F1-internship bioinformatics

Diarmaid Tobin
Master of Science, Biology
Matr.Nr.: 1755928
Institute for Molecular Infection Biology (IMIB)
Julius-Maximilians-Universitaet Wuerzburg

03. September 2015 - 09. October 2015

Contents

Motivation

During my biology studies in Wuerzburg I focused on microbiology as a main topic. Here I acquired theoretical knowledge and wet lab experience. A lecture in the second Master semester then dealt with bioinformatics. It made me realize that my knowledge about this topic is rather poor. It encouraged me to take part in the lecture 'Systems Biology', which confirmed my resolution to get a closer insight. Finally, it lead to this internship and the aim to learn basics about programming and get a better idea of operating principles in a bioinformatical working group.

1 Working environment

1.1 GNU/Linux

GNU/Linux is an Operating System (OS) consisting of a Linux kernel and a variety of GNU software packages. It derived from the original AT&T Unix, developed in the 1970s at Bell Labs research center [1]. Originally Unix was meant to be a programmer's platform for developing software. The OS eventually started spreading in academic circles where users added own tools to the system and customized it for their purpose. Over time this ongoing process resulted in the development and distribution of Unix-based GNU/Linux OS including several different distributions. For this internship the Debian-based distribution Ubuntu 15.04 was used. Using it over other OS, i.e. Windows, bears several advantages. For example, GNU/Linux and the majority of available programs are licensed under free software licenses with the goal of making computers available to everyone without restrictions of licenses and copyrights. Furthermore GNU/Linux is open-source, enabling the large community using it to fix bugs or improve certain aspects and also low-level software like the kernel up to graphical user interface (GUI) themes can be customized. The OS also supports more architectures and filesystems than comparable ones increasing the flexibility of users. Additionally GNU/Linux is considered as highly secure and suffers from very little malware. On the other hand having the power to configure a system and actually doing so requires a certain level of knowledge which results in a rather steep learning curve for new users. Still classical Unix-tools follow a philosophy which can be roughly described as KISS ('Keep it short and simple'). Those features make Unix a popular platform among the bioinformatical sector.

1.2 Bourne-again shell

Bourne-again shell (Bash) is a Unix shell and command language written for the GNU project [2]. As a command line interpreter it enables users to move through their system and manage practically any task with predefined command lines. Useful command lines are shown in Table 1 and several shortcuts being used in Table 2. For instance files/folders can be moved/changed throughout the system and between different systems/servers, documents/-files can be downloaded, (de-)compressed, moved or used as input for Unix tools. Command lines need a command and some programs additionally require parameters and/or arguments. (Simplified: The command is the action performed, a parameter is some sort of condition and the argument stands for the file/document on which the action takes place). The result of commands are usually stored within the *standard output* of the shell. To avoid unnecessary repition of tasks and accumulation of needless data-files on your system, command lines can be combined within shell scripts, whereat the *standard output* of a command serves as the *standard input* for the next command ('piping'). Once a shell script is launched within the shell, the different tasks get executed in the given order. Such scripts also minimize the risk of doing mistakes, as it can be checked on correct functionality and further used as a standard. Furthermore a certain level of reproducibility is given through this procedure as the single tasks are recorded within the script. Shell scripts can also be combined with other scripts, i.e. python. This enables one to write powerful bioinformatical tools to automatically perform otherwise manual tasks.

1.3 Secure Shell

Secure Shell (SSH) is a cryptographic network protocol to allow remote login in other networks/servers. This is often a prerequisite in bioinformatics as some programs need a lot capacity to handle a certain amount of data [3]. In this case SSH was used to create a secure channel between a SSH client and a SSH server through the bash. With following command using the software OpenSSH a public-private key pair to encrypt the network connection was generated:

```
$ ssh -keygen -t rsa -b 4096
```

The files containing the keys were saved locally in the directory '/.ssh', then a password was assigned to the private key. The private key was deposited on the local computer, the public key was deposited onto the server.

Command	**Action**
(sudo) apt-get (install)	Search for and install package(s)
bash	Execute shell script
bzip2/gzip	Compress data file
cat	Concatenate and print content of file(s)
cp/scp	Copy file/Secure copy
gedit	Gnome text editor
grep	Search lines that match a given pattern
head/tail	Output first/last part of file(s)
htop	Interactive process viewer
ls	List content
mkdir	Create new folder
mv	Move or rename file/directory
open	Open file(s)
pv	Monitor progress of data through a pipe
pwd	Print Working Directory
rename	Rename file
rm/rmdir	Remove file/folder
sort	Sort text file
touch	Change file timestamps
wget	Retrieve website/file via HTTP(S)/FTP

Table 1: Useful Bash command lines

Shortcut	**Meaning**
./	Current directory
../	Parent directory
$\sim$	Home directory
*	Substitutes any letter/number
Tab key	Auto-complete
|	Pipe

Table 2: Useful Bash shortcuts

In order to do so the connection with the server had to be created by typing following into bash (note: The IP address of the server is required):

```
$ ssh diarmaid@132.187.174.20
```

After login with a password, a folder had to be created ('mkdir') in which the public key was deposited. Following command executed locally was used to write the public key to the specified file and save it within the created folder '.ssh':

```
$ cat id_rsa.pub | ssh diarmaid@132.187.174.20 | \
'cat >> ~/.ssh/authorized_keys'
```

At this point, when logging on the server following procedure takes place: The server sends a randomly generated "challenge" to the local computer, which encrypts the challenge by the private key. Once the server is able to decrypt the encrypted challenge by using the public key, connection is established. Finally, a config file was created on the local system to login to the server without typing username or IP address:

```
Host Hawaii
        HostName 132.187.174.20
        User diarmaid
```

By creating the config file the command line for logging on to the server became:

```
$ ssh hawaii
```

A helpful feature for working on a server is tmux, a software application that can be used to multiplex several virtual consoles. This enables users to work on multiple terminal sessions within a single terminal window [4]. Once a process is started in tmux, the program can be closed and the server connection be terminated, with the process still running. A session can be started by typing 'tmux', and a closed session restarted by typing 'tmux a' into the bash. By hitting 'ctrl' and 'b' followed by a command tasks can be executed within tmux. Some helpful commands are listed in Table 3.

1.4 Markdown

Markdown is a so called 'markup language' with plain text formatting syntax [5]. Markdown documents (written with plain text editors i.e. GNU Emacs and the file extension .md) can be converted to other formats like HTML or PDF through a Markdown-tool. With word processors using formatted text usually wrapping binaries around the text or passages, converting to other formats may be problematic. In Markdown this problem is solved

Command	Action
c	create window
w	list windows
n/p	next/previous window
,	name window
&	kill window

Table 3: Useful tmux shortcuts

by using plain text 'tags' to add information to text/passages. The software is free, platform independent and relatively easy to learn. Drawbacks of Markdown might be a missing spellcheck or little layout options.

1.5 LaTeX

LaTeX is a word processor and a document markup language [6] which features functions similar to Markdown. In contrast to other word processors documents are written in plain text whereas the conversion relies on the markup tagging. The program originally was designed by physicians/mathematicians. Therefore mathematical notation and illustration (i.e. of graphs) is quite sophisticated and also the handling of quotations can be done intraformat with BibTex. Like Markdown, LaTeX is free and the user can focus on content taking care of the layout later but the possibilities regarding layout are greater. Disadvantages might be the missing spellcheck and the rather difficult handling of the program in the beginning.

1.6 Git

Git is a free version control software which was initially designed to version source codes. The software stores different versions of files including a timestamp within a local repository. Assuming a new version of ones document gets lost, the backup created through git can be restored. Another practical application of git is the highlighting of committed changes within a document by comparing the new with old versions. This is especially useful and timesaving when tracking down mistakes within programming scripts as short plain text files can be versioned without any problems. In contrast, binary files may not be versioned by git and versioning big files is space consuming for ones hard drive.

To create a local repository following command has to be executed within the directory containing the files a user wants to version:

```
$ git init
```

In order to only version certain files a .ignore file can be created containing any file that should not be versioned. Thereby certain types of files as a whole (i.e. *.py) or specific files can be excluded from being versioned. To add files and commit them to the repository following commands have to be executed:

```
$ git add <filename>
$ git commit –m "optional"
```

In order to highlight differences between two versions of a file following command has to be executed:

```
$ git diff <filename1(i.e. backup)> <filename2>
```

1.7 GitHub

GitHub is a web-based git repository hosting service which features distributed version control and source code management functionality of git. Additionally it includes own features like access control or collaboration features (i.e. the possibility to merge different versions of one document or bundling/adding information from different users within one document). Users can create public or private repositories for free and upload their data. In case of public repositories anybody has access to the uploaded information whereas private repositories require access rights. Especially in science systems like GitHub are reasonable as they minimize time consumption of processes like the creation of a paper through different authors. Documents can be interactively changed by variable persons who have exclusive access to the particular repositories. Additionally changes made within documents are linked to the person who committed the change providing clarification. To commit changes in a GitHub repository by a different user it has to be forked and cloned to the user's local system. Forking means to copy the repository and providing it to the other user who then can clone it to the local system. After committing changes the other user can send a pull request to the creator of the GitHub repository. The changes will be committed to the original document(s), once the pull request is accepted. In order to manage a GitHub repository in the internship an empty repository was created. After submitting a SSH-key the SSH-URL from the repository was used to add the remote GitHub repository to the local one as following:

```
$ git remote add origin <SSH_URL>
```

At this point, files could have been changed locally and added to the GitHub repository by following command:

```
$ git push origin master
```

In order to add changes made by a different person and added to the users repository, the changed version can be merged with the original document by following command:

```
$ git pull origin master
```

1.8 Python

Python is a general-purpose programming language which enables scripting programs on small and large scales [7]. Compared to other programming languages (C++/Java) python's design focuses on readability, its syntax allows fewer lines of code and is rather easy to learn. Multiple programming paradigms are supported, inter alia functional programming (1.8.1) and object-oriented (1.8.2) but also plotting of data is possible. Those and further functions are enabled through a variety of available libraries such as matplotlib or pandas. In this internship predominantly RNA-Seq reads mapped against reference genomes were analyzed. Those information were stored within certain file formats (i.e. FASTA) in form of lists, whereat every mapped gene makes up two lines, one containing general information (i.e. length or position), the other containing the sequence. To extract information from this kind of file with the help of python scripts some general principles are useful. When it comes to lists especially iteration is useful, which means line after line is checked for the information specified in the script. When looking for total numbers basic mathematical functions like giving out the sum of counts are possible. Some python functions are designed to handle only integers or strings so sometimes it is useful to convert strings to integers and vice versa. Those and much more possibilities of python make it a powerful and popular tool within the bioinformatical sector.

1.8.1 Functional programming

Functional programming has its origin in the 1930's research regarding the Lambda calculus, a formal system in mathematical logic for expressing computation [8, 9]. A python script written following this paradigm predominantly contains different functions in a row which fulfill specific tasks, i.e. iterating through an file and adding numbers of a certain content. Usually functional programs are stateless, which means functions are executed independently of what has happened earlier in the programs execution. Another core principle is functional programs being immutable resulting in thread-safe objects which is important for multi-threaded programs [10]. Functional

programs are rather suited for specific tasks and less for repetitive jobs with altering input data.

1.8.2 Object-orientated programming

This paradigm follows the approach of identifying all objects of one class to be manipulated and how they relate to each other ('data modeling') resulting in the definition of subclasses. Similar to functional programming, a python script written object-orientated contains functions, which are called methods. Methods, as well as attributes are connected to objects, which then represent instances of a class ('inheritance') [11]. Those objects can be seen as independent functional blocks, which execute the defined action, whereas the execution only takes place through calling the class. Once a object is not required it can be simply deactivated within the class and also new objects can be added without interfering with other functionalities within the script. So unlike the functional approach the object-orientated paradigm is suitable for multiple use with altering input data.
In order to analyze a BAM file following python script with a object-orientated programming approach was scripted:

```
import argparse
import pysam
import matplotlib
import matplotlib.pyplot as plt

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("bamfile", help = "input BAM file")
    parser.add_argument("output", help = "Name of your \
                                output file in csv format")
    parser.add_argument("outputgraph1", help = "Name of \
                your graphical output file[mapping/read]")
    parser.add_argument("outputgraph2", help = "Name of \
               your graphical output file[mappinglengths")
    args=parser.parse_args()

    bamstats = BAMstats(args.bamfile, args.output, \
                args.outputgraph1, args.outputgraph2)
    bamstats.nr_aligned_reads()
    bamstats.uniquely_aligned_reads()
    bamstats.nr_mapping_per_strand()
    bamstats.nr_refseq()
```

```
    bamstats.read_per_ref()
    bamstats.hist_nr_mappings_per_read()
    bamstats.hist_of_mapping_lengths()


class BAMstats():
    def __init__(self, input_file, output_file, \
                 output_graph1, output_graph2):
        self.bamfile = input_file
        self.output = output_file
        self.outputgraph1 = output_graph1
        self.outputgraph2 = output_graph2

    def nr_aligned_reads(self):
        bamfile = pysam.AlignmentFile(self.bamfile, "rb")
        count =len(list(bamfile.fetch("NC_000915.1")))
        output=open(self.output, "a")
        output.write("Nr of reads:\t{}".format(count))

    def uniquely_aligned_reads(self):
        bamfile=pysam.AlignmentFile(self.bamfile, "rb")
        output=open(self.output, "a")
        unique_reads = {"nr_reads" : 0}
        for read in bamfile.fetch():
            for tag in read.tags:
                if tag[0] == 'NH':
                    if int(tag[1]) == 1:
                        unique_reads["nr_reads"] += 1
                    else:
                        continue
        output.write("\nNr of uniquely mapped reads: \
               \t{}".format(unique_reads["nr_reads"]))

    def nr_mapping_per_strand(self):
        bamfile=pysam.AlignmentFile(self.bamfile, "rb")
        output=open(self.output, "a")
        count_read_rev = 0
        count_read_fwd = 0
        for read in bamfile.fetch():
            if read.is_reverse:
                count_read_rev += 1
```

```
            else:
                count_read_fwd += 1
        output.write("\nNr of reads on rev strand: \
                      \t{}".format(count_read_rev))
        output.write("\nNr of reads on fwd strand: \
                      \t{}".format(count_read_fwd))

    def nr_refseq(self):
        bamfile=pysam.AlignmentFile(self.bamfile, "rb")
        output=open(self.output, "a")
        output.write("\nNr of reference sequences: \
                  \t{}".format(bamfile.nreferences))

    def read_per_ref(self):
        bamfile=pysam.AlignmentFile(self.bamfile, "rb")
        output=open(self.output, "a")

        for reference in bamfile.references:
            ref_nr = 0
            for liste in bamfile.fetch(reference):
                ref_nr += 1
        output.write("\nRefID:\t{}\nNr of reads per \
           reference:\t{}".format(reference, ref_nr))

    def hist_nr_mappings_per_read(self):
        bamfile = pysam.AlignmentFile(self.bamfile, "rb")
        output = open(self.outputgraph1, "a")
        nr_mapped_reads = {}
        for read in bamfile.fetch():
            for tag in read.tags:
                if tag[0] == "NH":
                    nr_mapped_reads.setdefault(tag[1], 0)
                    nr_mapped_reads[tag[1]] += (1 / tag[1])
                else:
                    continue
        fig = plt.figure()
        ax = fig.add_subplot(1, 1, 1,)
        ax.bar(range(len(nr_mapped_reads)), \
                  nr_mapped_reads.values(), align="center")
        ax.set_xticks(range(len(nr_mapped_reads)))
        ax.set_xticklabels(sorted(list(nr_mapped_reads.keys())))
```

```
        plt.savefig(output)

    def hist_of_mapping_lengths(self):
        bamfile = pysam.AlignmentFile(self.bamfile, "rb")
        output = open(self.outputgraph2, "a")
        lenghts_frequencies = {}
        for read in bamfile.fetch():
            lenghts_frequencies.setdefault(len\
                        (read.query_alignment_sequence), 0)
            lenghts_frequencies[len(read.query_alignment_sequence)] \
                                                              +=1

        fig = plt.figure()
        ax = fig.add_subplot(1, 1, 1,)

        ax.bar(range(len(lenghts_frequencies)), \
                lenghts_frequencies.values(), align="center")
        ax.set_xticks(range(len(lenghts_frequencies)))
        ax.set_xticklabels(sorted(list(lenghts_frequencies.keys())))
        plt.savefig(output)

main()
```

1.9 R

R is a programming language especially designed for statistical analysis. Unlike python, R commands are rather specific than general, therefore the language is harder to learn but very effective when it comes to statistical evaluation and its presentation. Many bioinformatical tools are based on this programming language, i.e. Bioconductor's package DESeq2. This makes basic knowledge about R nearly unevitable when working in the bioinformatical sector.

1.10 Jupyter-notebook

A useful application for programmers may be the free server-client Jupyter-notebook. It allows editing and running notebook documents via web browser depending on implemented kernels, i.e. ipython- or R-kernel. Codes can be written and executed whereas results are shown instantly. When it comes to python programming the system allows the different libraries available for

python. Jupyter-notebook documents are being saved locally on the system as plain text files which makes their handling easy. The different files can be called within the web application from the local system for further usage. The advantages of this application are the possibilities using a variety of kernels and the immediate output of the written code including plots.

1.11 Licenses

The basic idea of science is to gather and share knowledge so anybody can build on this given basis and ideally advance know-how. Despite of this idea being practically impossible due to human nature, anyone who publishes information on the internet per default holds complete copyright on it. Because of this, there is need for a system to assign published data with certain copyright regulations. One approach is provided by the non-profit organization Creative Commons (CC). It provides six so called Creative Commons licenses, which state under which conditions third parties can further use published data. The available licenses are a combination of the four modules 'Attribution' (BY), 'Non-commercial' (NC), 'No Derivate Works' (ND) and 'Share-alike' (SA). BY allows copying, distribution, displaying and performance of the work, as well as derivates of it, if specified credits are given to the licensor. NC allows copying, distribution, displaying and performance of the work, as well as derivates of it, if the purpose is noncommercial. ND states that only verbatim copies of the work may be copied, distributed, displayed and performed. SA only allows distribution of derivative works, if licensed identical to the original work. Additionally one license containing none of the moduls and making published data public domain exists [12]. Publish data under licenses that encourage distribution does not only increase efficiency in science but also reproducibility of i.e. experiments. Reproducibility is a critical point in science, not only because false data may be published, but also because of an objective, third party verification of published Data. This may lead to correction and therefore more profound results which is beneficial for anybody involved [13].

1.12 Organizing bioinformatical projects

Another keypoint of making ones data set reproducible is the organization of it. This organization should follow the principle of someone unfamiliar with a project being able to understand what and why it was done by looking at the files referring to the project. To achieve this, a system to organize files and directories is useful and will be described in this section exemplary. First a root directory of different projects may be created, containing five differ-

ent directories (bin, data, doc, results, src). In 'doc' any kind of textfiles containing information about the project, for instance papers or protocols, are stored. The 'bin'-directory contains scripts and downloads, for instance regarding downloaded programs or compiled binaries, whereas 'src' contains only source codes. Both, the 'data' and 'results'-directories further contain directories labeled with the date of creation to get a chronological order of the projects. 'data' contains input data and 'results' the output data of the different projects. The exact organization of ones projects depends one personal taste, but the principle idea of organizing at all and especially chronological as well as logical directory organization is very useful to reproduce projects [14].

2 Mapping, Annotation and Quantification of RNA-seq data

2.1 Theoretical background

In recent years, a lot effort has been made to improve existing and develop new high-throughput sequencing (HTS) technologies. Beside established platforms like Roche 454, Sanger or IonTorrent and the rather new introduced Pacific Biosciences or Oxford Nanopore technologies, Illumina is currently market leader within the HTS sector. Its systems can create an output of up to 1.8 Tb in 3 days which equals the sequencing of 18.000 human genomes at 30x coverage per year [15]. By creating cDNA libraries HTS technologies can also be applied on RNA, which enables massive parallel sequencing of whole either procaryotic or eucaryotic transcriptomes ('RNA-Seq'). Thereby RNA-Seq provides a comprehensive picture of the transcription quantity and the structure of transcripts [16, 17, 18]. Compared to traditional methods, i.e. microarrays, RNA-Seq reveals an increased dynamic range and sensitivity of sequence data, as well as regions with high sequence similarity can be better discriminated [19]. By aligning RNA-Seq reads against reference genomes the quality of Mapping, Annotation and Quantification was improved and also new insights on whole transcriptomes of organisms were provided. Contrary to the original assumption of the bacterial transcriptome and its regulation being simple, RNA-Seq played an important role in the discovery of it being quite diverse with i.e. various RNA species, antisense transcription or riboswitches [20, 21, 22]. Meanwhile RNA-Seq is very popular within ongoing microbiological researches and also serves as a basis for more sensible methods like differential RNA-seq (dRNA-seq), which allows the discrimination of primary and processed transcripts. The method relays on the fact,

that primary transcripts reveal a 5' tri-phosphate group (5'PPP), but processed transcripts a 5' mono-phosphate (5'P). By splitting one bacterial RNA pool into two and treating one of them with terminator exonuclease, which degrades 5'P, a pool of total bacterial RNA and one enriched for primary transcripts is generated. This can also be applied on different bacterial samples grown under various conditions. In any case a cDNA library for each of the different pools is created, followed by sequencing of the libraries [23]. This technique already has led to new insights in the genetic property of pathogens like *Helicobacter pylori* or *Vibrio cholerae*, and will likely be further used in the future for a more precise annotation and quantification of bacterial genomes [24].
Following sections cover the process of gaining RNA-Seq read data, the corresponding reference sequences and the alignment of reads against the reference genome (2.1). The read data being used were generated in context of a dRNA-Seq approach for annotating transcriptional starting sites in *H. pylori* [25]. In section 2.2 exemplary steps of annotating a bacterial genome with bioinformatical methods is shown, using part of read data generated in another dRNA-Seq approach [26], namely sample SRR1602507, SRR1602509, SRR1602511 and SRR1602513. Section 2.3 displays an exemplary way of quantifying RNA-Seq data using the same data as in section 2.2. The reference sequences were chosen according to the publications describing the generation of the RNA-Seq data.

2.2 Alignment/Mapping

The alignment or mapping of generated HTS reads against a reference genome is fundamental, as it serves as a basis for ongoing HTS analysis. As most of the currently used sequencing technologies generate short reads, mapping can become quite challenging dependent on genome-size and sequencing-depth. Apart from processing potentially huge data sets, mapping programs face different challenges, which should encourage to choose the suitable program among the wide choice [27, 28, 29].
For this internship two mappers were used, Bowtie 2 version 2.2.6 (2.2.2) and Segemehl version 0.2.0 (2.2.3). Prior to mapping, the RNA-Seq data had to be acquired, checked on quality and edited using different programs (2.2.1).

2.2.1 Preparing HTS reads

First, the downloaded Sequence Read Archive (.sra) files were transformed to FASTQ files through the program fastq-dump from SRA toolkit version 2.3.5. FASTQ files contain similar to FASTA files information about the

sequence in addition to its per-letter Phred quality score. With the program FastQC version 0.11.2 the reads' attributes according to quality were visualized to get a first impression of the overall read-quality (Fig. 1).

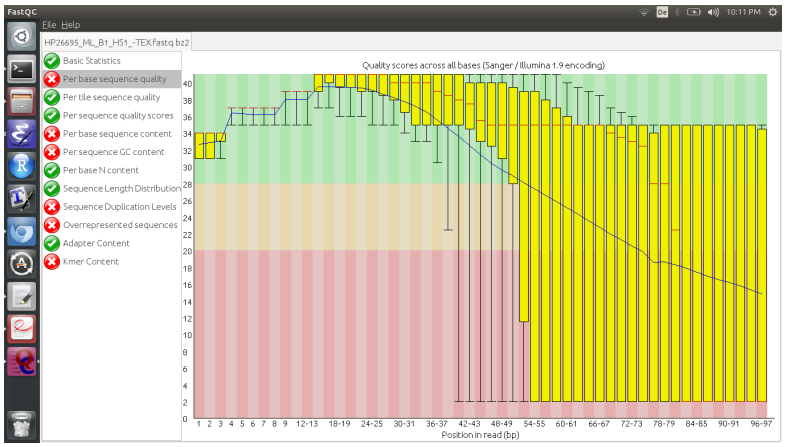

Figure 1: FastQC interface, displaying the per base sequence quality.

Subsequent to checking the quality, the reads were quality trimmed by the program fastq_quality_trimmer from the FASTX-Toolkit version 0.11.13. The cut-off Phred score of 20 which equals a base call accuracy of 99The shell script being used to prepare the data for mapping:

```
INPUT_FOLDER = <defined input folder>
OUTPUT_FOLDER = <defined output folder>

mkdir -p $INPUT_FOLDER
mkdir -p $OUTPUT_FOLDER

wget -P $INPUT_FOLDER <file URL1> \
                      <file URL2> \
                      <file URL3>


for FILE in $(ls $INPUT_FOLDER/*.sra)
do
    fastq-dump.2.3.5 --stdout $FILE | \
    ../../bin/bin_fastx/fastq_quality_trimmer -t20 -l10 -Q33 | \
    ../../bin/bin_fastx/fastq_to_fasta -Q33 | \
    python3 <Pythonscript>.py | \
```

```
    bzip2 > $OUTPUT_FOLDER/$(basename $FILE .fastq.bz2).fasta.bz2
done
```

The content of the python script, which is embedded within the shell script:

```
import sys

for line in sys.stdin:
    if line.startswith(">"):
        nextline = next(sys.stdin)
        if nextline.find("AAAAAAAAAA") > -1:
            cut_off = nextline.find("AAAAAAAAAA")
            seq = nextline[0:cut_off]+"\n"
        else:
            seq = nextline
        if len(seq) > 1:
            sys.stdout.write(line)
            sys.stdout.write(seq)
    else:
        sys.stderr.write("Error\n")
```

The result were bzipped, quality- and adapter-trimmed FASTA files which could serve as input for the mapping programs.

2.2.2 Bowtie 2

The mapper Bowtie 2 combines full-text minute index and SIMD accelerated dynamic programming to find gapped alignment with comperable little memory footprint [30]. According to the Bowtie 2 manual first the reference genome has to be indexed. To be processed by Bowtie 2, the file extension of the reference genome had to be changed from .fna to .fa with following command:

```
$ mv <reference genome>.fna <reference genome>.fa
```

Then indexing of the reference genome could take place:

```
$ <path>/bowtie2-build -f <reference sequence> HP_Index
```

The option '-f' states that the input file is in FASTA format. For the alignment following shell script was used:

```
INPUT_FOLDER = <input folder>
OUTPUT_FOLDER = <output folder>
```

```
export BOWTIE2_INDEXES=$OUTPUT_FOLDER

for FILE in $(ls $INPUT_FOLDER/*.bz2)
do
    <path>/bowtie2-2.2.6/bowtie2 \
        -f \
        -x HP_Index \
        -U $FILE \
        -S $OUTPUT_FOLDER/$(basename $FILE .fasta.bz2).sam
done
```

The argument '-U' specifies the folder containing the reads to be aligned, '-S' specifies the folder in which output files in SAM format are deposited in. The option '-x' specifies the Index file, the option '-f' states that reads are in FASTA format.

2.2.3 Segemehl

Segemehl is an alignment tool with advantages when it comes to multi split alignments. Compared to other mappers its demand of memory footprint is rather high but the performance in finding multiple hits shows better quality [31]. According to the Segemehl manual, first the reference genome was indexed. As Segemehl does not accept bzip2 compressed read files they had to be converted to gzip compressed files. Then the alignment process was executed. The described process is captured in following shell script:

```
INPUT_FOLDER = <input folder>
OUTPUT_FOLDER = <output folder>
<path>/ segemehl / segemehl.x \
-x<reference genome>.idx -d<reference genome>.fa
for FILE in $(ls $INPUT_FOLDER/*.bz2)
do
bunzip2 -c -d $FILE | \
gzip -v9 \
> $OUTPUT_FOLDER/$ (basename $FILE .bz2).gz
done

OUTPUT_FOLDER2 = <output folder 2>
mkdir -p $OUTPUT_FOLDER2
for FILE2 in $(ls $INPUT_FOLDER/*.gz)
do
<path>/ segemehl / segemehl . x \
```

```
-i<reference genome>.idx -d<reference genome>.fa -q $FILE2 \
                                                  --threads 2 \
> $OUTPUT_FOLDER2/$ (basename $FILE2.fa).sam
done
```

Both mapping programs create a file in Sequence Alignment/Map (SAM) format. SAM files are human readable and consist of the alignment section and an optional header section if present starting with '@' and prior to the alignments. In order to minimize memory footprint and also visualizing the alignments later, the SAM files were converted to the Binary Alignment/Map(BAM) format by using the program Samtools version 0.1.19 within following shell script:

```
INPUT_FOLDER = <input folder>
OUTPUT_FOLDER = <output folder>

mkdir $OUTPUT_FOLDER

for FILE in $(ls $INPUT_FOLDER/*.sam)
do
    BAMFILE=$(basename $FILE .sam).bam
    SORTED_BAM=$(basename $BAMFILE .bam)_sorted.bam

    samtools view -bS $FILE > $OUTPUT_FOLDER/$BAMFILE
    samtools sort -f $OUTPUT_FOLDER/$BAMFILE \
                          $OUTPUT_FOLDER/$SORTED_BAM
    samtools index $OUTPUT_FOLDER/$SORTED_BAM

done
```

The Samtool command 'view' prints the alignment to the specified output folder in BAM format because of the option '-b' (the 'S' option states input is in SAM format). The command 'sort' sorts alignments by leftmost coordinates, with the option '-f' the suffix .bam is not automatically added to the output file. The command 'index' finally indexes the BAM files.
Once converted to BAM format, the alignments can be visualized through tools like Integrated Genome Browser (IGB) or Integrative Genomics Viewer (IGV). Figure 2 exemplary shows mapping results visualized by IGB.

2.3 Genome Annotation

In general, genome annotation can be subdivided into two different processes, structural and functional genome annotation. Structural genome annotation

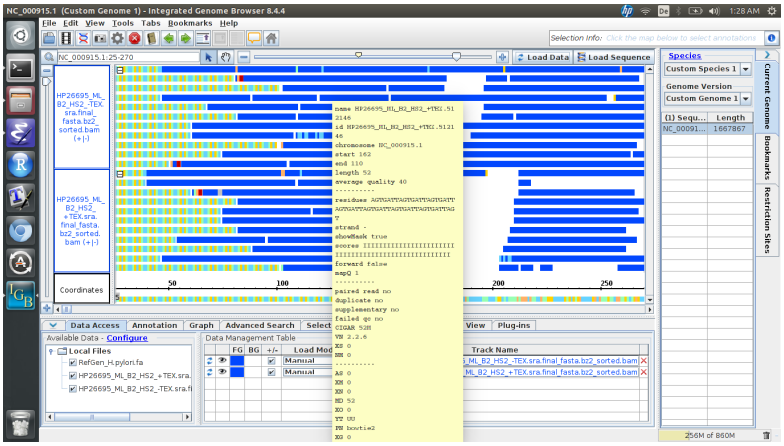

Figure 2: IGB showing the mapped processed reads against the reference genome.

describes identifying genes including their intron-exon structures, whereas functional genome annotation is assigning the functionalities to structural annotated genes [32]. Traditional approaches rather rely on knowledge gathered through experiments or the search for known genetic structures, such as the Pribnow-Box in Promotor regions of bacteria or bacteriophages. Since the possibility of analyzing whole transcriptomes of organisms through RNA-seq other approaches were introduced, i.e. dRNA-seq. With this method being established transcription starting sites (TSS) of RNA and also antisense RNA (asRNA) can be traced more accurate than before. To accomplish this a tool called TSSpredator has been developed to detect and classify TSS of dRNA-Seq data [33].
The internship covered the process of preparing dRNA-Seq data in order to analyze it through the TSSpredator version 1.04. To gain datasets suitable as input for TSS Predator the RNA-Seq processing pipeline READemption was used. Similar to section 2.2 READemption provides data preparation and mapping through the implemented program segemehl resulting in alignment BAM files and mapping statistics. Based on the preceding alignment and different normalization methods, cDNA coverage files in wiggle track format (WIG) can be created with the pipeline and are further used for TSS prediction. [34]. WIG files contain quantitative information about single-base read coverage of aligned RNA-Seq reads which is necessary to perform TSS prediction. According to the READemption documentation the tool was acquired and installed. Then the file extension of the FASTA file containing the reference genome was changed to .fa. In order to match the ID of downloaded annotation files (in GFF3 format) the header of the FASTA file was

renamed by following command:

```
$ sed -i "s/>/> <ID file> /" <reference genome>.fa
```

Following shell script covers the steps of creating the necessary folder structure, moving input files into the respective folders, as well as performing the alignment and coverage calculation with the WIG files as output:

```
main(){
        create_folder
        load_input
        alignment
        coverage_calculation
}

create_folder(){
        reademption create READemption_analysis
}

load_input(){
       cp <path>/*.fasta <path>/READemption_analysis/input/reads
       cp <path>/*.gff <path>/READemption_analysis/input/annotations
       cp <path>/*.fa <path>/READemption_analysis/input/ \
       reference_sequences
}

alignment(){
reademption align -p2 -- poly_a_clipping READemption_analysis
}

coverage_calculation(){
reademption coverage -p2 READemption_analysis
}
main
```

To use the program TSSpredator it has to be downloaded and unzipped. To use its graphical interface Java must be installed. The TSSpredator can either be run on a server or the local system. When running on a server the option -X is added to enable the program's graphical interface on the local computer. Following command executes the TSSpredator with the graphical interface on the local system:

```
$ java -jar <path>/TSSPredator.jar
```

In order to execute the TSSpredator on the server hawaii with a local graphical interface following command is required:

```
$ ssh -X hawaii
$ java -jar <path>/TSSPredator.jar
```

As input for TSSpredator the reference genome (.fa), the created coverage files (.wig) and the downloaded annotation files (.gff) were used. The output file 'superTSS.gff' was further used but first the name within the file must have been changed from 'super' to the reference ID. To rename following was done:

```
$ sed -i "s/super/NC_000915.1/" superTSS.gff
```

At this point the TSS prediction output could have been loaded to IGB to visualize the findings. Optional the annotated reference genome and the coverage files can be loaded to see if TSS were annotated correctly or if previous annotations may not have been correct. Also the coverage files can be loaded if required (Fig. 3).

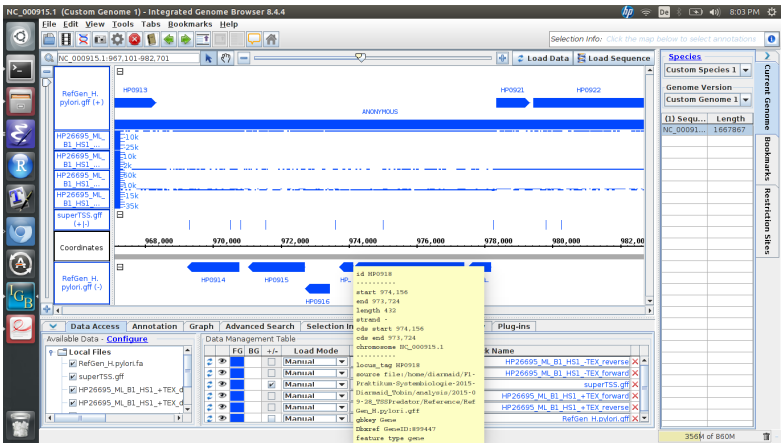

Figure 3: IGB showing the result of TSSpredator including the annotated reference genome, as well as coverage files created before with READemption.

2.4 Quantification

RNA-Seq has not only been a basis for new alignment and annotation approaches, but has also led to the evolvement of different quantitative assays [35, 36]. One of those quantitative approaches is called differential gene expression (DGE) analysis of RNA-Seq data. Whit mapping RNA-Seq reads are assigned to the corresponding area of the reference genome, ideally followed by annotation of those areas. The amount of reads aligned to one

genomic area is called 'read count'. DGE involves the comparison of read counts obtained from different biological samples, i.e. two samples of one bacterial strain representing different growth phases [37]. The fact that usually a small number of replicates is used for RNA-Seq and that the whole method underlays biological and technical variation has led to discussion [38, 39, 40]. The variance may be even greater when quantitative analysis of different samples/conditions is performed. Also non-normality of read counts illustrates a problem once different read counts of the same genes are being compared. One possible method to face those problems is DESeq2 [41]. DEseq2 offers the normalization of read counts which is a result of variable sequencing depths. Furthermore variances inter- and intra-samples are regarded. Also multiple testing adjustment is performed, which addresses the problem of p values of a set of genes is often higher than for the individual one, resulting in a adjusted p value (padj). In this case, every padj > 0.05 is discarded from the result. The aim of the internship was to analyze DGE data with DEseQ2 and create a MA plot based on the results. In general MA plots compare the log ratio of two variables and the mean values of the same variables in order to detect dependencies. In this case the x-axis of the MA plot equals the mean normalized count of the conditions. The y-axis represents the log2 fold change which measures up-/down-regulation of gene expression. A log2 fold change of 1 means that gene expression has roughly doubled, whereas a log2 fold change of -1 represents the decrease of gene expression to roughly the half.
For the exemplary execution RNA-Seq data of two bacterial samples of the same species gathered from different growth phases (optical density (OD) 0.1 and 0.2) were used. The analysis through DEseq2 was performed with the command 'reademption deseq' taken from the READemption pipeline. The command looked as following:

```
$ reademption deseq \
    -l <sample1_plus>.fa.bz2,<sample1_minus>.fa.bz2 \
       <sample2_plus>.fa.bz2,<sample2_minus>.fa.bz2 \
    -c <OD 0.1>,<OD 2>,<OD 0.1>,<OD 2>
```

Parameter -l determines the position in which the condition gets assigned through the parameter -c. The output .csv file used for plotting was taken from the READemption output folder 'deseq_with_annotations'. The plotting was performed with the Python 3 kernel of Jupyter-notebook with a code as following:

```
% matplotlib inline
import pandas as pd
import matplotlib
```

```
import matplotlib.pyplot as plt
import numpy as np

deseq_data = pd.read_table("output file.csv", skiprows=2)
filtered = deseq_data[(deseq_data["padj"] < 0.05) \
& (abs(deseq_data["log2FoldChange"]) > 2)]
filtered2 = deseq_data[(deseq_data["padj"] < 0.05) \
& (abs(deseq_data["log2FoldChange"]) > 4)]

plt.plot(np.log10(deseq_data["baseMean"]), \
deseq_data["log2FoldChange"], "k.")
plt.plot(np.log10(filtered["baseMean"]), \
filtered["log2FoldChange"], "r.")
plt.plot(np.log10(filtered2["baseMean"]), \
filtered2["log2FoldChange"], "g.")
plt.plot(np.log10(top_ten_low["baseMean"]), \
top_ten_low["log2FoldChange"], "y.")
plt.plot(np.log10(top_ten_high["baseMean"]), \
top_ten_high["log2FoldChange"], "b.")
```

The created MA plot is shown in Figure 4.

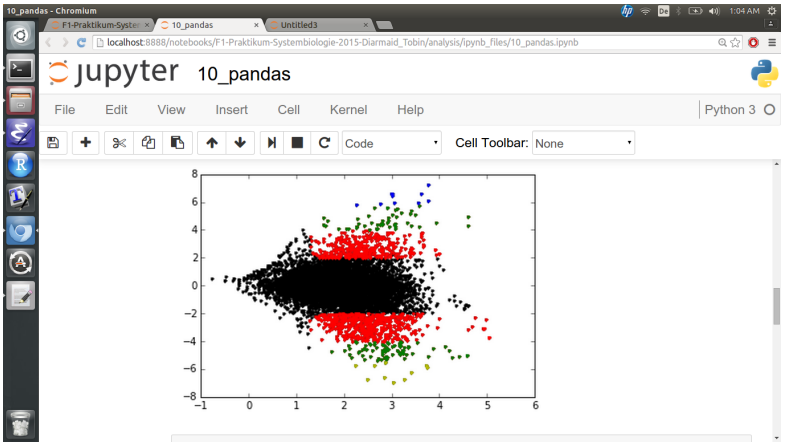

Figure 4: MA plot of DGE analysis of RNA-Seq data.

References

[1] Wikipedia. Linux — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[2] Wikipedia. Bash (unix shell) — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[3] Wikipedia. Secure shell — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[4] Wikipedia. Tmux — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[5] Wikipedia. Markdown — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[6] Wikipedia. Latex — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[7] Wikipedia. Python (programming language) — wikipedia, the free encyclopedia, 2015. [Online; accessed 3-November-2015].

[8] Wikipedia. Functional programming — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[9] Wikipedia. Lambda calculus — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[10] Wikipedia. Thread safety — wikipedia, the free encyclopedia, 2015. [Online; accessed 4-November-2015].

[11] Wikipedia. Object-oriented programming — wikipedia, the free encyclopedia, 2015. [Online; accessed 27-October-2015].

[12] Wikipedia. Creative commons license — wikipedia, the free encyclopedia, 2015. [Online; accessed 4-November-2015].

[13] Nature Publishing Group. Journals unite for reproducibility. *Nature*, 515:7, 2014.

[14] William Stafford Noble. A quick guide to organizing computational biology projects. *PLoS Comput Biol*, 5(7):e1000424, 07 2009.

[15] Jason A. Reuter, Damek V. Spacek, and Michael P. Snyder. High-throughput sequencing technologies. *Molecular Cell*, 58(4):586 – 597, 2015.

[16] Mark Gerstein Zhong Wang and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57 – 63, 2009.

[17] Arnoud H.M. Van Vliet. Next generation sequencing of microbial transcriptomes: challenges and opportunities. *FEMS Microbiology Letters*, 302(1):1–7, 2010.

[18]

[19] Nicole Cloonan, Qinying Xu, Geoffrey J. Faulkner, Darrin F. Taylor, Dave T. P. Tang, Gabriel Kolle, and Sean M. Grimmond. Rna-mate: a recursive mapping strategy for high-throughput rna-sequencing data. *Bioinformatics*, 25(19):2615–2616, 2009.

[20] Ondov BD Okou DT Zwick ME Bergman NH. Passalacqua KD, Varadarajan A. Structure and complexity of a bacterial transcriptome. *Journal of Bacteriology*, 191(10):3203 – 3211, 2009.

[21] Karla D. Passalacqua, Anjana Varadarajan, Charlotte Weist, Brian D. Ondov, Benjamin Byrd, Timothy D. Read, and Nicholas H. Bergman. Strand-specific rna-seq reveals ordered patterns of sense and antisense transcription in bacillus anthracis. *PLoS ONE*, 7(8):e43350, 08 2012.

[22] Jens Georg and Wolfgang R. Hess. cis-antisense rna, another level of gene regulation in bacteria. *Microbiology and Molecular Biology Reviews*, 75(2):286–300, 2011.

[23] Cynthia M. Sharma, Steve Hoffmann, Fabien Darfeuille, Jeremy Reignier, Sven Findeisz, Alexandra Sittka, Sandrine Chabas, Kristin Reiche, Joerg Hackermuller, Richard Reinhardt, Peter F. Stadler, and Joerg Vogel. The primary transcriptome of the major human pathogen helicobacter pylori. *Nature*, 464(7286):250–255, 2010.

[24] Cynthia M Sharma and Joerg Vogel. Differential rna-seq: the approach behind and the biological insight gained. *Current Opinion in Microbiology*, 19:97 – 105, 2014.

[25] Thorsten Bischler, Hock Siew Tan, Kay Nieselt, and Cynthia M. Sharma. Differential rna-seq (drna-seq) for annotation of transcriptional start sites and small rnas in helicobacter pylori. *Methods*, 86:89 – 101, 2015.

[26] Kai Papenfort, Konrad U. Foerstner, Jian-Ping Cong, Cynthia M. Sharma, and Bonnie L. Bassler. Differential rna-seq of vibrio cholerae identifies the vqmr small rna as a regulator of biofilm formation. *Proceedings of the National Academy of Sciences*, 112(7):E766–E775, 2015.

[27] Nuno A. Fonseca, Johan Rung, Alvis Brazma, and John C. Marioni. Tools for mapping high-throughput sequencing data. *Bioinformatics*, 2012.

[28] Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann, Jeremy Peterson, Owen White, and Steven L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369–2376, 1999.

[29] Michael Hoehl, Stefan Kurtz, and Enno Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18(suppl 1):S312–S320, 2002.

[30] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature Methods*, 9(4):357 – 359, 2012.

[31] Christian Otto, Peter F. Stadler, and Steve Hoffmann. Lacking alignments? the next-generation sequencing mapper segemehl revisited. *Bioinformatics*, 30(13):1837–1843, 2014.

[32] Mark Yandell and Daniel Ence. A beginner's guide to eukaryotic genome annotation. *Nat Rev Genet*, 13(5):329 – 342, 2012.

[33] Maureen K. Thomason, Thorsten Bischler, Sara K. Eisenbart, Konrad U. Foerstner, Aixia Zhang, Alexander Herbig, Kay Nieselt, Cynthia M. Sharma, and Gisela Storz. Global transcriptional start site mapping using differential rna sequencing reveals novel antisense rnas in escherichia coli. *Journal of Bacteriology*, 197(1):18–28, 2015.

[34] Konrad U. Foerstner, Joerg Vogel, and Cynthia M. Sharma. Reademption - a tool for the computational analysis of deep-sequencing-based transcriptome data. *Bioinformatics*, 2014.

[35] Bo Li and Colin Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.

[36] Joshua Bloom, Zia Khan, Leonid Kruglyak, Mona Singh, and Amy Caudy. Measuring differential gene expression by short read sequencing: quantitative comparison to 2-channel gene expression microarrays. *BMC Genomics*, 10(1):221, 2009.

[37] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010.

[38] Yuwen Liu, Jie Zhou, and Kevin P. White. Rna-seq differential expression studies: more sequence or more replication? *Bioinformatics*, 30(3):301–304, 2014.

[39] Paul L. Auer and R. W. Doerge. Statistical design and analysis of rna sequencing data. *Genetics*, 185(2):405–416, 2010.

[40] Davis J. McCarthy, Yunshun Chen, and Gordon K. Smyth. Differential expression analysis of multifactor rna-seq experiments with respect to biological variation. *Nucleic Acids Research*, 2012.

[41] Michael Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 15(12):550, 2014.

List of Figures

List of Tables