size
- 0.2: deviation from the read size
- FR: forward revers

Bowtie is a tool to map reads and is used from SSPACE.

With this parameters, SSPACE can be run in the follow-up:

Example command to perform scaffolding of the 63 kmer dataset using SSPACE:

$ SSPACE_Standard_v3.0.pl -l **sspace_libraries.txt** -s **first_test_assembly_trimmed.fasta** -k 3 -T 4 -b **sspace_results**

- -l: TEXTFILE, which contains information about the reads
- -s: ASSEMBLY, SOAP assembly result file (_trimmed.fasta)
- -k: NUMBER_OF_LINKS, how many connectionfragments must exist between two contigs to combine them to a scaffold.
- -T: Number of CPUs used
- -b: BASE_NAME, name of the output file, will be saved in a new folder

SSPACE works with the trimmed.fasta files. In this case, it only uses the ends of the sequence to determine the overlap and is therefore faster finished with calculating when trimmed data is passed.
After scaffolding has taken place, the sequence length distribution, or quality of the assembly (N50 etc.) is analyzed again with the python script contig_stats.py and the results are compared with the SOAP contigs, as well as the SOAP scaffold results.

Example command to use the python script contig_stats.py to determine the quality of the assembler after scaffolding was performed using SSPACE:

$ python contig_stats.py --input **sspace_results.final.scaffolds.fasta**

To compare the results, the 127 kmer data set of the SOAPdenovo assembler is also analyzed using SSPACE etc. (Repetition of the above commands):

Example command to perform scaffolding of the 127 kmer dataset using SSPACE:

$ SSPACE_Standard_v3.0.pl -l **sspace_libraries.txt** -s **secondfirst_test_trimmed.fasta** -k 3 -T 4 -b **sspace_results2**

Example command to use the python script contig_stats.py to determine the quality of the assembler:

$ python contig_stats.py --input **/sspace_results2.final.scaffolds.fasta**

The results of the various assembly quality analyses are then compared to evaluate the result of scaffolding (Did scaffolding improve the assembly?). Among other things, the average contig size was compared with the average scaffold size (Before scaffolding with SSPACE: 63 k Assembly are the SOAPdenovo results):

Table 5: Scaffold evaluation and results of various assemblyquality analyses.

The table lists the contig- and scaffoldnumber, respectively. Scaffold count, the average contig-and scaffold size, the size of the smallest or largest contig or scaffold, the total number of base pairs of the assembly with and without Ns, the GC content in percentage, as well as the N25-/N50-/N75-, and N90 value of the 63 kmer- and 127 kmer-assembly before and after scaffolding with SSPACE.

Characteristics	**Before scaffolding via SSPACE: 63 k assembly**	**After scaffolding via SSPACE: 63 k assembly**	**Before scaffolding via SSPACE: 127 k assembly**	**After scaffolding via SSPACE: 127 k assembly**
Number of contigs bzw. scaffolds	**679**	**659**	**223**	**200**
Average contig length bzw. scaffolds	4268	13263	13263	14799
Minimal contig length bzw. scaffolds	500	591	591	591

Maximal contig length bzw. scaffolds	24865	109064	109064	109064
Total number of bases	2898318	2957821	2957821	**2959986**
Total number of bases without Ns	2898318	2957821	2957821	2957821
GC content [%]	36	36	36	36
N25	11436	47421	47421	47421
N50	**6792**	**24972**	**24972**	**27583**
N75	3750	12948	12948	14453
N90	2031	6753	6753	7301

63 k before scaffolding:
Assembly name: first_test_assembly_trimmed_stats.txt
63 k after scaffolding:
Assembly name: sspace_results.final.scaffolds_trimmed

127 k before scaffolding:
Assembly name: secondfirst_test_trimmed
127 k after scaffolding:
Assembly name: sspace_results2.final.scaffolds_trimmed

The comparison shows that the best result could be achieved with the 127 kmer approach and subsequent scaffolding using SSPACE (Table 5). The number of contigs was reduced to 200 and therefore 459 more contigs could be connected to scaffolds compared to the 63 kmer approach. This conclusion is supported by the values of the N50, which is also significantly higher in the 127 kmer approach (according to scaffolding, 27583) at compared to the N50 of the 63 kmer approach (after scaffolding, 24972).
With the 63 kmer approach, 20 contigs could be assigned to scaffolds via scaffolding. With the 127 kmer approach, 23 contigs could be assigned to scaffolds via scaffolding.
In addition, the large difference of the N50 values is striking: It is 6792 (before scaffolding) for the 63 kmer approach compared to 24972 (before scaffolding) for the 127kmer approach. The difference is 18.180. This is because the 127 kmer approach can use larger pieces of the reads and thus find more easily and quickly overlaps compared to the 63 kmer approach, which consists of smaller read pieces (just 63 bp pieces). In addition, this large difference shows that the 63 kmer approach is not a good setting, because it breaks off quickly before scaffolding and therefore only reaches a small N50 value.
On the other hand, the 127 kmer approach allows to connect contigs immediately to long contigs that are similar in size to later result in scaffolds. Therefore, there is no large difference in N50

value between the two 127 kmer approaches (contigs and according to scaffolding).
In addition, the confidentiality of scaffolding can be verified by means of read mapping. Rearrangements can be identified.

In summary, for our data, the 127 kmer approach or assembly is of a better quality than that with the 63 kmer approach (Table 5). The quality could be improved even further if you choose the kmer size even larger.
However, it does not apply “the greater the better”, since the increased probability of sequencing errors within a larger kmer must be observed. The kmer caused by sequencing errors should be discarded.
It also does not apply “the smaller the better” because very small kmers repeats can not be dissolved well or not at all. With larger kmers, this is conditionally possible, so fewer scaffolds (200) are produced.
It is valid: The lower the scaffold count and the larger the average scaffold size compared to the average contig size (before scaffolding) the better.

Our data for the 127 kmer approach after scaffolding with SSPACE provide a 3 Mbp assembly. The average scaffold size is 0.15 Mbp (3 Mbp (size of total assembler, rounded 2959986 bp = 3 Mbp, see Table 5). This is calculated as follows:
3 Mbp/200 (number of scaffolds) = 0.15 Mbp

To get an impression of other scaffold sizes and assemblies, we searched for reference data:

Table 6: Average scaffold size.
The table lists the average scaffold size in Mbp with the corresponding species.

Species	**Average scaffold size**
Arabidopsis thaliana Nd-1	0,15 Mbp (27x coverage)
Solanum lycoperscum (Tomate)	0,2-0,3 Mbp
Gossypium arboretum	140kb-5.9 Mbp
Brassica rapa	0,35 Mbp (72x coverage)

Due to the fact that we only used data from two data sets and because of the low coverage (about 27x) our value is rather poor, although we did not get a region near the centromere (pre-selected data!).
It is valid, if the quality of the data is good, the more data the better and the more contigs can be generated and the better the quality of the assembler.

In addition, the coverage can be calculated as follows:
fw 40 Mbp, rv 40 Mbp

A total of 80 Mbp data: From this we want to generate a sequence with 3 Mbp (size of the entire assembly) = > 80 Mbp/3 Mbpulate 27x (coverage)

In general, a 80x coverage would be desirable and usually results in a "good" assembly.

5 Annotation

5.1 Theory

- **Annotation**
  - In genetics and bioinformatics, annotation refers to a functional mapping that can come from both experimental findings and computer-assisted prediction. Thus, the annotation of a DNA sequence/gene describes among other things the exact location of exons and introns, protein-coding areas, promoter elements, repetitive DNA elements (structural annotation) including the encoded protein (functional annotation).
  - In order to annotate, it must first be clarified: "What is a gene?":
  - **Prokaryotic gene term**
    - Sequence starting at the start and ends at the stop codon (without gaps)
  - **Eukaryotic gene concept**
    - Eukaryotic gene structure
      - Upstream Regulatory Regions
      - Promoter (CAAT, GC, TATA box, transcription start point (+ 1))
      - 5 ' UTR of mRNA
      - Starting codon of protein biosynthesis (AUG)
      - Exon intron structure
      - Stop codon of protein biosynthesis
      - 3 ' UTR of mRNA
        - It contains polyadenylation signal and 3 ' trimming signal, as well as the corresponding 3 ' end of mRNA
      - Termination site of transcription
    - Transcribed region: Region between transcription start point (+ 1) to the termination site of transcription
    - Coding region: Region from the first base of the corresponding starting codon of protein biosynthesis (AUG) to the corresponding stopping codon of protein biosynthesis
    - 3 ' UTR and 5 ' UTR are difficult to identify (as opposed to the start and stop codon). There are no clear signals (such as a codon).
- The promoter is therefore ignored in gene predictions
- **Gene prediction – process and problems**
  - It is carried out on the basis of a sequence using HMM (Hidden Markov Model)

        - Takes a base as a start (randomly) and then probabilities are calculated for wether the next base is an A of the ATG/the starting codon. As long as this is not the case, you are in the 5 ' UTR.
        - After that, exons and introns are identified, among other things by using the donor and receptor splicesite (GT ... A (doesn't always have to be an A) ... AG) identified
            - Problems detecting exon and intron structure:
                - Relative frequency of intron boundary sequences used to detect Exon and Intron structure:
                - 999% GT AG, 0.8% GC AG, 0.2% AT AG
                - The length of the sequence of only two bases per boundary increases the probability of hitting in the sequence enormously, making identification more difficult
                - In addition, non canonical splice sites are often ignored and not found by tools (Augustus) due to their low frequency
                    - AUGUSTUS does not know the distance between an exon and intron exactly, distance must be at least 50 (because of branch site), to ensure that a formation of a lasso structure is possible
        - Finally, the stop codon is determined and then the end of the sequence
        - There are always exceptions, even from the gene definition
            - E.g. there may be exons that are in the 3 ' UTR but are not protein-coding
        - To meet the conditions for gene definition, it searches for many structures that are present in the definition
            - But because of the above problems, the same applies: **The gene prediction is not perfect!**
- **Gene prediction – different modes**
    - *Ab initio*: Gene forecast, without advance information
    - Hint guided: Uses RNA-Seq data, among others, to improve gene prediction
        - RNA-Seq data can provide information on where the transcription sequence is located
    - Reference-based: Annotated sequences of a reference can be transferred to the new sequence to be annotated
    - We did not have Nd-1 RNA-Seq data, so we used *ab initio*
- **AUGUSTUS – A gene prediction program**
    - AUGUSTUS is collection of pearl scripts

- *Ab initio* gene prediction

- **TAIR and Araport11-*A. thaliana* databases**
  - Used for reference-based gene prediction

- **BLAST – Basic local alignment search tool**
  - Tool for sequence comparisons
  - The most cited tools of all (Altschul *et al.* 1990)
  - The comparison is based on random base matches, which are then extended
  - Web-based service possible, but also possible via a console
    - Command line is more efficient (e.g. for large data sets)

5.2 Practical application

After a "good" assembly of the Nd-1 reads has been generated, a structural and functional annotation is performed with AUGUSTUS and the python script map_annotation (or BLAST). These identify potential genes and functionally characterize them in subsequent steps through other programs. We started with an *ab initio* gene prediction using AUGUSTUS. Structural annotation was performed three times with three previously generated different Nd-1 datasets (fast format needed): secondfirst_test.contig (determined with SOAPdenovo), secondfirst_test.scafSeq (determined with SOAPdenovo), sspace_results2.final.scaffolds.fasta (determined with SSPACE), i.e. with contigs and two different scaffold data sets. The associated trimmed data sets could also have been used to compare them with the untrimmed ones. It does nott make much difference whether the trimmed or untrimmed data was used because the contig-and scaffold number is the same and AUGUSTUS only needs a little more time for a gene prediction based on untrimmed data.

Example command to start gene prediction using AUGUSTUS:

$ augustus-3.2 --species=arabidopsis --gff3=on --codingseq=on **secondfirst_test.contig** > **result_contigs_SOAP_2**

$ augustus-3.2 --species=arabidopsis --gff3=on --codingseq=on **secondfirst_test.scafSeq** > **result_scaffolds_2_SOAP**

$ augustus-3.2 --species=arabidopsis --gff3=on --codingseq=on **sspace_results2.final.scaffolds.fasta** > **result_scaffolds_2**

Parameter set should be given:

- species: Select which species to take as a reference for gene prediction. This file also contains information about the average exon and intron size or gene structure of the organism (always different with other organisms!)
- gff3ularly on: output in gff3 format is generated = > table
- codingseq = on: Gives protein encoded sequence and the CDS (mRNA)
- the > prevents the results from being displayed in the console
- > stores the results in the path given behind it

- ! Attention! > also overrides the file if something is already in it!

Out of three different input files a corresponding output file (gff format) is generated. For example: **result_scaffolds_2** (GFF3 Format)

The file **result_scaffolds_2** (GFF3 format) contains details about all annotated features on each sequence sorted by the position of a gene on the reference sequence. Columns are: sequence name, source, feature type, feature start position, feature end position, quality, orientation (+ = forward strand, - = reverse strand), comments. Genes on the 200 scaffolds in **sspace_results2.final.scaffolds.fasta** were identified.

In addition to these lines with multiple columns, there are comment lines indicated by a leading '#'. They can contain the predicted CDS or peptide sequences. AUGUSTUS is generating a GFF3 file with these additions if specified during the gene prediction process. The Perl script getAnnoFasta.pl enables the extraction of these sequences into separate FASTA files. After downloading this script, it might be necessary to modify the permissions by checking the 'allow to execute' box.

Example command to change directory:

> **cd getAnnoFasta/**

Slash at the end oft he path is optional if only a direct subfolder is accessed.

Executing getAnnoFasta.pl to extract mRNA, peptide, CDS, and CDS exon sequences:

> getAnnoFasta.pl --seqfile=**sspace_results2.final.scaffolds.fasta result_scaffolds_2**

Output in terminal: Read in 200 sequence(s) from **sspace_results2.final.scaffolds.fasta.**

Executing this script generates four new files in the working directory:

result_scaffolds_2.mrna

result_scaffolds_2.codingseq

result_scaffolds_2.cdsexons

result_scaffolds_2.aa

AUGUSTUS names genes based on running numbers (g1, g2, g3, …). Alternative transcripts are indicated by increasing numbers after the '.t' extension e.g. g1.t1, g1.t2, g2.t1, g3.t1, … Just looking at the number in the last gene name in one of the FASTA files is an efficient way to determine the total number of predicted genes.

The CDS exons, the mRNA, and the peptide sequence were extracted:

CDS exon

>g776.t1.cds2

attggcatcaaagctgaatggagaggacaccatggtagacttgttctcttcatgggaaacaagaatacttctccacttacatcagttcaagctct
gatactacccccctgcccacttaagactggatctgtctccagtacctgatacaattcctcctcgagcac

mRNA

>g776.t1

cttatcggaaatattgctgagagatttcatgctttatgtttgaaagacagtggtgtcttatacgaggatcctcatatccagattggcatcaaagctg
aatggagaggacaccatggtagacttgttctcttcatgggaaacaagaatacttctccacttacatcagttcaagctctgatactacccccctgcc
cacttaagactggatctgtctccagtacctgatacaattcctcctcgagcac

encoded peptide sequence

>g776.t1

LIGNIAERFHALCLKDSGVLYEDPHIQIGIKAEWRGHHGRLVLFMGNKNTSPLTSVQALILPPAHLRLDLSPVPDTIP
PRA

All assemblies were subjected to *ab initio* gene prediction via AUGUSTUS and sequences were extracted via getAnnoFasta.pl. In total, 786 genes were predicted on SOAP contigs, 726 on SOAP scaffolds, and 776 genes on SSPACE scaffolds (`Table 7`).

Table 7: Assembly statistics and gene prediction results.

This table contains the number of predicted genes, the number of contigs/scaffolds, and the N50 of the assemblies. Different kmer sizes were compared.

	63mer SOAPdenovo2 assembly	**63mer SOAPdenovo2 assembly + SSPACE scaffolding**	**127mer SOAPdenovo2 assembly**	**127mer SOAPdenovo2 assembly + SSPACE scaffolding**
Annotated gene	**786**	**726**	**-**	**776**
Number of contigs / scaffolds	**679**	**659**	**223**	**200**

Average contig / scaffold length	4268	13263	13263	14799
N50	6792	24972	24972	27583

These numbers indicate that more and smaller contigs/scaffolds result in higher numbers of predicted genes. This could be explained by fragmentation of gene models over multiple contigs/scaffolds. Single exons of genes might be reported as separate genes (Table 7).

Following the ab initio gene previdiont via AUGUSTUS, CDS and peptide sequences were extracted from TAIR and Araport11. Mapping of *A. thaliana* Col-0 reference sequences (TAIR and Araport11) to the Nd-1 gene prediction allows the transfer of functional annotaitons. Comparison of TAIR10 and Araport11 properties:

- Araport11 provides unrestricted access
- TAIR10 limits the number of searches without subscription
- TAIR is manuylla curated and might have better annotations in some cases
- Araport11 provides data released by TAIR

Course tasks/questions:

1) How many protein coding genes are annotated in Col-0?

27.667 (Araport11)

27.416 (TAIR10)

2) How can you find puplications, which describe specific genes?

TAIR allows searches via Arabidopsis Gene Identifier (AGI, example: AT4G22880) or gene name (example: LDOX). In addition to detailed information about the respective gene, TAIR provides a list of publications mentioning this gene. Araport11 provides a similar functionality and lists publications as well.

3) Are there any specific website for other model organisms?

Table 8: Specific databases for model organisms.

Organism	**Specific database**
M. musculus	Mouse Genome Informatics http://www.informatics.jax.org/
C. elegans	Wormbase
D. melanogaster	FlyBase
Corynebakterium	Corynenet
Volvox	http://genome.jgi.doe.gov/Volca1/Volca1.home.html

Although sequences are available at the NCBI/EBI/DDBJ, dedicated databases for certain model organisms might have additional information. In addition, the visualization is often superior on webpages developed by these research communities.

After successful *ab initio* gene prediction, BLAST was deployed to map representative Col-0 exons (provided by Araport11) to the predicted Nd-1 sequence. This step was done to assess the assembly completeness. The speed of BLAST searches can be increased by constructing a BLAST database instead of searching against a FASTA file. However, this step is optional.

Example command for nucleotide database construction:

$ makeblastdb -in **sspace_results2.final.scaffolds_trimmed.fasta** -out **/db_blast_trimmed_scaffolds_2_sspace** -dbtype 'nucl'

Three database files were generated:

db_blast_trimmed_scaffolds_2_sspace.nsq

db_blast_trimmed_scaffolds_2_sspace.nin

db_blast_trimmed_scaffolds_2_sspace.nhr

However, only the base name of this database without any file extension is provided when running a BLAST search.

Example command to run a BLAST search and thus perform the functional annotation:

$ blastn -query **Araport11_based_repr_exon_file.fasta** -db **db_blast_trimmed_scaffolds_2_sspace** -out **blast_Col0_vs_Nd1_results** -outfmt 6 -evalue 0.001 -num_threads 4

- -outfmt: specifies the output format (table). This result format can be easily read by scripts in downstream analyses.
- -evalue: cutoff value to remove spurious hits. This value provides the probability of getting a hit with a similar quality by chance.
- -num_threads: specifies number of threads to use during search

Araport11_based_repr_exon_file.fasta contains the representative transcripts of all genes in Araport11. Alternative transcripts were discarded by selecting for the transcript which encodes the longest peptide sequence.

Output: blast_Col0_vs_Nd1_results

This file contains the results of a BLAST search of Col-0 exons vs. the Nd-1 assembly:

AT1G04090:exon:2	scaffold45	71.04	518	127	22	933	1440	3079	3583	2e-22	104

BLAST manual explaining output format 6:

http://www.metagenomics.wiki/tools/blast/blastn-output-format-6

	Name	Meaning
1.	qseqid	query (e.g., gene) sequence id
2.	sseqid	subject (e.g., reference genome) sequence id
3.	pident	percentage of identical matches (Identität)
4.	length	alignment length (paar hundert bp)
5.	mismatch	number of mismatches (möglichst klein 100/500, 1/5)
6.	gapopen	number of gap openings (noch kleiner als mm)
7.	qstart	start of alignment in query (links immer kleiner als das Ende)
8.	qend	end of alignment in query
9.	sstart	start of alignment in subject (hit on reverse strand results in sstart > send)
10.	send	end of alignment in subject
11.	evalue	expect value (probability of getting a hit of similar quality by chance)
12.	bitscore	bit score (quality of alignment)

Customization of the returned columns is possible as explained here:
http://www.metagenomics.wiki/tools/blast/blastn-output-format-6
Desired output columns and their order can be specified like this:
-outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore"
(default values)
This BLAST result table provides a good overview about the similarities between Col-0 and Nd-1 and allows to answer the questions: ‘how complete is the assembly?‘, ‘how similar are genes?’.

Reciprocal Best Hits (RBHs)

RBHs can be used to transfer functional annotations between two data sets e.g. from Col-0 to Nd-1. The question of highest similarity should be answered. Similarity is used as indicator for a close relationship. RHBs are pairs of genes wich are mutually the best BLAST hit.
A customized python script was deployed to run BLAST in both directions and to compare the results for the identification of RBHs. Peptide sequences extracted via getAnnoFasta.pl from the Nd-1 gene prediction were used for this comparision against the Araport11 peptides.

Executing the python script identify_RBHs.py:

$ python identify_RBHs.py

$ python identify_RBHs.py -help

Both commands will display the usage of this script.

$ python identify_RBHs.py --prefix **RBH_results** --input1 **Araport11_genes.201606.pep.repr.fasta** --input2 **result_scaffolds_2.aa** --seq_type 'prot'

entries in data1: 9705

entries in data2: 759

number of matches: 691 (numer of identified RBHs)

Course tasks/questions:

1) How many RBHs did you identified?

691

2) How many percent of the predicted genes are in RBH pairs?

691/776 = 0,89 => 89 %

3) Map functional annotation from TAIR10 to identified RBHs via map_annotation.py!

Execution of map_annotation.py:

$ python map_annotation.py --input_file **RBH_file.txt**

number of entries in mapping table: 33602 (number of annotated genes in TAIR10)

number of annotated genes: 691

Output: RBH_file.txt_annotated.txt

The python script for identification of RBHs automatically generates a new output folder if needed. The identification of RBHs is independent of the order in which the two sequence files are provided. Functional annotation transfer was successful for 691 genes of Nd-1 which were annotated based on the functional annotation of their RBH in TAIR10.

6 Variant analysis

6.1 Theory

- **Resequencing**
  - Requires high quality reference sequence and annotation (e.g. model organism)
  - New genotypes/strains are sequenced and reads are mapped to reference sequence (no assembly necessary, just read mapping)
  - Detection of SNPs and small insertions/deletions (InDels)
  - Identified differences can provide hints about evolution
  - Detection of rearrangements and large InDels is challenging
  - Identification of novel genes requires a *de novo* assembly
- **Read mapping**
  - Alignments of reads against a given reference sequence
  - BLAST is too slow for this alignments step due to extremly high numbers of reads
  - Dedicated ‘mappers‘ e.g. BWA (Burrows-Wheeler Alignment) are required
  - Read mappers are very fast, but have reduced accuracy
- **Variant calling**
  - Search for differences between reference sequence (Col-0) and reads (Nd-1)
  - IMPORTANT: only small differences in sequences represented in the reference sequence can be detected and regions without mapped reads could indicate TEs
- **GATK (Genome Analysis Tool Kit) concept**
  - Frequently applied in human genomics projects
  - GATK starts with a read mapping and realigns reads around putative InDels
    - Requires mapping of reads to reference sequence and removal of PCR duplicates
  - InDel detection is challenging due to ambigious read mapping; realignment / local *de novo* assembly increase the reliability of InDel detection
  - Realignment is followed by Base Recalibration => correction of per base quality value
  - Joint Genotyping: variant calling can be performed on multiple samples in parallel => increased sensitivity (not applied here)
    - Multiple samples displaying the same variant at low frequency support each other
  - Variant calling results in reported SNPs and InDels（Filtering dependent on variant type)
  - Variant calling is followed by downstream analyses e.g. impact prediction / functional annotation

6.2 Practical application

BWA-MEM is deployed to map the Nd-1 sequencing reads tot he Col-0 reference sequence. This requires the generation of an index file (TAIR10.fai). Afterwards, BWA-MEM is running a preparation of the FASTA file prior to the actual mapping process.

Indexing of reference via BWA:

$ bwa index **TAIR10.fa**

Multiple new files are generated in this step:

TAIR10.fa.sa; TAIR10.fa.pac; TAIR10.fa.ann; TAIR10.fa.amb; TAIR10.fa.bwt

Construction of these index files needs to be done only once. The same index can be used again for other mappings to the same reference sequence.

Mapping of Nd-1 reads to the Col-0 reference sequence:

$ bwa mem -M -t 4 **TAIR10.fa nddata_fwpaired.fq nddata_rvpaired.fq** |gzip -3 > **bwa_results**

- bwa mem: starts BWA with specific settings for long reads
- -M: suppresses soft clipping
  - single reads are flagged as low quality but not discarded
  - increases the alignment stringency
  - soft clipping could cause reads to be mapped in wrong positions due to clipping of all unmached bases
- -t: specifies number of CPUs to use
- Reference sequence
- Input forward reads
- input reverse reads
- paired-end mapping is performed if two files with reads are provides
- single end reads would come in a single file
- |gzip -3 OUTPUT_FILE
  - Compression via gzipg with compression level 3
  - Output is not written into file first, but directly compressed via gzip (| = pipe)

BWA generates a SAM file which contains the mapping details and is human readable. The command ‘cat’ in combination with ‘| head –n 100’ can be used to look at the mapping results. While cat would print the whole file content to the screen, head reduces it to just a couple of lines. Conversion of resulting SAM files into BAM files can speed up following processes, because this format is intended for automatic processing by tools. Although the content is not easy to read, zcat can be used to look at this compressed file.

Running zcat and head combination:
$ zcat FILE_NAME | head

This command reduces the printing of the complete file content to just a couple of lines. It is

usually helpful to check the content of files prior to downstream analyses.

$ zcat **bwa_result** | head

BWA is generating SAM files, but these can be converted into BAM files using samtools.

Conversion of SAM to BAM:

$ samtools view -bS -F 4 SAM_FILE >BAM_FILE

Samtools view would result in displaying the SAM file content on screen, but redirecting this into the desired BAM file is achieved via '>'.

$ samtools view -bS -F 4 **bwa_result.sam** > **bwa_result.BAM**

- -F 4: reads are discarded if flagged by -M as low quality mapping. There are different flags (numbers) assigned to each mapped read thus allowing filtering in downstream steps.
- -bS: output is modified to BAM format, while input is SAM format

The resulting BAM file can be subjected to downstream analyses like variant calling. However, several steps are required prior to the actual variant calling.

GATK comes with some requirements:

1. Dictionary file needs to be constructed for the reference sequence FASTA file CreateSequenceDictionary (piccard tools)
2. Index file needs to be constructed fort he reference sequence FASTA file: samtools faidx
3. input BAM file needs to be sorted by positions: samtools sort. This step will sort all reads by (1) order of sequences in the FASTA file and by (2) position on these sequences.
4. **Index of sorted BAM file** via BuildBamIndex (piccard tools)
5. **Read groups need to be added to BAM file** via AddOrReplaceReadGroups (GATK). The header of a BAM file is slightly modified to match GATK specifications.
6. Indexing of final BAM file via BuildBamIndex (GATK). Everytime the BAM file is slightly modified (sorting, adding read group), a new index is needed.

1. **Generating dictionary of reference sequence (TAIR10.fa):**

$ java -Xmx8g -jar picard.jar CreateSequenceDictionary R=**TAIR10.fa** O=TAIR10.dict

Output in terminal:

[Thu Mar 09 11:48:09 CET 2017] picard.sam.CreateSequenceDictionary REFERENCE=TAIR10.fa OUTPUT= TAIR10.dict TRUNCATE_NAMES_AT_WHITESPACE=true NUM_SEQUENCES=2147483647 VERBOSITY=INFO QUIET=false VALIDATION_STRINGENCY=STRICT COMPRESSION_LEVEL=5 MAX_RECORDS_IN_RAM=500000 CREATE_INDEX=false CREATE_MD5_FILE=false GA4GH_CLIENT_SECRETS=client_secrets.json
[Thu Mar 09 11:48:09 CET 2017] Executing as xxxx@yyy on Linux 4.8.14-300.fc25.x86_64 amd64; Java HotSpot(TM) 64-Bit Server VM 1.8.0_112-b15; Picard version: 2.5.0(2c370988aefe41f579920c8a6a678a201c5261c1_1466708365)
[Thu Mar 09 11:48:11 CET 2017] picard.sam.CreateSequenceDictionary done. Elapsed time: 0.04

minutes.
Runtime.totalMemory()=2058354688

Output file: TAIR10.dict
The file extension .fa is replaced by .dict in this file name.

2. **Indexing reference (TAIR10.fa):**
   $ samtools faidx **TAIR10.fa**

   Output file: TAIR10.fa.fai
   The file extension .fai is appended on the original file name to get the new name.

3. **Sorting BAM file (bwa_result.BAM):**
   $ samtools sort -@ 4 -m 4G -o **bwa_results_sorted erg_readmappingbam**

   Sorted BAM file should be smaller than original file, because the sorted reads can be compressed more efficiently.

4. **Indexing BAM file:**
   $ java -Xmx8g -jar picard.jar BuildBamIndex I=**bwa_result_sorted**

   Output-Datei: bwa_result_sorted.bai

5. **Adding readgroups:**

   $ java -Xmx8g -jar picard.jar AddOrReplaceReadGroups I=**bwa_result_sorted** O= **bwa_result_sorted_readgroups** RGID=1 RGLB=lib1 RGPL=illumina RGPU=unit1 RGSM=20

   These readgroups were selected randomly as they are meaningless for the next steps:
   1. RGID= read group ID
   2. RGLB= read group library
   3. RGPL= read group platform lane
   4. RGPU= read group platform unit
   5. RGSM= read group sample name

   The resulting BAM file is indexed again (see step 5).

After preparing the BAM file in this way, PCR duplicates need to be removed via MarkDuplicates (piccard tools). Since the Illumina protocol for library preparation includes a PCR step, it is possible that identical fragments are sequenced multiple times. The resulting information is useless or even detrimental if considered as replicates. Therefore, it is important to remove such duplicates. There are two ways to remove duplicates:

1) Based on identical terminal alignment positions of reads. It is highly unlikely that two identical DNA fragments are generated during DNA fragmentation and both end up being sequenced. Even if this occurs, removing such fragments should not have a big impact.
2) Reads originating from neighboring spots on the flow cell might be due to errors in the cluster identification process (e.g. one cluster read twice).

The following steps are necessary to mark PCR duplicates for exclusion from downstream analyses:

Executing MarkDuplicates:
$ java -Xmx8g -jar picard.jar MarkDuplicates INPUT=**bwa_result_sorted_readgroups** OUTPUT=**marked_duplications** METRICS_FILE=**metrics** OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 CREATE_INDEX=true TMP_DIR=**tmp**

- Metrics: log file with reports about identified duplicates
- OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500: distance of clusters on flow cell
- CREATE_INDEX=true: index oft he resulting BAM file is generated automatically
- TMP_DIR: directory for temporary files

The first step in GATK is the realignment of reads around putative InDels (IndelRealignment). As the reliable detection of InDels from reads mappings is challenging, this step substantially contributes to the quality of reported InDels. First, a list of all relevant positions is generated:

Execute RealignerTargetCreator (GATK):

$ java -Xmx8g -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -R **TAIR10.fa** -I **marked_duplications.bam** -U ALLOW_N_CIGAR_READS -o **realignment_targets.list**

- -T RealignerTargetCreator: generates list with all targets for read realignment
- -R: reference sequence file
- -I BAM_FILE: BAM file with marked PCR duplicates
- -U ALLOW_N_CIGAR_READS: suppresses error messages from erroneous alignments
- -o RESULT_FILE: it is very important to use the .list file extension here

Output: realignment_targets.list
This resulting list contains all positions of putative InDels which require realignment of the reads mapped in the neighbourhood. Example:
Chr1:5.044.381-5.044.386
Chr1:5.044.686

Although most steps do not check the file extensions, it is very helpful to provide them. Most tools are not casesensitive with respect to file extensions anyway.

After identification of the realignment targets, the actual realignment can be started. GATK is performing a *de novo* assembly at critical positions to resolve all alleles correctly.

InDel realignment via IndelRealigner (GATK):
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T IndelRealigner -R **TAIR10.fa** -I **marked_duplications.bam** -targetIntervals **realignment_targets.list** -U ALLOW_N_CIGAR_READS -o **realignments.bam**

Output: realignment.bai, realignment.bam

After the realignment step, the actual variant calling was performed using the HaplotypeCaller (GATK) to detect variants between Col-0 and Nd-1. SNPs and InDels are detected in this process and written to a joined VCF (=Variant Calling Format) file.

Executing variant calling via HaplotypeCaller (GATK):
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R **TAIR10.fa -I marked_duplications.bam** --genotyping_mode DISCOVERY -stand_emit_conf 10 -stand_call_conf 30 -U ALLOW_N_CIGAR_READS -o **haplotypecaller_result.vcf**

- genotyping_mode DISCOVERY: allows detection of new variants in contrast to just checking predefined positions
- stad_emit_conf 10: variants above this cutoff are considered
- stand_call_conf 30: variants above this cutoff are reported

The resulting VCF file contains all detected variants between Nd-1 and Col-0:

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	20
Chr1	5044354	.	T	C	685.77	.	AC=2;AF=1.00;AN=2;DP=31;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=31.17;SOR=1.931		

When searching for new variants (in plants), there is no ID for variants available. This is different in human genetics, where many variants are already known and many are even well described. REF and ALT display the alleles in the reference and the reads, respectively. Depending on ploidy and heterozygosity, many different alternative alleles are possible. The reported quality value depends on the sequence around the variant and the resulting mapping quality scores of the considered reads.
As the mapped Nd-1 reads were originating from chromosome 5, it is not surprising that most variants were detected on chromosome 5 of the Col-0 reference sequence (90%). Variants on other chromosomes could be due to translocations (e.g. TEs) or repeats between both accessions.

For further filtering, the VCF file is splitted into one VCF containing onely SNPs and one VCF containing only InDels, respectively. Different filter criteria are applied to these files during the next step.

Selection of SNPs:
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T SelectVariants -R **TAIR10.fa** -V **haplotypecaller_result.vcf** -selectType SNP -o **SNPs.vcf**

Output:
SNPs.vcf: file containing all SNPs
SNPs.vcf.idx: automatically generated index file

Filtering of SNPs via VariantFiltration:
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T VariantFiltration -R **TAIR10.fa** -V **SNPs.vcf** --filterExpression "QD < 2.0" --filterName "QD_filter" --filterExpression "FS > 60.0" --filterName "FS_filter" --filterExpression "MQ < 40.0" --filterName "MQ_filter" -o **clean_snp.vcf**

Output:
clean_snp.vcf
clean_snp.vcf.idx

Three different filers are applied: QD, FS, MQ (LowQual-Filter of GATK). The resulting VCF file contains a column where the filtering status is stated: PASS or filter name, respectively. Although no variants are removed from the file, they are flagged accordingly. Example:

#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT 20
Chr1 5044354 . T C 685.77 PASS
AC=2;AF=1.00;AN=2;DP=31;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;
QD=31.17;SOR=1.931
Chr1 12692203 . T C 91.28 MQ_filter
AC=2;AF=1.00;AN=2;DP=3;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=36.70;Q
D=30.43;SOR=2.833 GT:AD:DP:GQ:PL 1/1:0,3:3:9:119,9,0
Chr1 12845531 . A G 18.59 LowQual
AC=2;AF=1.00;AN=2;DP=13;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=49.06;
QD=18.59;SOR=1.609 GT:AD:DP:GQ:PL 1/1:0,1:1:3:45,3,0

The first SNP passed the filtering, the second one failed the MQ filter, and the third one was automatically filtered by GATK due to a low quality. The resulting VCF file could be cleaned by a customized Pthon script to actually remove SNPs that failed a filter.

After extraction of SNPs, this process is repeated for InDels. Separate processing of both variant types is necessary due to different filter critera.

Extraction of InDels via SelectVariants:
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T SelectVariants -R **TAIR10.fa** -V **haplotypecaller_result.vcf** -selectType INDEL -o **indel_extracted.vcf**

Output:
indel_ extracted.vcf.idx
indel_ extracted.vcf

Filtering of InDels via VariantFiltration:
$ java -Xmx8g -jar GenomeAnalysisTK.jar -T VariantFiltration -R **TAIR10.fa** -V **indel_extrahiert.vcf** --filterExpression "QD < 2.0" --filterName "QD_filter" --filterExpression "FS > 60.0" --filterName "FS_filter" --filterExpression "MQ < 40.0" --filterName "MQ_filter" -o **clean_indels.vcf**

Output:
clean_indels.vcf.idx
clean_indels.vcf

Example:

#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT 20
Chr1 5044385 . C CT 818.73 PASS
AC=2;AF=1.00;AN=2;DP=28;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=25.
68;SOR=1.022
Chr1 12691952 . T TG 252.77 MQ_filter
AC=2;AF=1.00;AN=2;DP=7;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=36.73;QD=29.5
5;SOR=0.941 GT:AD:DP:GQ:PL 1/1:0,7:7:21:290,21,0
Chr2 9030531 . TG T 22.75 LowQual;MQ_filter
Chr5 5210791 . A
ACGTAAACGAGACCTTTACATGAAACTATCAAACCAGTTTGAGTTTATCCGATTTTGCTAATGAGACGAGTT

770.73 PASS
AC=2;AF=1.00;AN=2;DP=15;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=55.93;
QD=30.68;SOR=1.765 GT:AD:DP:GQ:PL 1/1:0,14:14:53:808,53,0

The first InDel passed the filtering process, while the second and third one failed the MQ filter. In addition, the third one was flagged by GATK as low quality.

To roughly estimate the SNP and InDel frequencies, the respecive VCF file was searched for variant on ‚Chr5'. Additional quantification was performed for the other chromosomes (Table 9). The distribution over the chromosomes matches the expectation that most variants should be located on chromosome 5 due to the origin of the mapped reads from chromosome 5. Hits on chromosomes 1-4 could be considered as false positive variants unless they are located in translocations between Nd-1 and Col-0.

Table 9: Variants per chromosome.

Listed are the number of detected SNPs and InDels per chromosome. Due to the reduced data set only variant on chromosome 5 were expectd.

Chromosom	**SNP Variationen**	**Indel Variationen**
Chr1	261	23
Chr2	383	66
Chr3	220	33
Chr4	390	75
Chr5	15.305	4.506

Calculation of variant frequencies:

total contig size = 3 Mbp = 3,000,000 bp (Nd-1 contig which is source of mapped reads)

SNPs: 3000000 Mbp/14000 SNPs = 1 SNP / **214 bp**
Indels: 3000000 Mbp/4000 Indels = 1 InDel / **750 bp**

Average frequencies of SNPs and InDels based on literature (Table 10):
1 SNP / 50-500 bp
1 InDel / 300-3.000 bp

Table 10: Variant frequencies of different species.

This table contains variant frequencies in various species collected from publications.

species	**SNP frequency**	**InDel frequency**
Oryza indica	170 bp	
Oryza japonica	350 bp	
Homo sapiens	1 kb in non-coding DNA 3 kb in proteincoding DNA	

The variant calling results between Nd-1 and Col-0 are biased due to the read selection. However, the observed trends are matching reports in the literature e.g. higher SNP frequency than InDel frequency. In addition, more careful estimation of the variant frequency and the variant distribution over the chromosomes could improve the results.

7 Annotation of variations

7.1 Theory

- **Annotation of variation**
    - Each variation can have one or (more often) multiple effects and affect multiple genes at the same time
    - The strength of this effect depends on the type and position of the variation
        - Mutations, which have a strong effect, are usually in the CDS
        - A functional annotation usually occurs on the influence (type, position, strength) of the variation on the particular gene
            - Usually the variation, which has the strongest effect on the target gene is examined.
- **SnpEff (SNP Effects)**
    - Tool to determine variation effects
    - Is the first step in answering the question "What is the effect of the variation on the gene product?"
    - Many different databases with genome annotations are available for comparison
        - Uses genome annotation of organisms in the form of a GFF (General Feature Format) file
            - For this purpose, the genome sequence and associated annotation of the organism are put together (externally or do it yourself)
    - Annotates each variation in a VCF file by creating a new annotated VCF file
    - Big advantage: The program is very fast (1000 variation per minute)
    - Big downside: The program looks at only one variation alone, thus does not find effects triggered by multiple variations

7.2 Practical application

After the variations between Col-0 and Nd-1 have been identified and the variation frequencies have been determined, the effects of the variations are then examined in detail.

Example command to determine the effects of the variations:

$ java-Xmx4g-jar snpEff.jar-v athaliana130 haplotypecaller_result.vcf > snpEff_result_Effects_variationen.vcf

- Xmx4g: 4 Gb RAM
- v: Specify the database to be used for comparison. We use the latest *A. thaliana* database (athaliana130)

Output: snpEff_result_effects_variations.vcf, snpEff_genes.txt, snpEff_summary.html

Addition: Determination of all databases available at SnpEff via $ java-jar snpEff.jar databases. It is possible to output all the databases that are available (not only *A. thaliana* possible).

Example command to store a list of available database from SnpEff locally (in the snpEff folder):

$ java-jar snpEff.jar databases > database_of_snpEff.txt

Output: Database_by_snpEff.txt

Links are specified in the file under which you can download the respective database of interest: http://downloads.sourceforge.net/project/snpeff/databases/v4_1/snpEff_v4_1_athaliana130.zip

The first entry refers to the database of “Hordeum_vulgare”. To get its database, "Hordeum _ vulgare" would be entered into the console. For *A. thaliana* there are several databases, but the program automatically takes the latest. You would enter athaliana130 (see sample command above).

The snpEff_summary.html-file shows a summary of the results.
The summary is opened by double-clicking on the file in the browser:

Summary	
Genome	athaliana130
Date	2017-03-10 11:46
SnpEff version	SnpEff 4.1f (build 2015-05-12), by Pablo Cingolani
Command line arguments	SnpEff athaliana130 /vol/agrcourse/members/hschilbert/GATK_SNP_Indel_Identifizierung/ha
Warnings	479
Errors	0
Number of lines (input file)	21,266
Number of variants (before filter)	21,274
Number of not variants (i.e. reference equals alternative)	0
Number of variants processed (i.e. after filter and non-variants)	21,274
Number of known variants (i.e. non-empty ID)	0 (0%)
Number of multi-allelic VCF entries (i.e. more than two alleles)	8
Number of effects	94,867
Genome total length	119,667,750
Genome effective length	119,146,348
Variant rate	1 variant every 5,600 bases

Figure 2: Summary of the annotation of the variations via SnpEff.

The summary shows that more effects than variations have been found (Figure 2):

Number of variations: About 21.270
Number of effects: 94,867

This is because a variation can (depending on its position) affect multiple genes at the same time (e.g. variation x lies in gene 2, then it could have at the same time an effect on gene 1 located in front of gene 2 or on the gene 3 behind gene 2).

The variation rate (last point in the summary) indicates how many variations were found per bp.

Looking at the variation rate of individual chromosomes it can be seen that the rate differs from chromosome to chromosome. This is due to the different size of the chromosomes and to the fact that we received pre-selected data and thus could not identify all the variations on the remaining four chromosomes.

SnpEff's results also show that the variation frequency on chromosome five is significantly higher compared to the other four chromosomes (Figure 3).

For chromosome five, we got a variation rate of 1.360 (Figure 3). This is much higher than the variation frequency that was calculated manually after filtering the SNPs and InDels. In addition, SnpEff has calculated about the same number of variations as we did manually.

Variants rate details			
Chromosome	Length	Variants	Variants rate
1	30,427,671	284	107,139
2	19,698,289	449	43,871
3	23,459,830	253	92,726
4	18,585,056	467	39,796
5	26,975,502	19,821	1,360
Total	119,146,348	21,274	5,600

Figure 3: Calculation of the variant frequency via SnpEff.

The determined variation frequencies per chromosome are shown. It also lists the total length of variations and the number of variations per chromosome.

How does the increased frequency of variation come about via SnpEff?

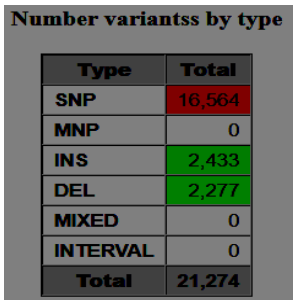
1. SnpEff expects the complete genome sequence of *A. thaliana* (120 Mbp), we only expected 3 Mbp!

= > 21,000/3 Mbp provides a much higher variation frequency than 21,000/120 Mbp!

2. Is filtered again, but this effect is very low

In addition, the filtered out variations via SnpEff were removed and thus were not considered in the calculation of the variation frequency (as it was the case with us). Furthermore, the higher the coverage, the more variations can be identified.

The ratio of Indels to SNPs is approximately 1:3 (4,700/16:500)(`Figure 4`). Manual a similar ratio was calculated (4,500/17,000).



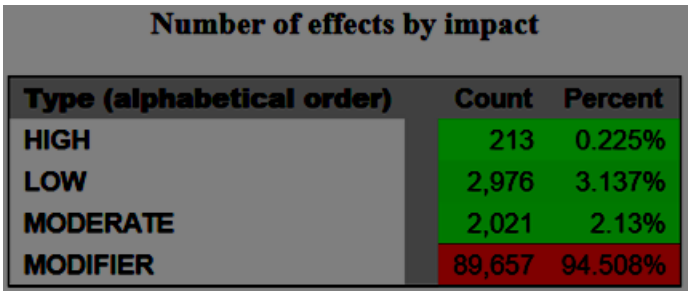
Number variantss by type

Type	Total
SNP	16,564
MNP	0
INS	2,433
DEL	2,277
MIXED	0
INTERVAL	0
Total	21,274

Figure 4: Variation types.

This ratio is consistent with the expectation that more SNPs can be identified as Indels, because SNPs have less impact than Indels. Silent mutations (term not entirely correct, they can still have an impact) triggered by SNPs often occur without having an effect on the resulting gene product (due to the degeneration of the genetic code). Thus, SNPs are subject to less selection pressure than Indels (SNPs "manifest" faster than Indels).

The effects identified by SnpEff are evaluated and counted according to their strength (`Figure 5`). `Figure 5` represents the individual gradations: HIGH, LOW, MODERATE, and MODIFIER of the effects of the variations.



Number of effects by impact

Type (alphabetical order)	Count	Percent
HIGH	213	0.225%
LOW	2,976	3.137%
MODERATE	2,021	2.13%
MODIFIER	89,657	94.508%

Figure 5: Number and percentage of high/low/moderate/modifier effects/variations.

Certain types of variations are classified as HIGH. For example, if the line with HIGH is green, this means that there are few variations of this type, which have a strong influence. However, it is recommended to pay more attention to numbers, as 0.225% is still a lot. Due to our pre-selected region, we got a comparatively high value at this point, as it the region is very rich in genes.

In addition, there are 213 HIGH classified effects, which account for 0.225% of all effects and about 90,000 effects, which are classified as MODIFIER and thus have virtually no influence on e.g. a gene product (`Figure 5`). This would be the case if the variations are localized in the intergenic area or (applies only to a limited extent if there are no enhancers etc.) in the 5 ' or 3 ' UTR.

In addition, the percentage of variations can be determined depending on their localization via SnpEff (`Figure 6`). The bar chart provides an overview of the distribution of variations on the gene structure.

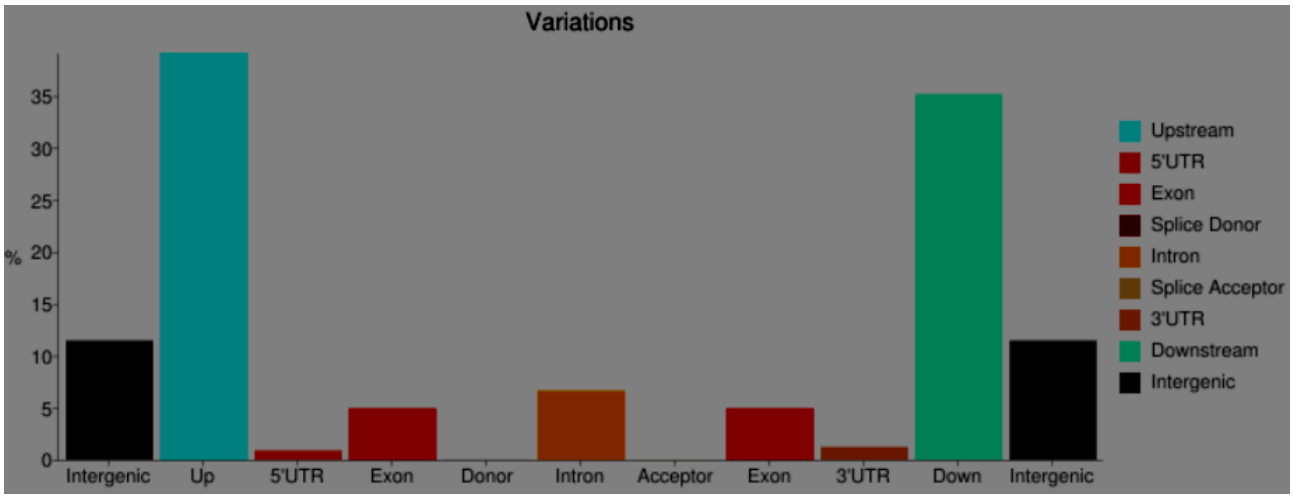


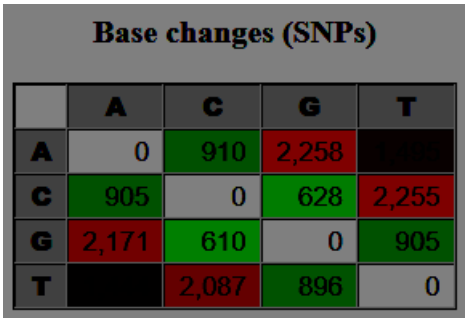
Figure 6: Localisation of the variations.

Shown is the localization of the variations via SnpEff. The x-axis maps the localization possibilities. The y-axis shows the proportion of variations on the respective localization in percentage. Here, the variations can be localized upstream or downstream (or both). Variations can lie in the 5 ' or 3 ' UTR. They can also be located exons, introns, as well as intergenic areas or in splice sites.

- About 37% of all variations are upstream (Figure 6). This means that the variation is in front of the gene. Such a variation usually has no effect. 35% of the variations are located downstream of genes.
- In addition, there are (almost) no variations in the splice-donor and acceptor sites (Figure 6), as splice sites are subject to strong selection pressure. Furthermore, this low value can be explained by the size of the splice sites comprising only two bp. Thus, it is unlikely that a variation is localized there.
- Conversely, this means that the chances of hitting an exon (5%), intron (8%) or another larger region (e.g. intergenic area: 25%) are much higher (Figure 6).
- **Why do we find only few variations in the big 3 ' and 5 ' UTRs (Figure 6)?**

Because it is not always known where exactly these areas lie. They are both experimental and bioinformatics difficult to find and therefore not well or not annotated in the reference sequence. Therefore, (almost) no variations in the 3 ' and 5 ' UTRs could be identified.

In addition, SnpEff determines the number of base changes or SNPs for each of the four bases of the genetic code (Figure 7).



Base changes (SNPs)

	A	C	G	T
A	0	910	2,258	
C	905	0	628	2,255
G	2,171	610	0	905
T		2,087	896	0

Figure 7: Base changes (SNPs) of purine and pyrimidine bases.

The number of base changes or SNPs on the four bases of the genetic code are shown.

The result is not in line with the expectation that all four bases are equally likely to be affected by SNPs, so SNPs happen to occur randomly. This is due to the specific structure of the bases: Transversions are more common because their effects are milder: For example, when an A is swapped for a G, this does not produce a stop as often (Stoppcodons: TAG, TAA, TGA) as when a C is swapped for an A. In addition, it is easier to carry out a transition from a chemical point of view, as the bases to be replaced are very similar.
Red marked errors indicate that this exchange is particularly common. In our case, as explained above, this is the case with the transitions.

Transition: Purin vs. Purin (G <=> A)
Transition: Pyrimidine vs. pyrimidine (T <=> C)
Transversion: Purine vs. pyrimidine (G <=> T or A <=> C or G <=> C or T <=> A)

In addition to the .html file, which illustrates the results of SnpEff in the browser as described above, SnpEff also produces a .vcf file (snpEff_result_Effects_variation.vcf). This contains a list of all effects, first ordered by chromosome/localization, then according to the respective variation (e.g. a SNP), followed by the genes that experience a specific effect through the variation (also ordered by the position in the genome). The sample entry of the .vcf file below shows that for each SNP or InDel (e.g., on Chr1 in position 5.044.354) all possible effects on different genes are listed with the accompanying evaluation (e.g. modifiers):

#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT 20
Chr1 **5044354** . T C 685.77 .
AC=2;AF=1.00;AN=2;DP=31;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=31.17;SOR=1.931;ANN=C**|upstream_gene_variant|MODIFIER|AT1G14687|AT1G14687|transcript|PAC:17348431|Noncoding||n.-244T>C||||3449|,**C|downstream_gene_variant|MODIFIER|AT1G14670|AT1G14670|transcript|PAC:17348425|Noncoding||n.*1836T>C||||3826|,C|downstream_gene_variant|MODIFIER|AT1G14680|AT1G14680|transcript|PAC:17348426|Noncoding||n.*873T>C||||2621|,

Exaplanation of the structure of an "effect entry" based on the red marked example above:

| Localization or type of effect | Effect strength evaluation | Gene ID, the affected gene (trivial name) | Gene-ID, the affected gene (systematic name) | What is affected? E.g. the transcript | PAC number = unique ID for effect in Col-0 | | Variation | | | | | Distance of variation to the gene in Col-0 |

- All information from the entry relate to the possible effects in Col-0
- Several genes can also be affected by the same effect
  - A variation can have several effects. For example, if a variation is localized simultaneously in the CDS of Gen1, but also downstream of Gen 0 and upstream of Gen3. As a result, you often get multiple entries for one SNP
  - (E.g.: In the case of an intergenic effect: AT5G19257-AT5G19260 | AT5G19257-AT5G19260)
- Trivial name and systematic name is the same at Col-0
- n.-244T>C means that the base at position 244 of the gene, i.e. a thymine, has become a cytosine: T in *A. thaliana* Col-0 and C in Nd-1
- 3449 distance to the gene
  - For example, if it is an upstream gene variant
- Commas mark the beginning and end of an entry
- For a detailed description of the entries, you can read the manuals in the internet

Course tasks/questions:

1) Which mutation types are high deletirious?

The identification of the "high " mutations is carried out as follows: In the snpEff_result_Effects_variation.vcf file is searched for "HIGH " (or the other classifications or even individual mutations) via Strg + F, which can be found in the .html summary.

High mutations include:

frameshifts_variant by far the most common!
stop_lost
splice_region_variant
start_lost
splice_acceptor_variant
intron_variant
splice_donor_variant
disruptive_inframe_deletion
splice_region_variant
stop_gained

Some mutations also occur together only.

2) How many critical mutations are included in the resequenced interval?

Answering the question depends on what is considered as critical. The high mutations are considered as critical.

A total of about 707 (`Table 11`). For this purpose, the list "Number of effects by type and region" in the .html summary was used to identify the frequency of the mutations identified in task 1. Assuming one mutation per gene would mean to have about 700 genes in 3 Mbp and this is unlikely.
Once again, the problem is that there is no 1:1 relationship! A gene has n different variations and each variation has n different effects. **So how to filter sensibly (that's question 3)?**

Table 11: Extract of the frequencies of the occurring mutations.
The frequency of the "HIGH" mutations occurring are listed.

Mutation	Frequency
frameshifts_variant	160
stop_lost	1
start_lost	4
splice_acceptor_variant & intron_variant	8
intron_variant & splice_donor_variant	6
splice_region_variant & intron_variant	402
disruptive_inframe_deletion	37
splice_region_variant	20
stop_gained	13
splice_acceptor_variant+splice_region_variant+intron_variant	1

3) How to filter the most important effects of each mutation?

You could write a python script for this, because manually it would be very elaborate.
For this, the documentation (manual page) would first require a classification of all possible mutations. Then one would analyze each block entry of each mutation chronologically. Next, HIGH/LOW/MODERATE/MODIFIER could be searched in every effect entry. This order should be ranked to use it in a way that as soon as a mutation with HIGH strength is found within an effect entry, it is further analyzed. If there is only one HIGH entry (in one block), it will be issued. For two possible HIGH entries (in one block), it must be filtered according to another criterion: This information is obtained by the script from the classification of mutations, which were previously sorted according to HIGH/LOW/MODERATE/MODIFIER. This allows the most critical variation to be issued for each block.

Ways to create a HIGH classification file:
1.Stop_gained
2.Splice_site
3.Frameshift

It could also be filtered for genes. Then the first thing to do would be to search for the gene ID. This would allow to pay attention to mutations that lie behind the gene and therefore filter according to the position of the mutations.

4) What is the function of mutant genes?

Examples 1: AT5G16286 Tair link to locus:
https://www.arabidopsis.org/servlets/TairObject?id=1501131229&type=locus
T | start_lost | HIGH | AT5G16286 | AT5G16286|transcript | PAC:17373965 |
Noncoding|1/1|n.3G>A|p.Met1? | 3/141 | 3/-1 | 1/-1 | | | |

Short description of the mutation and the effect:

Mutation: HIGH; Start_lost at third base n.3, where a G (Col-0) became an A (in Nd-1), so the original ATG (Col-0) became an ATA and therefore there the start codon is lost.
This would mean that no functional gene product would be formed.

For annotation, details can be searched under annotation:
https://www.arabidopsis.org/servlets/Search?action=search&type=annotation&tair_object_id=4515103575&locus_name=AT5G16286

The search revealed that the mutation lies on mitochondrial DNA and the function of the gene product is unknown.

Possible resulting phenotypes:
Example 1: No statement possible because no function is known.

5) Why don't the base patterns change randomly?

Observing the distribution of the InDels revealed that InDels, which are multiples of three are more common within coding regions. In addition, the longer an InDel is, the rarer are InDel lengths, which can not be divided by three. Moreover, shorter InDels are generally more common. This is related to the selection pressure: An InDel consisting of a multiple of three bp does not produce a frameshift and is therefore rather tolerated. In addition, a small InDel of a base size can result in a silent mutation, so the selection pressure is low. Indels are rarely found in protein-coding regions when they are not multiples of three or very large. In addition, InDels with a length of three bp are more common compared to InDels with a length of two, as there is a stronger selection pressure against InDels with a length of two bp.
An InDel with a length of one bp can also be called a SNP. Single nucleotide polymorphism is also known as SNP: However, these can be larger than a base.

8 RNA-Seq: RNA to reads to counts

8.1 Theory

- **RNA-Seq**
  - Method for transcriptomic analyses
  - RNA is isolated, reverse transcripted into cDNA, amplified, and sequenced => data
  - Application: one of the best method to investigate gene expression
  - **Gene expression analysis:**
    - Many tags required (single end sequencing)
      - Tags are reads (single end) or fragments (paired-end)
      - Paired-end reads can only be counted once, because the originate from the same molecule
      - Paired-end allows more specific read mapping
    - Comparision of genotypes/conditions/tissues
      - Healthy vs. damaged/infected tissue
      - Investigation of one gene in different samples
  - ***De novo* transcriptom assembly**
    - Samples from different tissues/conditions are used to increase diversity
    - Paired-end applied to increase assembly continuity
    - Identification of novel transcripts/genes
      - Analysis without (genomic) reference sequence
    - Differences in coverage pose an additional problem to the assembly process
      - Genomic reference should be used if available
    - Ttranscriptom assembly is cost-efficient way to identify most genes
      - Transcriptom size of all diploid plants is around 70-100 Mbp
      - Stronger variation in genome size

- **Microarray**
  - Classical method to quantify transcription
  - Detection of novel transcripts is impossible
    - Detection is limited to known cDNAs
    - Not suitable for small RNAs
  - Microarrays come with limited dynamic range
      - Oligonucleotides (probes of microarrays) can be saturated by high transcription

level => upper limit for detection

- RNA-Seq enables unlimited quantification

- Costs: microarrays are cost-efficient if used in high throughput

- **Genexpression analysis - workflow**
  - Collecting sample
  - Generating reads (sequencing)
  - Read mapping (STAR)
  - Read counting (summarization of mapped reads)
  - Statistical analysis to identify differences between samples

- ***De novo* transcriptom assembly**
  - mRNA isolated and reverse transcripted into cDNA; subjected to sequencing => *de novo* assembly
  - Calculation of statistics to assess assembly
    - Distribution of contig (=unigene) lengths
    - Functional annotation of unigenes with GO (GO = Gene Ontology)
  - Answering biological questions

- **RNA and DNA structure and differences**
  - RNA: ribose, single strand, C/G/A and U bases
  - DNA: desoxyribose, double stranded, C/G/A and T bases
  - RNA hast o free OH groups and is thus more reactive compared to DNA
    - Biochemical differences between RNA and DNA are important when specifically enriching RNA by DNA degradation

- **Types of RNA**
  - rRNA, tRNA, mRNA, miRNA, ncRNA… and many more!

- **RNA isolation – workflow**
  - Optimal protocol needs to be identified empirically
  - Trizol-based
    - Separation of phases: RNA in upper phase, DNA in interphase, and proteins in lower

phase; precipitation of RNA via ethanol

- *A. thaliana*: NucleoSpin® RNA Plant
- *V. vinifera*: Spectrum™ Plant Total RNA
- Selection of kit depends on species and tissue
  - RNases (proteines) need to be inactivated
    - RNase inhibitors
    - Specific buffers
    - Kits are fast => reduced degradation through RNase removal
- **DNA contamination**
  - DNA traces might remain due to imperfect nucleid acid separation
  - DNA degradation by...
    - DNase I treatment
      - DNase I digests specifically dsDNA
      - Step is crucial as gDNA cannot be distinguished from cDNA in later steps (repeated DNase I treatment would degrade cDNA)
    - Magnesium (cofactor of DNases) captured by EDTA
      - Important: EDTA amounts must be adjusted to magnesium concentration as excess would inhibit downstream applications
      - Reverse transkriptase depends on magnesium
    - Inactivation of DNasen by heat treatment
    - gDNA needs to be removed …
      - Effects gene expression quantification and *de novo* transcriptom assembly
        - gDNA contamination can cause highly fragmented assembly due to presence of intron sequences at low coverage
        - Deep sequencing picks up traces of gDNA

- Which methods could be used in RNA-Seq workflows (example):
  - Animals: https://www.ncbi.nlm.nih.gov/pubmed/21481219
  - Plants: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4103122/
  - Bacteria: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4673735/
  - Which kits are recommended: http://www.laborjournal.de/rubric/produkte/alle/LjPr-15-04.pdf

- **RNA quality control**
  - RNA agarose gel
    - Two clear bands indicate high RNA quality: 28S rRNA and 18S rRNA
    - Degraded RNA would cause smear
    - gDNA contamination can show up as large fragment band
  - NanoDrop
    - 260/280 ratio: indicates protein contamination based on phenylalanine, tryptophan, and tyrosine (aromatic amino acids)
    - 260/230 ratio: indicates small nucleic acid fragments and phenolic compounds
  - PicoGreen & RIN (=RNA integrity number)
    - Chip reveals RIN
      - RIN is between 1 (bad) and 10 (best RNA quality)
        - 8 indicates good quality
        - Below values of 3-4 RNA-Seq becomes impossible
      - RIN = ratio of area of 18S- and 28S- rRNA to total area under graph
        - 18S- and 28S-rRNA indicate RNA quality
        - Information about mRNA (not detectable) is inferred from rRNA
        - Peak der 28S rRNA ist eher instabil
      - Output is similar toRNA gel
        - Peak at end of diagramm would result from sample at top of RNA gel
        - Precise quantification via diagramm possible

- **RNA-Seq – library construction**
  - 80-85 % of total RNA is rRNA
  - rRNA depletion or mRNA (3-5 % of total RNA) enrichment required
  - RNA wird fragmentiert, um die gewünschte Fragmentgröße zu erlangen
  - RT into cDNA
  - Adapter ligation and processing (see gDNA library construction)
  - **Prokaryotes: rRNA depletion**
    - Via „ribosylation“
    - rRNA binds to biotinylad oligonucleotides
      - Biotin binds to magnetic beads
      - rRNA can be pulled down with magnetic beads

- **Eukaryotes: mRNA enrichment**
  - Via poly-A-tail or 5'-cap
    - Via poly-A-tail and oligo(dT)-column
    - 5' enrichment via adapter ligation and fragmentation of cDNA; RT focuses on transcript starts => identification of transcriptions start sites
  - Via adapters ligated at both ends
    - Advantage: introduces tags for multiplexing
    - Sequencing of pools and sorting of reads afterwards
    - Up to 72 samples can be pooled

- **Enriched mRNA is resulting**
  - mRNA can be fragmented prior to cDNA synthesis
  - Advantage: complete mRNA represented; cDNA synthesis is generating small molecules; avoids risk of underrepresented central mRNA parts

- **RNA-Seq read mapping**
  - Reads (or read pairs) represent transcripts
  - Reads need to be associated to genes
  - BLAST ist o slow => mapping tool required
    - **BWA und Bowtie** (mapper) cannot handle introns
      - Mapping only works for prokaryotes
    - Mapping parameters can be adjusted (e.g. alignment similarity, alignment length)
    - Mapper assume continuous alignment of read to reference sequence
  - **STAR is split-read mapper:**
    - Trimming is not necessary as low quality bases do not need to be mapped
    - Accounts for introns by splitting the alignment
    - Split read can improve gene prediction by identification of introns

- **Assignment and counting of reads – are reads mapped to genes?**
  - Mapped reads are assigned to genes
  - Reads (or pairs of reads) per gene are counted
  - Assignment can be performed on different feature levels (CDS, exon, transcript, gene)

- Tools: HT-Seq, featureCounts (extrem fast!)

8.2 Practical application

The following experiment compares the transcriptomes of *A. thaliana* Col-0 wildtype (WT) and a triple myb mutant. First, RNA-Seq reads are subjected to quality control:

Col0-1_ATCACG_L001_R1_001.fastq = Col-0 WT reads in text document
3xmyb_GGCTAC_L001_R1_001.fastq = Col-0 3xmyb mutant reads in text document

As the triple myb mutant lacks three transcription factors, effects on the whole transcriptome are expected.

Example commands to assess the quality via FastQC:

$ fastqc **Col0-1_ATCACG_L001_R1_001.fastq**
$ fastqc **3x**m**yb-1_GGCTAC_L001_R1_001.fastq**

Output of first command:

Col0-1_ATCACG_L001_R1_001_fastqc.zip
Col0-1_ATCACG_L001_R1_001_fastqc.html

Output of second command:
3xmyb-1_GGCTAC_L001_R1_001_fastqc.zip
3xmyb-1_GGCTAC_L001_R1_001_fastqc.html

Course tasks/questions:

1) Is the technical quality of the RNA-Seq reads OK?

Col0-1_ATCACG_L001_R1_001.fastq = Yes, the average phred score is 37 (30-40).
3xmyb-1_GGCTAC_L001_R1_001.fastq = Yes, the average phred score is 37 (30-40).

2) How long are the RNA-Seq reads?

Col0-1_ATCACG_L001_R1_001.fastq = 130 nt
3xmyb-1_GGCTAC_L001_R1_001.fastq = 130 nt

3) Any interesting observations?

The triple myb mutant data set is much smaller: 5,819,938 compare to 9,120,899 (WT). There are more duplicates reported in the larger data set of the WT sample. No adapter sequences were detected in any of the samples. Enrichment of k-mers is substantially higher than in the DNA analyses performed before. These overrepresented k-mers are the result of highly expressed genes. Both data sets display some reads with low quality due to tile-specific issues.

GC content of both data sets is approximately 45 %. This value exceeds the average GC content of *A. thaliana* (36 %), because transcripts are likely to have a GC content closer to 50%.

- 64 codons (4^3), genetic code / codon usage causes the increased GC content
- Equal frequency of all codons would result in a GC content of 50%
- AT-rich regions are genome-wide more frequent than in protein encoding regions due to

constrains to encode all amino acids AT-reiche Regionen, die kodierenden Regionen sind aber GC reicher um alle Aminosäuren
- Sequencing bias towards high GC fragments might contribute as well

Example commands to process data via trimmomatic:

WT data set:
$ java -jar trimmomatic-0.35.jar SE -phred33 **Col0-1_ATCACG_L001_R1_001.fastq Col0-1_ATCACG_L001_R1_001_trimmed.fq** ILLUMINACLIP:Illumina_adapters.fa:2:30:10 LEADING:0 TRAILING:5 SLIDINGWINDOW:4:15:10 MINLEN:100

3xmyb data set:
$ java -jar trimmomatic-0.35.jar SE -phred33 3xm**yb-1_GGCTAC_L001_R1_001.fastq** 3xm**yb-1_GGCTAC_L001_R1_001_trimmed.fq** ILLUMINACLIP:Illumina_adapters.fa:2:30:10 LEADING:0 TRAILING:5 SLIDINGWINDOW:4:15:10 MINLEN:100

Via 'TOPHRED' it was possible to set the phred score to phred33 thus avoiding issues related to different Illumina pipelines (v1.3, v1.5, v1.8) which are working with different encodings.

Mapping of sequencing reads requires some dedicated reference files which are constructed by STAR prior to the actual mapping process.

Example command for reference construction via STAR:

$ STARlong --runMode genomeGenerate --genomeDir ./ath_genome**/** --genomeFastaFiles **TAIR10.fa** --runThreadN 4 --limitGenomeGenerateRAM 40000000000 --genomeSAindexNbases 4

- --runMode genomeGenerate: command to construct reference for mapping
- --genomeDir: specifies directory for reference files
- --genomeFastaFiles: input FASTA file
- --runThreadN: number of threads for the reference construction
- --limitGenomeGenerateRAM: limits RAM consumed in the process
- --genomeSAindexNbases: sets index size for reference construction; more fragments slow down the mapping process, but also reduce the amount of required memory

Output: Log.out
The following files are generated in the output directory:
SAindex, SA, Genome, chrStart.txt, chrNameLength.txt, chrName.txt, chrLength.txt, genomeParameters.txt

The slash at the end of the --genomeDir path is important during the reference construction, but must be removed prior to the read mapping process as an additional slash is added by STAR.

Example command for read mapping via STAR:

$ STARlong --genomeDir ./ath_genome --readFilesIn **Col0-1_ATCACG_L001_R1_001_trimmed.fq** --runThreadN 4 --outFileNamePrefix ./**Col-0_WT /** --limitBAMsortRAM 40000000000 --outBAMsortingThreadN 2 --outSAMtype BAM SortedByCoordinate --outFilterMismatchNoverLmax 0.05 --outFilterMatchNminOverLread 0.8

Log in terminal:
Mar 13 15:13:12 ..... Started STAR run
Mar 13 15:13:13 ..... Loading genome
Mar 13 15:13:14 ..... Started mapping

- --genomeDir: uses previously constructed reference files for mapping
- --readFilesIn: trimmed RNA-Seq reads
- --outFileNamePrefix: output directory for mapping results
- --limitBAMsortRAM: limits consumed memory
- --outBAMsortingThreadN: limits number of threads for sorting of BAM file
- --outSAMtype BAM SortedByCoordinate: generates BAM file with reads sorted by mapping position on the reference
- --outFilterMismatchNoverLmax and --outFilterMatchNminOverLreadFilter
  - Filters alignment based on similarity and length
- If reads are available in compressed files, zcat can be used to subject them to STAR without creating spaceous uncompressed files
- Output path must end with slash

Output:

1) folder _STARtmp with temporary files
2) Result files: Log.progress.out, Log.out, SJ.out.tab, Aligned.sortedByCoord.out.bam

Example Log.final.out:

Started job on	Mar 13 15:29:56
Started mapping on	Mar 13 15:29:58
Finished on	Mar 13 15:37:48
Mapping speed, Million of reads per hour	64.41
Number of input reads	8409150
Average input read length	128
UNIQUE READS:	
Uniquely mapped reads number	8201439
Uniquely mapped reads %	**97.53%**
Average mapped length	128.42
Number of splices: Total	3239201
Number of splices: Annotated (sjdb)	0
Number of splices: GT/AG	3195740
Number of splices: GC/AG	26645
Number of splices: AT/AC	1480
Number of splices: Non-canonical	15336
Mismatch rate per base, %	0.10%
Deletion rate per base	0.00%
Deletion average length	1.60
Insertion rate per base	0.00%
Insertion average length	1.15
MULTI-MAPPING READS:	
Number of reads mapped to multiple loci	90574
% of reads mapped to multiple loci	1.08%

Number of reads mapped to too many loci | 0
% of reads mapped to too many loci | 0.00%
UNMAPPED READS:
% of reads unmapped: too many mismatches | 0.13%
% of reads unmapped: too short | 1.24%
% of reads unmapped: other | 0.01%
CHIMERIC READS:
Number of chimeric reads | 0
% of chimeric reads | 0.00%

In total, 9.1 millionen WT reads and 5.8 millionen 3xmyb reads were mapped to the Col-0 reference genome sequence. After mapping the reads, assignment and counting via featureCounts is required.

Example command to run featureCounts on mapping results:

$ featureCounts -t gene -g ID -a **Araport11_GFF3_genes_transposons.201606.gff** -o **featureCounts_results.txt Aligned.sortedByCoord.out.bam**

- -t gene: feature type used for counting reads (e.g. reads per gene are counted)
- -a: GFF3 file containing the annotation
- -o: path to place the resulting count table
- mapping file (BAM) is provided as last argument

Output:
featureCounts_results_WT.txt.summary and featureCounts_results_WT.txt

featureCounts_results_WT.txt.summary:

Status Aligned.sortedByCoord.out.bam
Assigned 7.766.889 (92,6 %)
Unassigned_Ambiguity 391.071
Unassigned_MultiMapping 182.297
Unassigned_NoFeatures 43.479
Unassigned_Unmapped 0
Unassigned_MappingQuality 0
Unassigned_FragmentLength 0
Unassigned_Chimera 0
Unassigned_Secondary 0
Unassigned_Nonjunction 0
Unassigned_Duplicate 0

STAR and featureCounts – discussion of results

- substantial differences in sample sizes after trimming
- 96.44% - 97.65% of all genes were mapped ‘uniquely’ (to only one place)
- Conclusion: technical quality of both data sets is suitable and equal

9 Statistical Analyses

9.1 Theory

- **Statistical Analyses**
  - Identification of differentially expressed genes requires statistical analyses
  - **DESeq2 (differential expression analysis for sequence count data)**
    - Established tool for statistical analyses (R package)
    - FeatureCounts output can be converted into DESeq2 input files via customized python script (contruct_DESeq2_input.py)
    - This script generates triplicates and introduces some random noise to reduce computational expenses of this course
    - FeatureCounts is available a spart of bioconductor thus providing the option to run the complete analysis in R
  - Customized python script is available for functional annotation (map_annotation.py)
    - Maps functional description to identified differentially expressed genes

9.2 Practical application

After quality control, data sets are available to downstream analyses. RNA-Seq read counting results in values which are not normally distributed but follow a Poission distribution.DESeq2 input needs to be formatted in a specific way. Therefore, count tables and additional text files need to adjusted by a customized python script which requires some information about the samples:

Structure of sample_sheets:

#Sample ID	SystematicName	SeqResult
1	Col0	wt
2	myb3x	mut

Column SeqResult needs to be edited to provide paths/names of count tables:

#Sample ID	SystematicName	SeqResult
1	Col0	**featureCounts_results_WT**
2	myb3x	**featureCounts_results_myb**

All count tables need to have the extension count_table.txt to be automatically recognized by this python script:
featureCounts_results_WT.count_table.txt
featureCounts_results_myb.count_table.txt

Replicates required by DESeq2 are generated by a random function as part of this processing step to reduce computational time in this course. Only folders are provided and this script collects all input count tables.

Example command to generate DESeq2 input files:

$ python construct_DESeq2_input.py --sample_sheet **RNA_seq_sample_sheet.csv** --sample_dir ./**featureCounts/** --output_dir ./**DESeq2_input/**

Output:
1. clean_sample_table.txt = contains meta information for DESeq2
2. clean_data_matrix.txt = contains values for DESeq2

The row names of file 1 are matching the column names of file 2. The R script used in the next step takes both files to generate a data matrix. Pseudoreplicates are generated by multiplying all count values with certain factors and introducing some noise. Therefore, no meaningful biological conclusions are possible.Next, an R script was deployed to run DESeq2 on the new files. DGE_analysis.R is edited via Rstudio to adjust the input file names.

Running DGE_analysis.R in Rstudio:

Bioconductor packages are used for analyses. First, a summary of the input data is presented:
genotype
Col0 :3
myb3x:3

Next, a summary of the count tables is shown:

Col0_1 Col0_2 Col0_3 myb3x_1 myb3x_2
Min. : 0 Min. : 0 Min. : 0 Min. : 0 Min. : 0
1st Qu.: 0 1st Qu.: 0 1st Qu.: 0 1st Qu.: 0 1st Qu.: 0
Median : 27 Median : 27 Median : 27 Median : 17 Median : 18
Mean : 1772 Mean : 2062 Mean : 2056 Mean : 1148 Mean : 1383
3rd Qu.: 192 3rd Qu.: 193 3rd Qu.: 192 3rd Qu.: 131 3rd Qu.: 131
Max. :5792000 Max. :4447000 Max. :7156000 Max. :1684000 Max. :4432000
myb3x_3
Min. : 0
1st Qu.: 0
Median : 18
Mean : 1990
3rd Qu.: 131
Max. :17791000

On the one hand, minimal value of 0 indicates that there are several genes without any assigned reads. These genes are probably not expressed at all. Since the first quantil is 0 as well, there are probably more than 25% genes without detectable expression. These genes could be discarded prior to downstream analyses as they are not helpful for the identification of differentially expressed genes.
On the other hand, RNA-Seq analysis can reveal genes with an extremely high coverage values e.g. over 17.7 million reads per gene. Therefore, it is not possible to remove PCR duplicates based on identical start and end points of reads.
While transcription factor genes usually display a low expression, important enzymes like the RuBisCO show extremely strong expression. Due to these extreme differences, box-whisker plots are a frequently applied way to display the distribution of expression values.

After removal of lowly expressed genes (<10 reads) from the initial set of 33,296 genes, 17,597 genes remain for further analysis. The total gene numbers included TEs and pseudogenes, many of which are probably discarded due to low expression if any. The remaining genes are used for a principal component analysis to assess the sample properties e.g. variance. Replicates should cluster together as they are likely to have similar expression values for all genes (Figure 8). Due to the nature of a PCA, there needs to be one principal component which is stretching the replicates in one dimension (PC1). Although the replicates of both genotypes might be similar, the 3xmyb samples are far apart in this plot (Figure 8). PCA is an easy method to identify swapped samples and other issues.

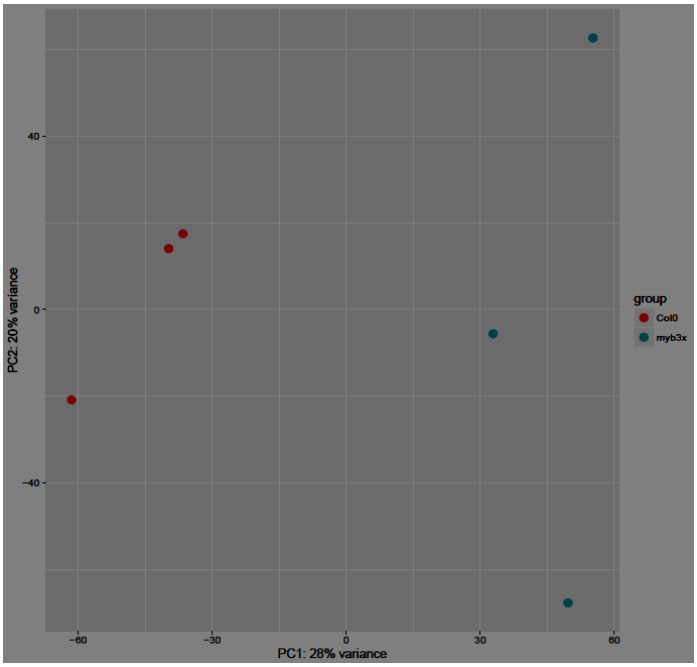


Figure 8: Principal component analysis (PCA).

Displayed is the PCA result of three replicats of both genotypes.

Next, differentially expressed genes were identified. The initial adjusted p-value cutoff was set to 0.1 to identify candidates with high sensitivity. Multiple testing requires p-value correction and stricter filtering. Therefore, the resulting genes were ranked and only the top 20 hits were inspected. Only 14 of these genes displayed an adjusted p-value below 0.1:

out of 17597 with nonzero total read count
LFC > 0 (up) : 9, 0.051%
LFC < 0 (down) : 5, 0.028%

lock fold change (LFC) = log(2)(FC)
The expression values of the same gene in the different genotypes are used to calculate this ratio.
Correction for multiple testing is automatically performed in DESeq2:

Bonferroni-Holm correction: p-value / sample size = adjusted p-value
example: 0.05/30,000 (genes) = 0.000001

DESeq2 returns the genes matching the specified criteria. These gene are collected in a text file for further analyses:
> write(sig.gene.names, DEG.txt, ncolumns=length(sig.gene.names), sep="\n")

Finally, the command sessioninfo() is applied to collect the versions of all involved R packages.

Output:

1) PDF of PCA plot
2) DEG.txt

There are 20 differentially expressed genes between both genotypes:
AT1G04700, AT1G26540, AT2G20440, AT3G22750, AT4G31720, AT1G21550, AT1G29195,

AT3G02090, AT3G19380, AT3G49760, AT4G33430, AT5G15930, AT4G14020, AT4G05170, AT1G01010, AT1G01020, AT1G01030, AT1G01040, AT1G01050, AT1G01060

Each participant identified different genes as differentially expressed due to the random effects introduced by the python script deployed for the count table generation.

Important expressions associated with gene expression values are:
TPMs = tags per million assigned tags (tags= reads or fragments)
Values are only normalized to sample size.

RPKMs = reads per kb exon per million mapped reads (single end)
Values are normalized to sample size and transcript length.
((reads per gene)/((gene length [kb]/1 kb)*(total number of reads/1.000.000))

FPKMs = fragments per kb exon per million mapped fragments (paired-end)
Values are normalized to sample size and transcript length.

Major challenge during normalization is the correction for the transcript length. Although longer transcripts are more likely to result in sequenced cDNAs, their chance of being represented does not scale linearly with their length. As a result, the use or RPKMs/FPKMs should be avoided if possible. Just using TPMs for investigations is restricting analyses to comparisons of one gene in different genotypes / under different conditions.

After identifying differentially expressed genes, they were annotated based on Araport11 using a customized python script.

Check usage of script:

$ python map_annotation.py

Example command to add annotation to identified DEGs:

$ python map_annotation.py --input_file **differentially_expressed_genes.txt**

Output in terminal:
number of entries in mapping table: 33602
number of annotated genes: 20

Output:
differentially_expressed_genes.txt_annotated.txt

The output of this scripts looks as follows:
AT1G04700 PB1 domain-containing protein tyrosine kinase
AT1G26540 Agenet domain-containing protein
AT2G20440 Ypt/Rab-GAP domain of gyp1p superfamily protein
AT3G22750 Protein kinase superfamily protein
AT4G31720 TBP-associated factor II 15
AT1G21550 Calcium-binding EF-hand family protein
AT1G29195 unknown protein

AT3G02090	Insulinase (Peptidase family M16) protein
AT3G19380	plant U-box 25
AT3G49760	basic leucine-zipper 5
AT4G33430	BRI1-associated receptor kinase
AT5G15930	plant adhesion molecule 1
AT4G14020	Rapid alkalinization factor (RALF) family protein
AT4G05170	**basic helix-loop-helix (bHLH) DNA-binding superfamily protein**
AT1G01010	NAC domain containing protein 1
AT1G01020	Arv1-like protein
AT1G01030	AP2/B3-like transcriptional factor family protein
AT1G01040	dicer-like 1
AT1G01050	pyrophosphorylase 1
AT1G01060	Homeodomain-like superfamily protein

QUESTION: Which methods allow a validation of these results?

1) qRT-PCR
2) semi quantitative RT-PCR
3) mircoarray
4) reporter gene fusion
5) Yeast two hybrid or yeast one hybrid
6) transient protoplast transfection