Genomics Tutorial

Release 2017.3beta

Sebastian Schmeier

Mar 03, 2017

CONTENTS

1 Introduction **3**
1.1 The workflow . . . 3
1.2 Learning outcomes . . . 3

2 NGS - Tool installation **5**
2.1 Install the conda package manager . . . 5
2.2 Create environment . . . 5
2.3 Install software . . . 5
2.4 General conda commands . . . 6

3 NGS - Quality control **7**
3.1 Preface . . . 7
3.2 Overview . . . 7
3.3 Learning outcomes . . . 7
3.4 The data . . . 7
3.5 The fastq file format . . . 10
3.6 The QC process . . . 10
3.7 PhiX genome . . . 10
3.8 Adapter trimming . . . 11
3.9 Quality assessment of sequencing reads (SolexaQA++) . . . 11
3.10 Sickle for dynamic trimming (alternative to SolexaQA++) . . . 13
3.11 Quality assessment of sequencing reads (FastQC) . . . 16

4 NGS - Genome assembly **21**
4.1 Preface . . . 21
4.2 Overview . . . 21
4.3 Learning outcomes . . . 21
4.4 Before we start . . . 21
4.5 Creating a genome assembly . . . 21
4.6 Assembly quality assessment . . . 23
4.7 Assemblathon . . . 24
4.8 Further reading . . . 24
4.9 Web links . . . 25

5 NGS - Read mapping **27**
5.1 Preface . . . 27
5.2 Overview . . . 27
5.3 Learning outcomes . . . 27
5.4 Before we start . . . 27
5.5 Mapping sequence reads to a reference genome . . . 29
5.6 Bowtie2 . . . 29
5.7 BWA . . . 30
5.8 The sam mapping file-format . . . 31
5.9 Mapping post-processing . . . 32
5.10 Mapping statistics . . . 33

5.11 Sub-selecting reads 34

6 NGS - Taxonomic investigation 37
6.1 Preface 37
6.2 Overview 37
6.3 Kraken 37
6.4 Before we start 37
6.5 Installation 39
6.6 Usage 39
6.7 Investigate taxa 40

7 NGS - Variant calling 45
7.1 Preface 45
7.2 Overview 45
7.3 Learning outcomes 45
7.4 Before we start 45
7.5 Installing necessary software 47
7.6 Preprocessing 47
7.7 Calling variants 47
7.8 Post-processing 48

8 NGS - Genome annotation 51

9 NGS - Variants-of-interest 53

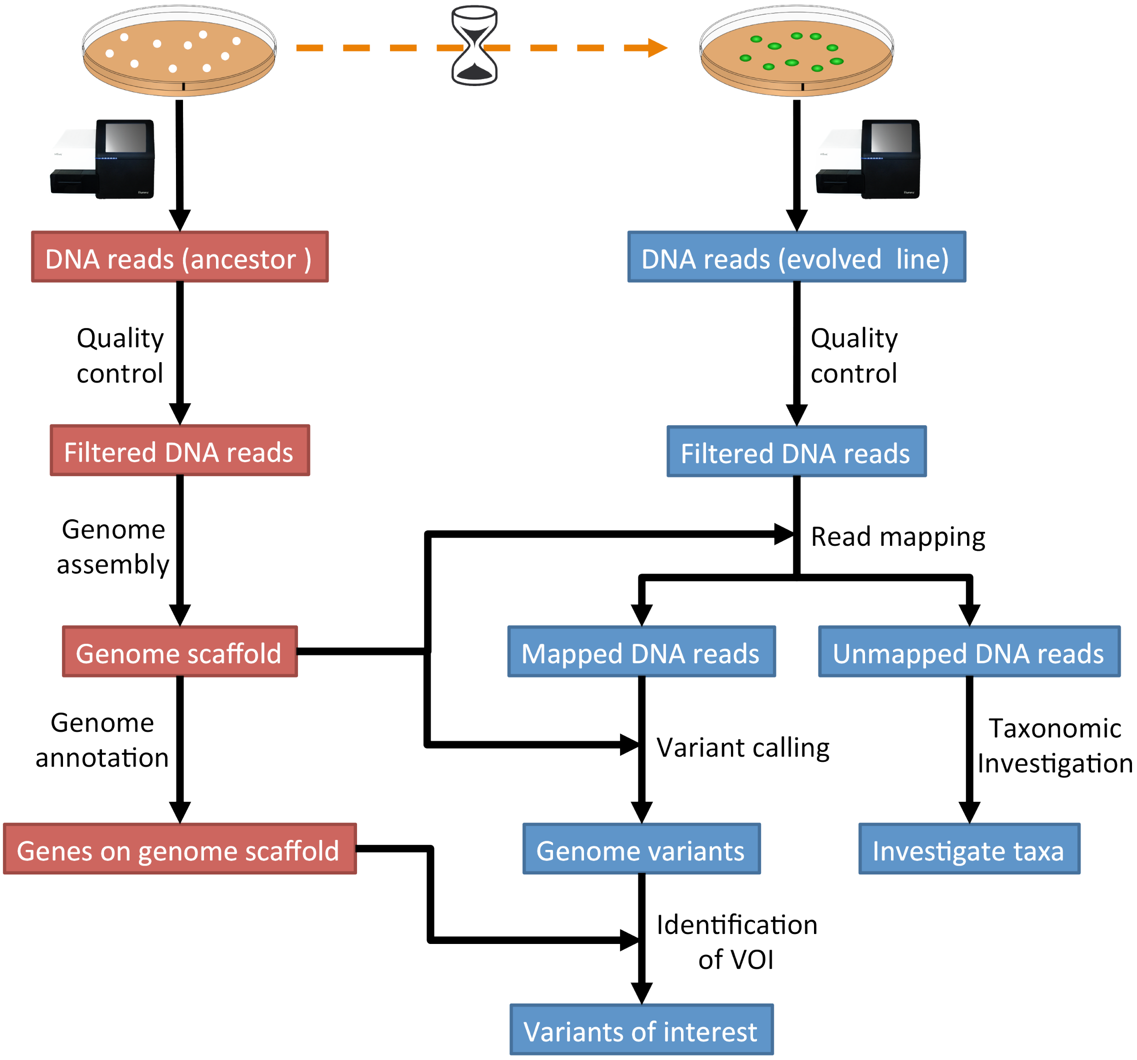
10 Quick command reference 55
10.1 Shell commands 55
10.2 General conda commands 55

11 Coding solutions 57
11.1 QC 57
11.2 Assembly 58
11.3 Mapping 58

12 Downloads 59
12.1 Tools 59
12.2 Data 59
12.3 Software 59

Bibliography 65

This is an introductory tutorial for learning genomics mostly on the command-line. You will learn how to analyse next-generation sequencing (NGS) data. The data you will be using is actual research data. The final aim is to identify the genome variations in evolved lines of wild yeast that can explain the observed biological phenotype.


DNA reads (ancestor)
DNA reads (evolved line)
Quality control
Quality control
Filtered DNA reads
Filtered DNA reads
Genome assembly
Read mapping
Genome scaffold
Mapped DNA reads
Unmapped DNA reads
Genome annotation
Variant calling
Taxonomic Investigation
Genes on genome scaffold
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

CHAPTER

ONE

INTRODUCTION

This is an introductory tutorial for learning genomics mostly on the Linux command-line. Should you need to refresh your knowledge about either Linux or the command-line, have a look here (http://linux.sschmeier.com/).

In this tutorial you will learn how to analyse next-generation sequencing (NGS) data. The data you will be using is actual research data. The experiment follows a similar strategy as in what is called an "experimental evolution" experiment *[KAWECKI2012]* (page 65), *[ZEYL2006]* (page 65). The final aim is to identify the genome variations in evolved lines of wild yeast that can explain the observed biological phenotype.

1.1 The workflow

The tutorial workflow is summarised in Fig. 1.1.

1.2 Learning outcomes

During this tutorial you will learn to:

- Check the data quality of an NGS experiment
- Create a genome assembly of the ancestor based on NGS data
- Map NGS reads of evolved lines to the created ancestral reference genome
- Call genome variations/mutations in the evolved lines
- Annotate a newly derived reference genome
- Find variants of interest that may be responsible for the observed evolved phenotype

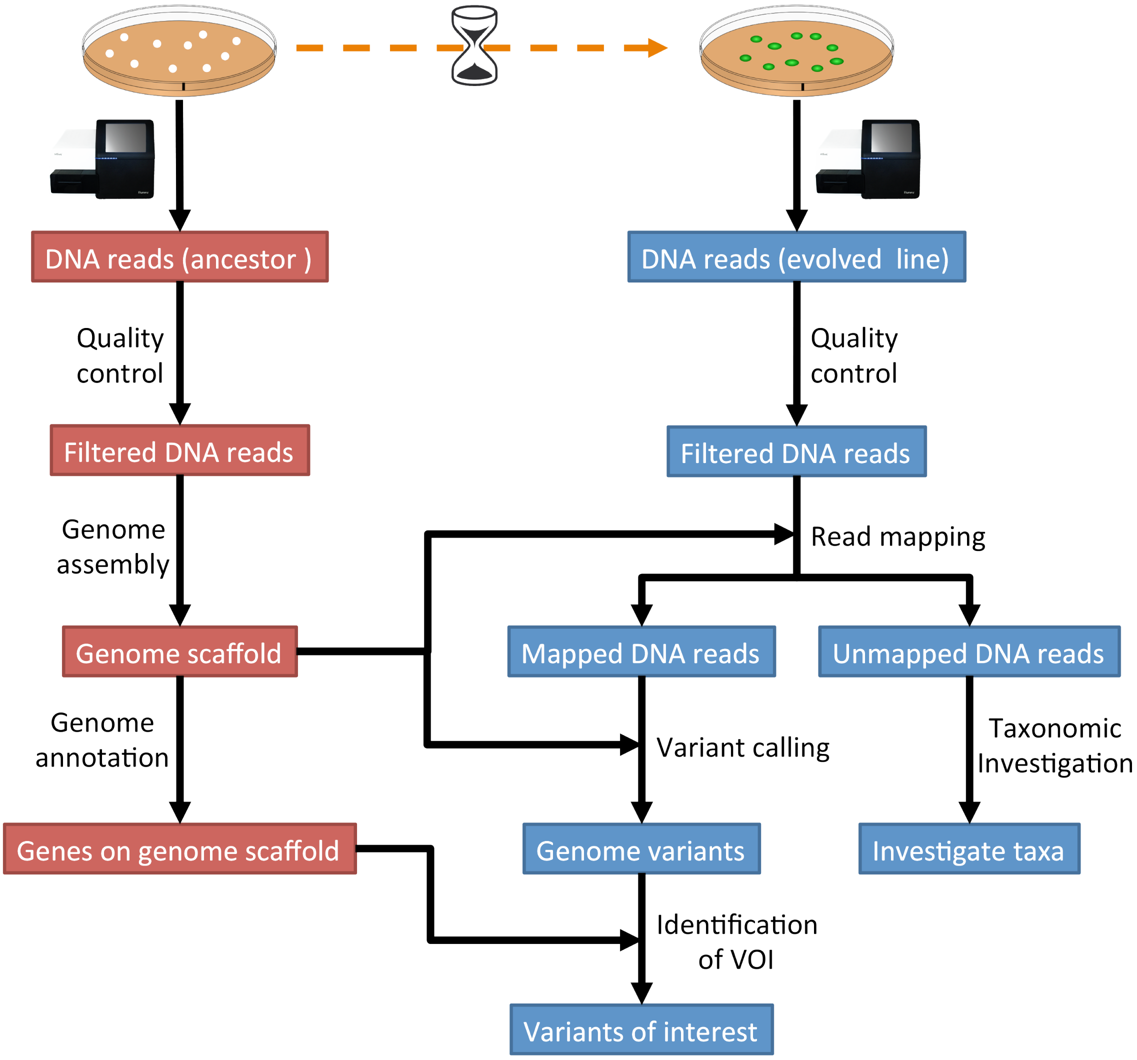

DNA reads (ancestor)
DNA reads (evolved line)
Quality control
Quality control
Filtered DNA reads
Filtered DNA reads
Genome assembly
Read mapping
Genome scaffold
Mapped DNA reads
Unmapped DNA reads
Genome annotation
Variant calling
Taxonomic Investigation
Genes on genome scaffold
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

Fig. 1.1: The tutorial will follow this workflow.

CHAPTER

TWO

NGS - TOOL INSTALLATION

2.1 Install the conda package manager

We will use the package/tool managing system conda (http://conda.pydata.org/miniconda.html) to install some programs that we will use during the course. It is not installed by default, thus we need to install it first to be able to use it.

```
# download latest conda installer
curl -O https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh

# run the installer
bash Miniconda3-latest-Linux-x86_64.sh

# delete the installer after successful run
rm Miniconda3-latest-Linux-x86_64.sh

# Install some conda channels
# A channel is where conda looks for packages
conda config --add channels conda-forge
conda config --add channels defaults
conda config --add channels r
conda config --add channels bioconda
```

Close shell/terminal, **re-open** new shell/terminal.

```
conda update conda
```

> **Attention:** Should the conda installer download fail. Please find links to alternative locations on the *Downloads* (page 59) page.

2.2 Create environment

We create a conda (http://conda.pydata.org/miniconda.html) environment for some tools This is useful to work **reproducible** as we can easily re-create the tool-set with the same version numbers later on.

```
conda create -n ngs python=3
# activate the environment
source activate ngs
```

2.3 Install software

To install software into the activated environment, one uses the command `conda install`.

```
# install more tools into the environment
conda install package
```

Note: To tell if you are in the correct conda environment, look at the command-prompt. Do you see the name of the environment in round brackets at the very beginning of the prompt, e.g. (ngs)? If not, activate the `ngs` environment with `source activate ngs` before installing the tools.

2.4 General conda commands

```
# to search for packages
conda search [package]

# To update all packages
conda update --all --yes

# List all packages installed
conda list [-n env]

# conda list environments
conda env list

# create new env
conda create -n [name] package [package] ...

# activate env
source activate [name]

# deavtivate env
source deactivate
```

CHAPTER

THREE

NGS - QUALITY CONTROL

3.1 Preface

There are many sources of errors that can influence the quality of your sequencing run *[ROBASKY2014]* (page 65). In this quality control section we will use our skill on the command-line interface to deal with the task of investigating the quality and cleaning sequencing data *[KIRCHNER2014]* (page 65).

Note: You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

3.2 Overview

The part of the workflow we will work on in this section can be viewed in Fig. 3.1.

3.3 Learning outcomes

After studying this tutorial you should be able to:

1. Describe the steps involved in pre-processing/cleaning sequencing data.
2. Distinguish between a good and a bad sequencing run.
3. Compute, investigate and evaluate the quality of sequence data from a sequencing experiment.

3.4 The data

First, we are going to download the data we will analyse. Open a shell/terminal.

```
# create a directory you work in
mkdir analysis

# change into the directory
cd analysis

# download the data
curl -O http://compbio.massey.ac.nz/data/203341/data.tar.gz

# uncompress it
tar -xvzf data.tar.gz
```

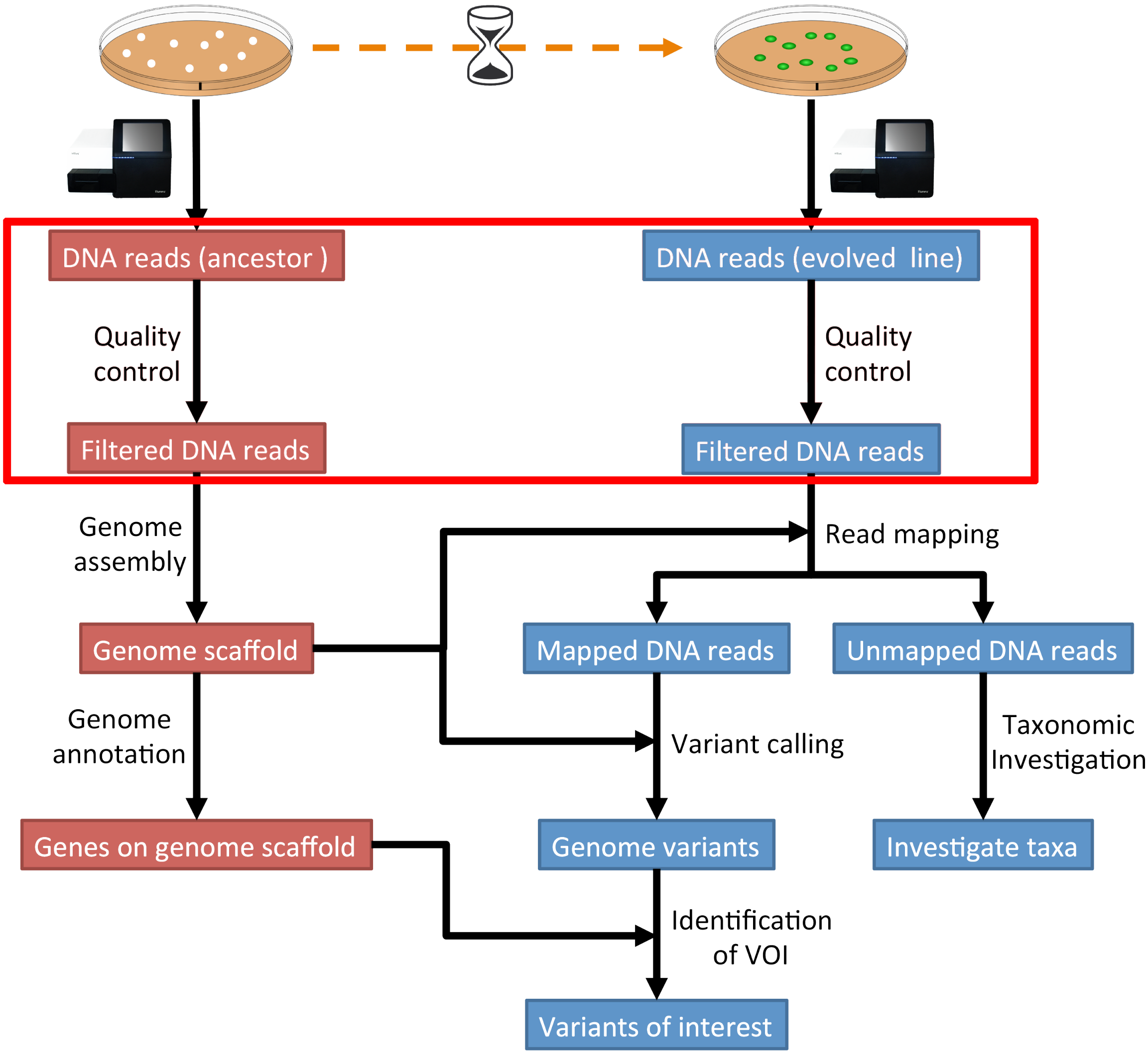

DNA reads (ancestor)
DNA reads (evolved line)
Quality control
Quality control
Filtered DNA reads
Filtered DNA reads
Genome assembly
Read mapping
Genome scaffold
Mapped DNA reads
Unmapped DNA reads
Genome annotation
Variant calling
Taxonomic Investigation
Genes on genome scaffold
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

Fig. 3.1: The part of the workflow we will work on in this section marked in red.

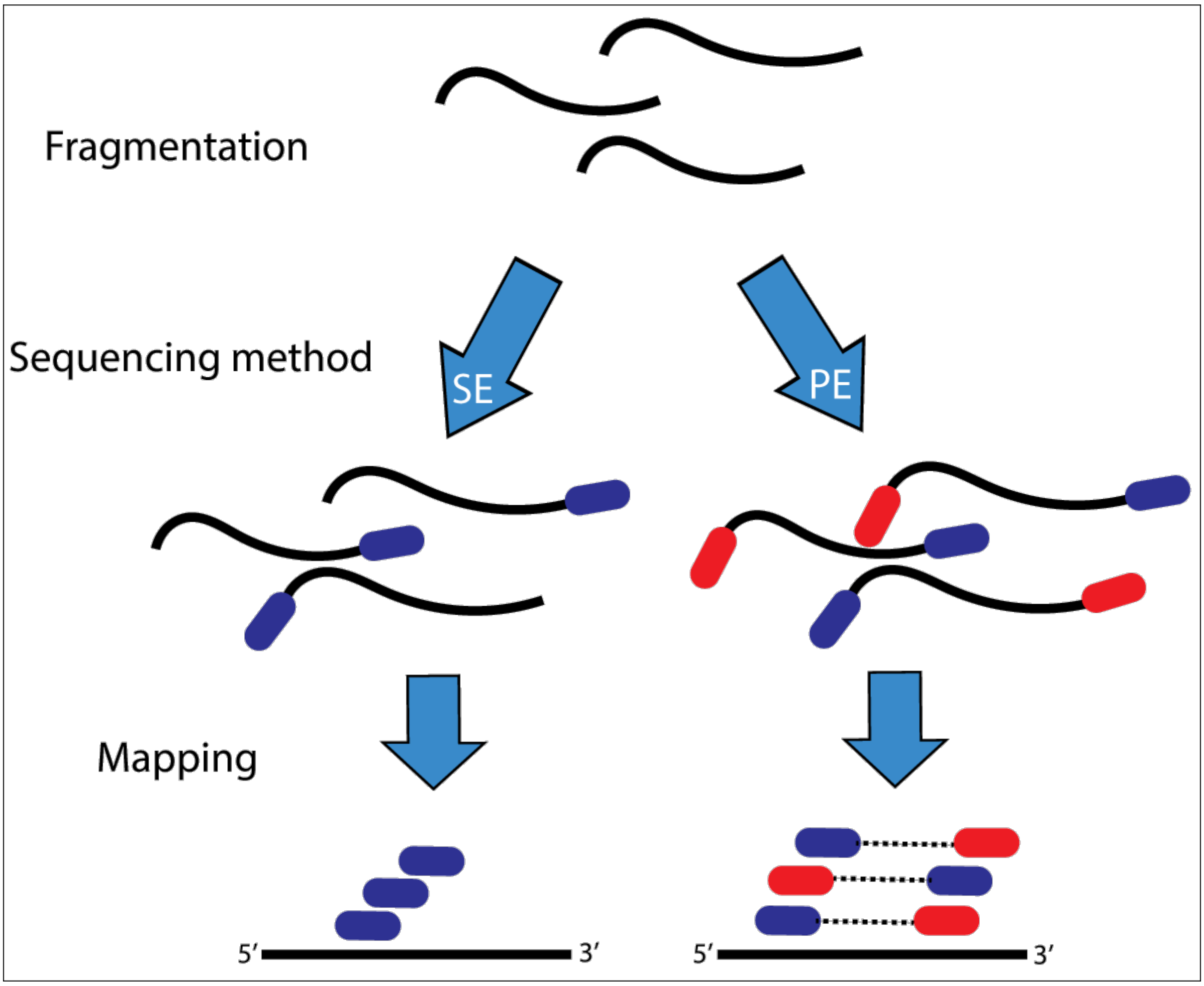

Fragmentation
Sequencing method
SE
PE
Mapping
5′
3′
5′
3′

Fig. 3.2: Illustration of single-end (SE) versus paired-end (PE) sequencing.

The data is from a paired-end sequencing run data (see Fig. 3.2) from an Illumina (http://illumina.com) MiSeq *[GLENN2011]* (page 65). Thus, we have two files, one for each end of the read.

If you need to refresh how Illumina (http://illumina.com) paired-end sequencing works have a look at the Illumina technology webpage (http://www.illumina.com/technology/next-generation-sequencing/paired-end-sequencing_assay.html) and this video (https://youtu.be/HMyCqWhwB8E).

Note: The data we are using is "almost" raw data coming from the machine. This data has been post-processed in two ways already. All sequences that were identified as belonging to the PhiX genome have been removed. This process requires some skills we will learn in later sections. Illumina (http://illumina.com) adapters have been removed as well already! The process is explained below but we are not going to do it.

3.4.1 Investigate the data

Make use of your newly developed skills on the command-line to investigate the files in `data` folder.

Todo

1. Use the command-line to get some ideas about the file.
2. What kind of files are we dealing with?
3. How many sequence reads are in the file?

3.5 The fastq file format

The data we receive from the sequencing is in `fastq` format. To remind us what this format entails, we can revisit the fastq wikipedia-page (https://en.wikipedia.org/wiki/FASTQ_format)!

A useful tool to decode base qualities can be found here (http://broadinstitute.github.io/picard/explain-qualities.html).

Todo

Explain briefly what the quality value represents.

3.6 The QC process

There are a few steps one need to do when getting the raw sequencing data from the sequencing facility:

1. Remove PhiX sequences
2. Adapter trimming
3. Quality trimming of reads
4. Quality assessment

3.7 PhiX genome

PhiX (https://en.wikipedia.org/wiki/Phi_X_174) is a nontailed bacteriophage with a single-stranded DNA and a genome with 5386 nucleotides. PhiX is used as a quality and calibration control for sequencing runs (http://www.illumina.com/products/by-type/sequencing-kits/cluster-gen-sequencing-reagents/phix-control-v3.html). PhiX is often added at a low known concentration, spiked in the same lane along with the sample or used as a separate lane. As the concentration of the genome is known, one can calibrate the instruments. Thus, PhiX genomic sequences need to be removed before processing your data further as this constitutes a deliberate contamination *[MUKHERJEE2015]* (page 65). The steps involve mapping all reads to the "known" PhiX genome, and removing all of those sequence reads from the data.

However, your sequencing provider might not have used PhiX, thus you need to read the protocol carefully, or just do this step in any case.

Note: We are not going to do this step here, as this has been already done. Please see the *NGS - Read mapping* (page 27) section on how to map reads against a reference genome.

3.8 Adapter trimming

The process of sequencing DNA via Illumina (http://illumina.com) technology requires the addition of some adapters to the sequences. These get sequenced as well and need to be removed as they are artificial and do not belong to the species we try to sequence.

Note: The process of how to do this is explained here, however we are not going to do this as our sequences have been adapter-trimmed already.

First, we need to know the adapter sequences that were used during the sequencing of our samples. Normally, you should ask your sequencing provider, who should be providing this information to you. Illumina (http://illumina.com) itself provides a document (https://support.illumina.com/downloads/illumina-customer-sequence-letter.html) that describes the adapters used for their different technologies. Also the FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/) tool, we will be using later on, provides a collection of contaminants and adapters (https://github.com/csf-ngs/fastqc/blob/master/Contaminants/contaminant_list.txt).

Second, we need a tool that takes a list of adapters and scans each sequence read and removes the adapters. Install a tool called fastq-mcf (https://github.com/ExpressionAnalysis/ea-utils/blob/wiki/FastqMcf.md) from the ea-utils suite (https://expressionanalysis.github.io/ea-utils/) of tools that is able to do this.

```
# install
conda install ea-utils
```

Using the tool together with a adapter/contaminants list in fasta-file (here denoted as `adapters.fa`):

```
fastq-mcf -o  -o cleaned.R1.fq.gz -o cleaned.R2.fq.gz adapaters.fa infile_R1.fastq␣
↪infile_R2.fastq
```

- `-o`: Specifies the output-files. These are fastq-files for forward and reverse read, with adapters removed.

3.9 Quality assessment of sequencing reads (SolexaQA++)

To assess the sequence read quality of the Illumina (http://illumina.com) run we make use of a program called SolexaQA++ (http://solexaqa.sourceforge.net) *[COX2010]* (page 65). SolexaQA++ (http://solexaqa.sourceforge.net) was originally developed to work with Solexa data (since bought by Illumina (http://illumina.com)), but long since working with Illumina (http://illumina.com) data. It produces nice graphics that intuitively show the quality of the sequences. it is also able to dynamically trim the bad quality ends off the reads.

From the webpage:

> "SolexaQA calculates sequence quality statistics and creates visual representations of data quality for second-generation sequencing data. Originally developed for the Illumina system (historically known as "Solexa"), SolexaQA now also supports Ion Torrent and 454 data."

3.9.1 Install SolexaQA++

Unfortunately, currently we cannot install SolexaQA++ (http://solexaqa.sourceforge.net) with conda (http://conda.pydata.org/miniconda.html).

```
curl -O http://compbio.massey.ac.nz/data/203341/SolexaQA.tar.gz

# uncompress the archive
tar -xvzf SolexaQA.tar.gz

# make the file executable
chmod a+x SolexaQA/Linux_x64/SolexaQA++

# copy program to root folder
cp ./SolexaQA/Linux_x64/SolexaQA++ .

# run the program
./SolexaQA++
```

Note: Should the download fail, download manually from *Downloads* (page 59).

3.9.2 SolexaQA++ manual

SolexaQA++ (http://solexaqa.sourceforge.net) has three modes that can be run. Type:

```
./SolexaQA++
```

```
SolexaQA++ v3.1.3
Released under GNU General Public License version 3
C++ version developed by Mauro Truglio (M.Truglio@massey.ac.nz)

Usage: SolexaQA++ <command> [options]

Command: analysis      quality analysis and graphs generation
         dynamictrim    trim reads using a chosen threshold
         lengthsort   sort reads by a chosen length
```

The three modes are: `analysis`, `dynamictrim`, and `lengthsort`:

`analysis` - the primary quality analysis and visualization tool. Designed to run on unmodified FASTQ files obtained directly from Illumina (http://illumina.com), Ion Torrent or 454 sequencers.

`dynamictrim` - a read trimmer that individually crops each read to its longest contiguous segment for which quality scores are greater than a user-supplied quality cutoff.

`lengthsort` - a program to separate high quality reads from low quality reads. LengthSort assigns trimmed reads to paired-end, singleton and discard files based on a user-defined length cutoff.

3.9.3 SolexaQA++ dynamic trimming

We will use SolexaQA++ (http://solexaqa.sourceforge.net) dynamic trim the reads, to chop of nucleotides witha a bad quality score.

Todo

1. Create a directory for the result-files –> **trimmed/**.
2. Run SolexaQA++ (http://solexaqa.sourceforge.net) `dynamictrim` with the untrimmed data and a probability cutoff of 0.01., and submit result-directory **trimmed/**.
3. Investigate the result-files in **trimmed/**, e.g. do the file-sizes change to the original files?
4. SolexaQA++ (http://solexaqa.sourceforge.net) `dynamictrim` produces a graphical output. Explain what the graph shows. Find heklp on the SolexaQA++ (http://solexaqa.sourceforge.net) website.

Hint: Should you not get 1 and/or 2 right, try the commands in *Code: SolexaQA++ trimming* (page 57).

3.9.4 SolexaQA++ analysis on trimmed data

Todo

1. Create a directory for the result-files –> **trimmed-solexaqa**.
2. Use SolexaQA++ (http://solexaqa.sourceforge.net) to do the quality assessment with the trimmed data-set.
3. Compare your results to the examples of a particularly bad MiSeq run (Fig. 3.6 to Fig. 3.6, taken from SolexaQA++ (http://solexaqa.sourceforge.net) website). Write down your observations.
4. What elements in these example figures (Fig. 3.3 to Fig. 3.6) indicate that the show a bad run? Write down your explanations.

Hint: Should you not get 1 and/or 2 it right, try the commands in *Code: SolexaQA++ qc* (page 57).

3.10 Sickle for dynamic trimming (alternative to SolexaQA++)

Should the dynamic trimming not work with SolexaQA++ (http://solexaqa.sourceforge.net), you can alternatively use Sickle (https://github.com/najoshi/sickle).

```
source activate ngs
conda install sickle-trim
```

Now we are going to run the program on our paired-end data:

# create a new directory mkdir trimmed

# sickle parameters: sickle –help

# as we are dealing with paired-end data you will be using "sickle pe" sickle pe –help

# run sickle like so: sickle pe -g -t sanger -f data/ancestor-R1.fastq.gz -r data/ancestor-R2.fastq.gz -o trimmed/ancestor-R1.trimmed.fastq.gz -p trimmed/ancestor-R2.trimmed.fastq.gz

---

Hint: Should you be unable to run Sickle (https://github.com/najoshi/sickle) or SolexaQA++ (http://solexaqa.sourceforge.net) at all to trim the data. You can download the trimmed dataset here (http://compbio.massey.ac.nz/data/203341/trimmed.tar.gz). Unarchive and uncompress the files with `tar -xvzf trimmed.tar.gz`.

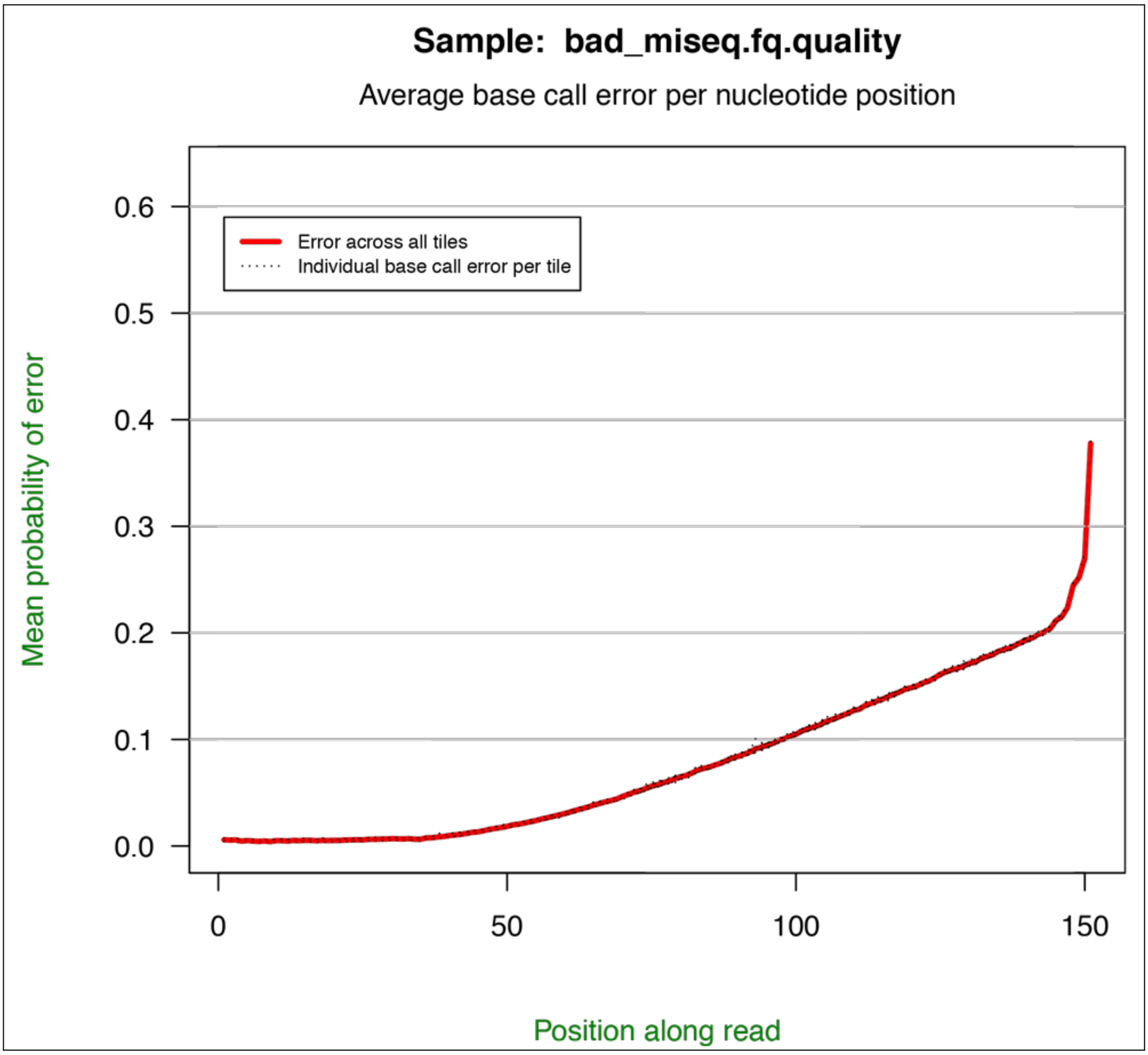


Fig. 3.3: SolexaQA++ example quality plot along reads of a bad MiSeq run

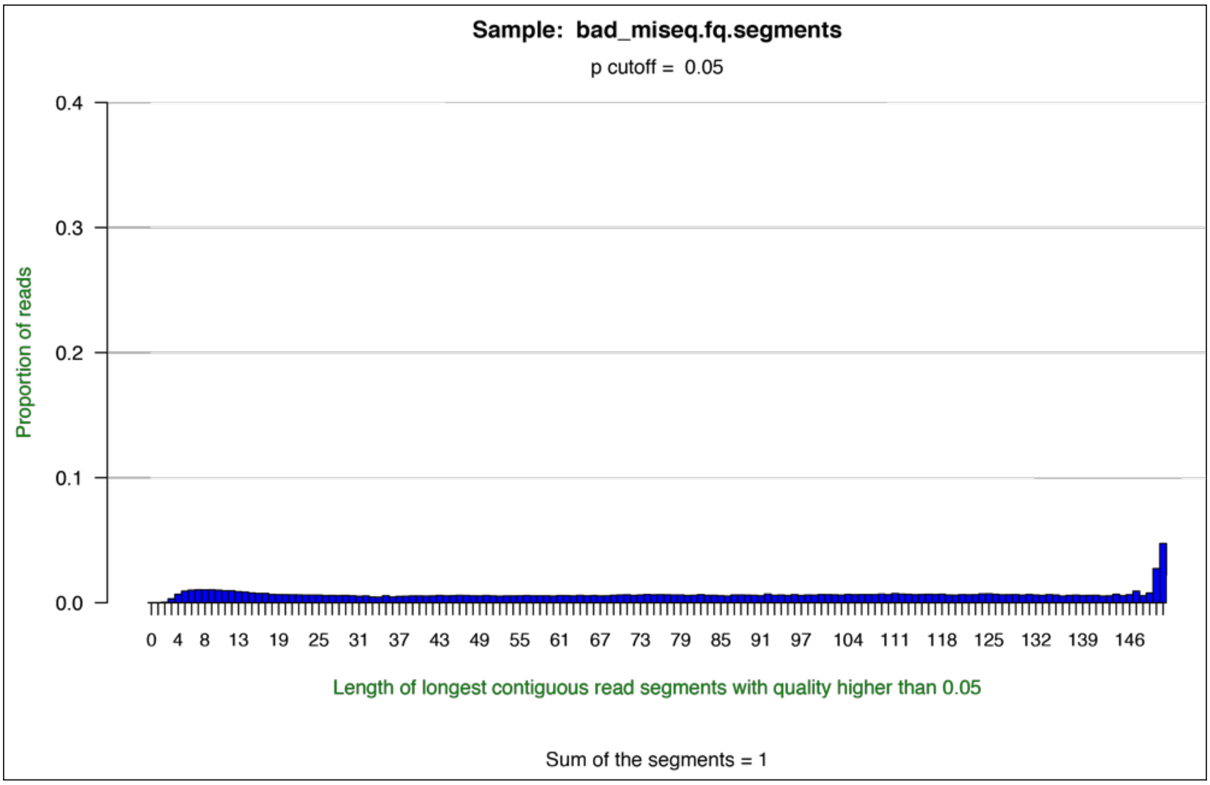


Fig. 3.4: SolexaQA++ example histogram plot of a bad MiSeq run.

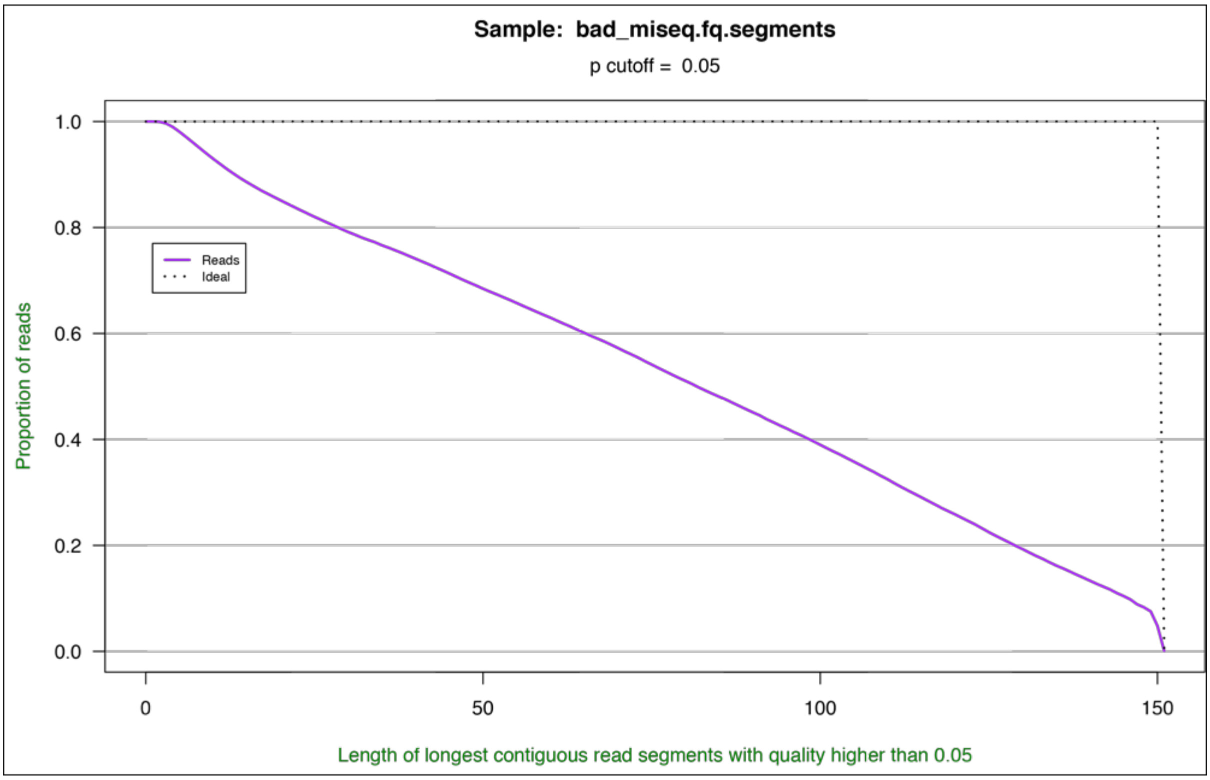


Fig. 3.5: SolexaQA++ example cumulative plot of a bad MiSeq run.

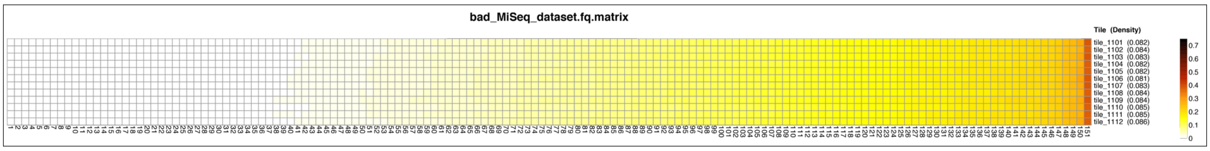


Fig. 3.6: SolexaQA++ example quality heatmap of a bad MiSeq run.

3.11 Quality assessment of sequencing reads (FastQC)

3.11.1 Installing FastQC

```
source activate ngs
conda install fastqc

# should now run the program
fastqc --help
```

```
            FastQC - A high throughput sequence QC analysis tool

SYNOPSIS

        fastqc seqfile1 seqfile2 .. seqfileN

    fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
           [-c contaminant file] seqfile1 .. seqfileN

DESCRIPTION

    FastQC reads a set of sequence files and produces from each one a quality
    control report consisting of a number of different modules, each one of
    which will help to identify a different potential type of problem in your
    data.

    If no files to process are specified on the command line then the program
    will start as an interactive graphical application.  If files are provided
    on the command line then the program will run with no user interaction
    required.  In this mode it is suitable for inclusion into a standardised
    analysis pipeline.
```

3.11.2 FastQC manual

FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/) is a very simple program to run that provides similar and additional information to SolexaQA++ (http://solexaqa.sourceforge.net).

From the webpage:

> "FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis."

The basic command looks like:

```
$ fastqc -o RESULT-DIR INPUT-FILE.[txt/fa/fq] ...
```

- `-o RESULT-DIR` is the directory where the result files will be written
- `INPUT-FILE.[txt/fa/fq]` is the sequence file to analyze, can be more than one file.

Hint: The result will be a HTML page per input file that can be opened in a web-browser.

3.11.3 Run FastQC on the untrimmed and trimmed data

Todo

1. Create a directory for the results –> **trimmed-fastqc**
2. Run FastQC on all **trimmed** files.
3. Visit the FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/) website and read about sequencing QC reports for good and bad Illumina (http://illumina.com) sequencing runs.
4. Compare your results to these examples (Fig. 3.7 to Fig. 3.9) of a particularly bad run (taken from the FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/) website) and write down your observations with regards to your data.
5. What elements in these example figures (Fig. 3.7 to Fig. 3.9) indicate that the example is from a bad run?

Hint: Should you not get it right, try the commands in *Code: FastQC* (page 57).

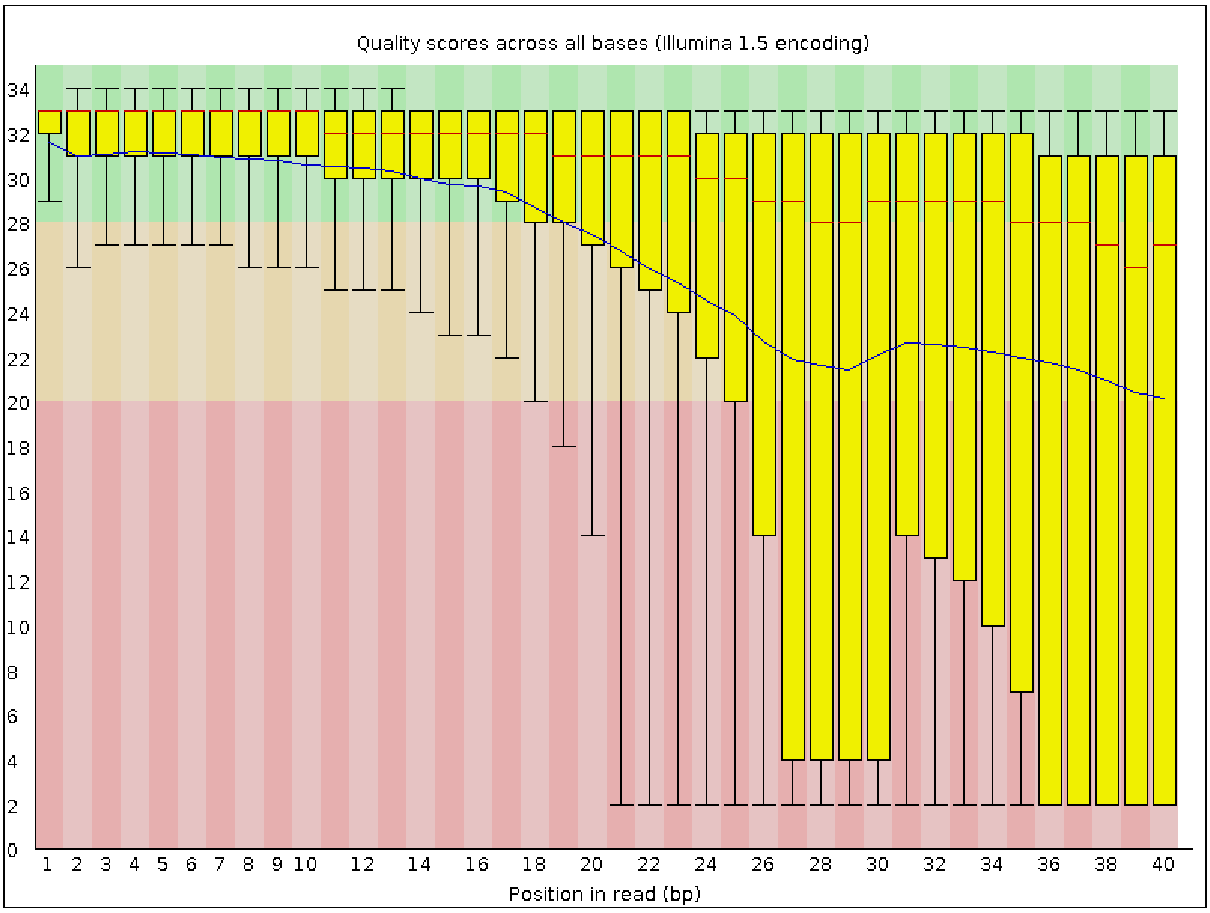


Fig. 3.7: Quality score across bases.

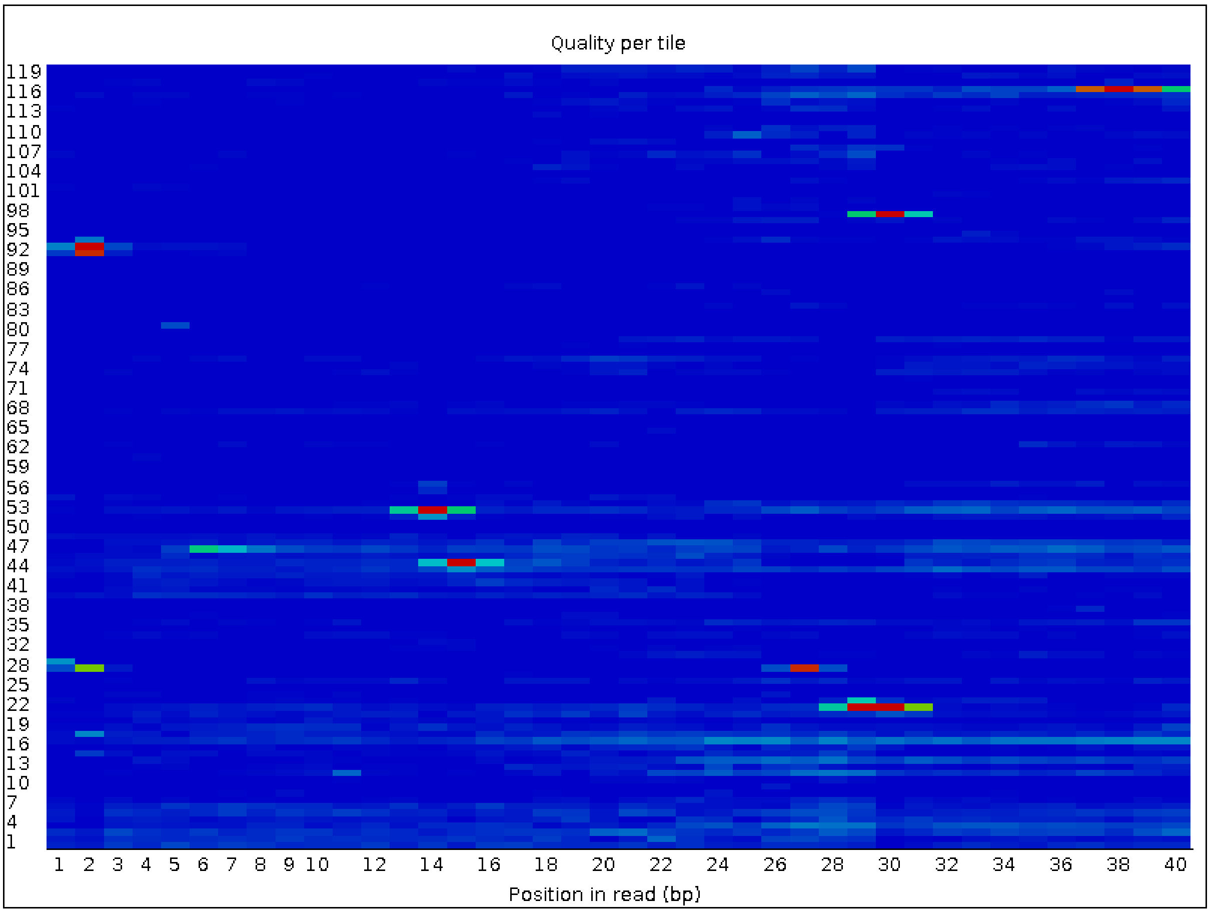


Fig. 3.8: Quality per tile.

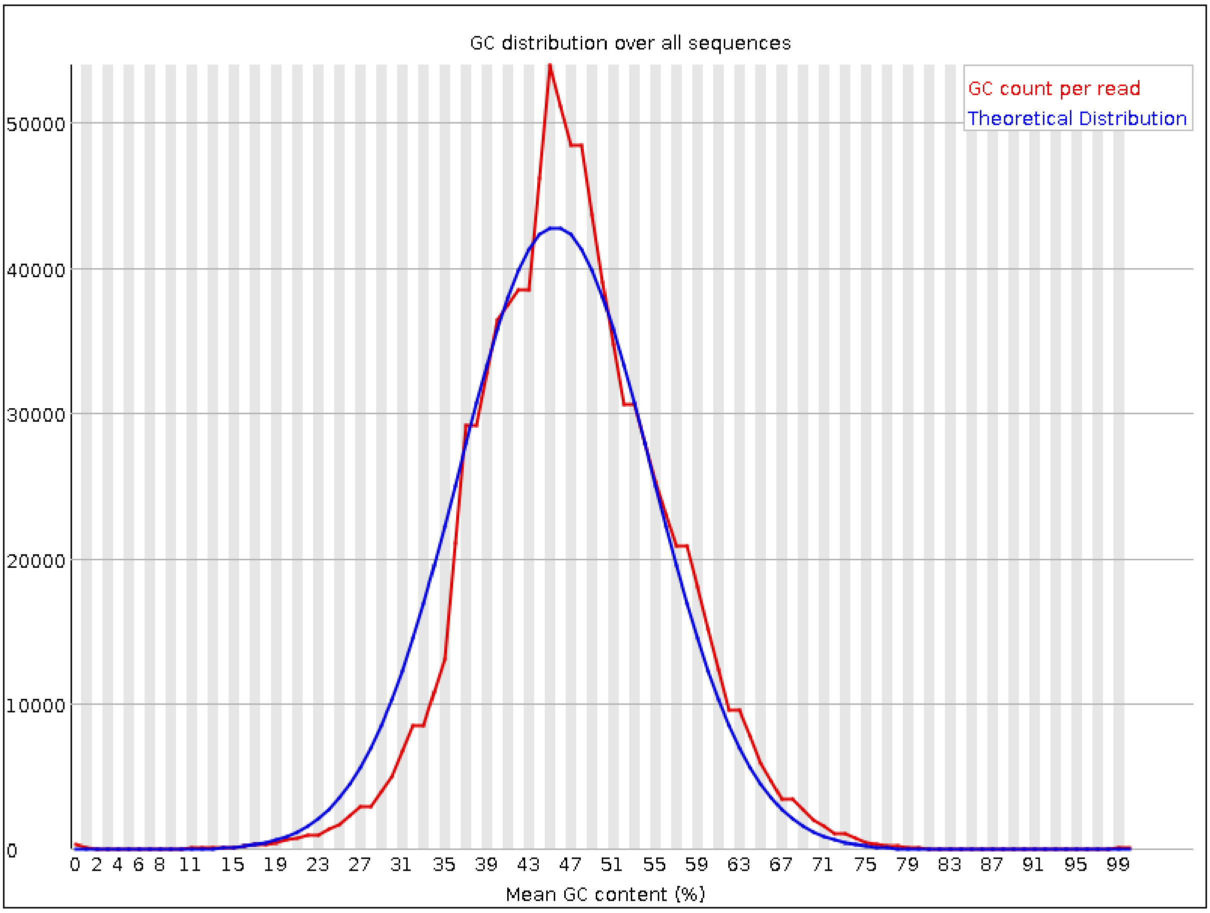


Fig. 3.9: GC distribution over all sequences.

CHAPTER

FOUR

NGS - GENOME ASSEMBLY

4.1 Preface

In this section we will use our skill on the command-line interface to create a genome assembly from sequencing data.

Note: You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

4.2 Overview

The part of the workflow we will work on in this section can be viewed in Fig. 4.1.

4.3 Learning outcomes

After studying this tutorial you should be able to:

1. Compute and interpret a whole genome assembly.
2. Judge the quality of a genome assembly.

4.4 Before we start

Lets see how our directory structure looks so far:

```
cd ~/analysis
ls -1F
```

```
data/
SolexaQA/
SolexaQA++
trimmed/
trimmed-fastqc/
trimmed-solexaqa/
```

4.5 Creating a genome assembly

We want to create a genome assembly for our ancestor. We are going to use the quality trimmed forward and backward DNA sequences and use a program called SPAdes (http://bioinf.spbau.ru/spades) to build a genome

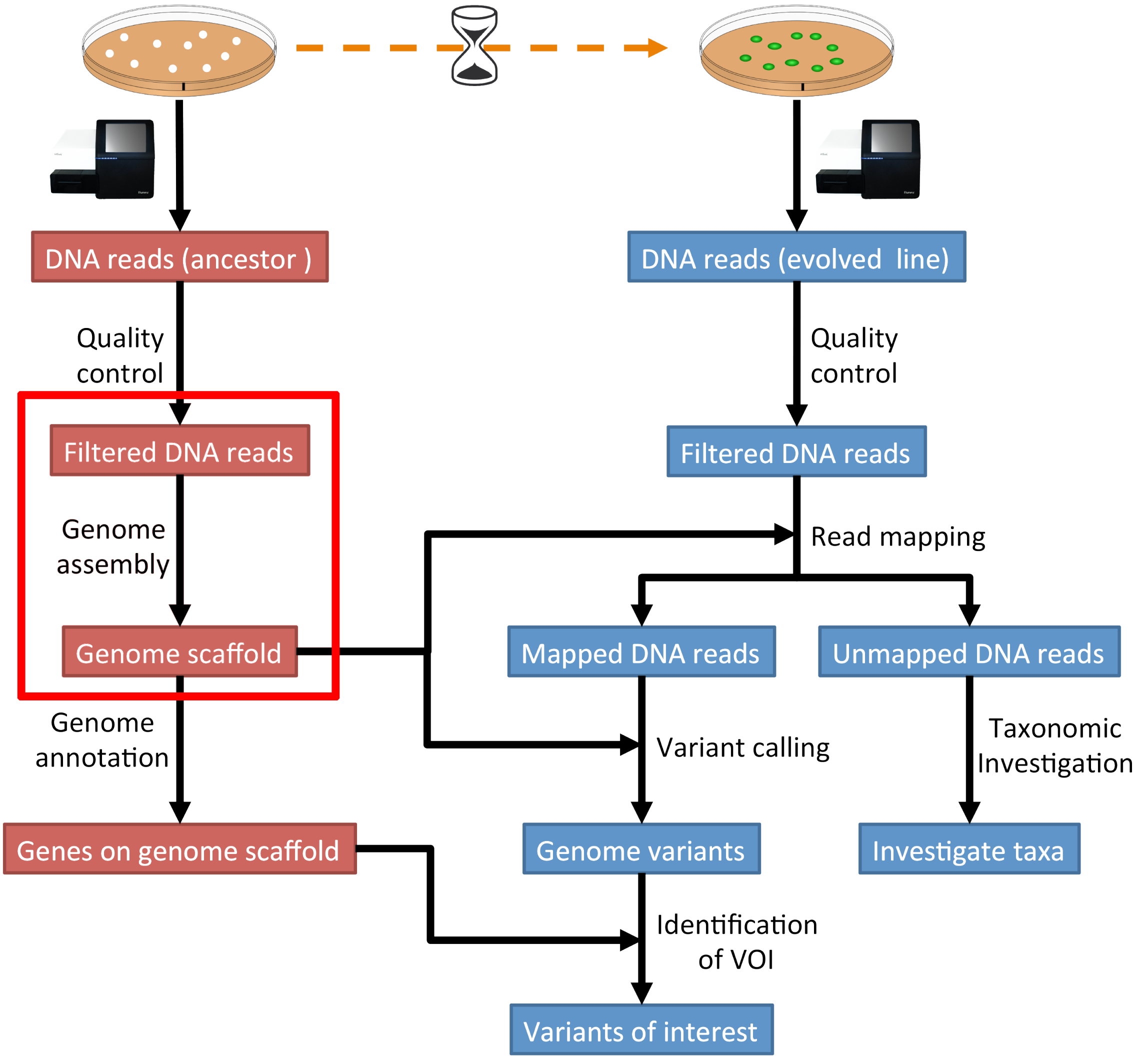

DNA reads (ancestor)
DNA reads (evolved line)
Quality control
Quality control
Filtered DNA reads
Filtered DNA reads
Genome assembly
Read mapping
Genome scaffold
Mapped DNA reads
Unmapped DNA reads
Genome annotation
Variant calling
Taxonomic Investigation
Genes on genome scaffold
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

Fig. 4.1: The part of the workflow we will work on in this section marked in red.

assembly.

Todo

1. Discuss briefly why we are using the ancestral sequences to create a reference genome as opposed to the evolved line.

4.5.1 Installing the software

We are going to use a program called SPAdes (http://bioinf.spbau.ru/spades) fo assembling our genome. In a recent evaluation of assembly software, SPAdes (http://bioinf.spbau.ru/spades) was found to be a good choice for fungal genomes *[ABBAS2014]* (page 65). It is also simple to install and use.

```
source activate ngs
conda install spades
```

4.5.2 SPAdes usage

```
# change to your analysis root folder
cd ~/analysis

# first create a output directory for the assemblies
mkdir assembly

# to get a help for spades and an overview of the parameter type:
spades.py -h
```

The two files we need to submit to SPAdes (http://bioinf.spbau.ru/spades) are two paired-end read files.

spades.py -o assembly/spades-default/ -1 trimmed/ancestor-R1.fastq.trimmed.gz -2 trimmed/ancestor-R2.fastq.trimmed.gz

---

Todo

1. Run SPAdes (http://bioinf.spbau.ru/spades) with default parameters on the ancestor
2. Read in the SPAdes (http://bioinf.spbau.ru/spades) manual about about assembling with 2x150bp reads
3. Run SPAdes (http://bioinf.spbau.ru/spades) a second time but use the options suggested at the SPAdes (http://bioinf.spbau.ru/spades) manual section 3.4 (http://spades.bioinf.spbau.ru/release3.9.1/manual.html#sec3.4) for assembling 2x150bp paired-end reads (are fungi multicellular?). Use a different output directory `assembly/spades-150` for this run.

Hint: Should you not get it right, try the commands in *Code: SPAdes assembly* (page 58).

4.6 Assembly quality assessment

4.6.1 Assembly statistics

Quast (http://quast.bioinf.spbau.ru/) (QUality ASsesment Tool) *[GUREVICH2013]* (page 65), evaluates genome assemblies by computing various metrics, including:

- N50: length for which the collection of all contigs of that length or longer covers at least 50% of assembly length
- NG50: where length of the reference genome is being covered
- NA50 and NGA50: where aligned blocks instead of contigs are taken
- missassemblies: misassembled and unaligned contigs or contigs bases
- genes and operons covered

It is easy with Quast (http://quast.bioinf.spbau.ru/) to compare these measures among several assemblies. The program can be used on their website (http://quast.bioinf.spbau.ru/).

We can install it locally with:

```
source activate ngs
conda install quast
```

Run Quast (http://quast.bioinf.spbau.ru/) with both assembly scaffolds.fasta files to compare the results.

Hint: Should you be unable to run SPAdes (http://bioinf.spbau.ru/spades) on the data, you can download the assemblies here (http://compbio.massey.ac.nz/data/203341/assembly.tar.gz). Unarchive and uncompress the files with `tar -xvzf assembly.tar.gz`.

quast -o assembly/quast assembly/spades-default/scaffolds.fasta assembly/spades-150/scaffolds.fasta

Todo

1. Compare the results of Quast (http://quast.bioinf.spbau.ru/) with regards to the two different assemblies.
2. Which one do you prefer and why?

4.7 Assemblathon

Todo

Now that you know the basics for assembling a genome and judging their quality, play with the SPAdes (http://bioinf.spbau.ru/spades) parameters to create the best assembly possible.

Todo

Once you have your final assembly, rename your assembly directory int `spades-final`, e.g. `mv assembly/spades-default assembly/spades-final`.

4.8 Further reading

4.8.1 Background on Genome Assemblies

- How to apply de Bruijn graphs to genome assembly. *[COMPEAU2011]* (page 65)
- Sequence assembly demystified. *[NAGARAJAN2013]* (page 65)

4.8.2 Evaluation of Genome Assembly Software

- GAGE: A critical evaluation of genome assemblies and assembly algorithms. *[SALZBERG2012]* (page 65)
- Assessment of de novo assemblers for draft genomes: a case study with fungal genomes. *[ABBAS2014]* (page 65)

4.9 Web links

- Lectures for this topic: Genome Assembly: An Introduction (https://dx.doi.org/10.6084/m9.figshare.2972323.v1)
- SPAdes (http://bioinf.spbau.ru/spades)
- Quast (http://quast.bioinf.spbau.ru/)
- Bandage (https://rrwick.github.io/Bandage/) (Bioinformatics Application for Navigating De novo Assembly Graphs Easily) is a program that visualizes a genome assembly as a graph *[WICK2015]* (page 65).

CHAPTER

FIVE

NGS - READ MAPPING

5.1 Preface

In this section we will use our skill on the command-line interface to map our reads from the evolved line to our ancestral reference genome.

Note: You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

5.2 Overview

The part of the workflow we will work on in this section can be viewed in Fig. 5.1.

5.3 Learning outcomes

After studying this section of the tutorial you should be able to:

1. Explain the process of sequence read mapping.
2. Use bioinformatics tools to map sequencing reads to a reference genome.
3. Filter mapped reads based on quality.

5.4 Before we start

Lets see how our directory structure looks so far:

```
cd ~/analysis
ls -1F
```

```
assembly/
data/
SolexaQA/
SolexaQA++
trimmed/
trimmed-fastqc/
trimmed-solexaqa/
```

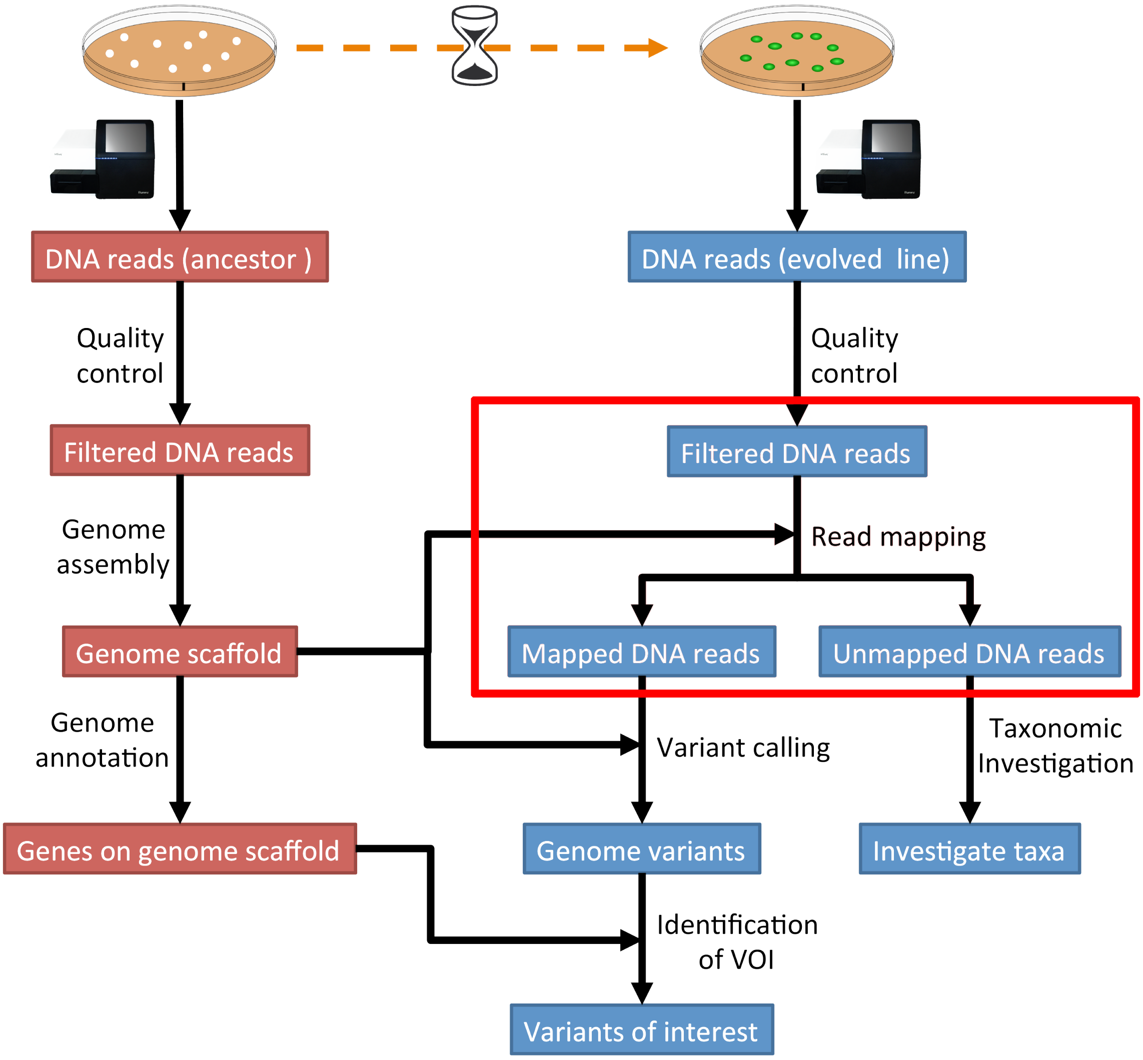

DNA reads (ancestor)
Quality control
Filtered DNA reads
Genome assembly
Genome scaffold
Genome annotation
Genes on genome scaffold
DNA reads (evolved line)
Quality control
Filtered DNA reads
Read mapping
Mapped DNA reads
Unmapped DNA reads
Variant calling
Taxonomic Investigation
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

Fig. 5.1: The part of the workflow we will work on in this section marked in red.

5.5 Mapping sequence reads to a reference genome

We want to map the sequencing reads to the ancestral reference genome we created in the section *NGS - Genome assembly* (page 21). We are going to use the quality trimmed forward and backward DNA sequences of the evolved line and use a program called BWA (http://bio-bwa.sourceforge.net/) to map the reads.

Todo

1. Discuss briefly why we are using the ancestral genome as a reference genome as opposed to a genome for the evolved line.

5.5.1 Installing the software

We are going to use a program called BWA (http://bio-bwa.sourceforge.net/) fo map our reads to a genome.

It is simple to install and use.

```
source activate ngs
conda install samtools
conda install bamtools
conda install bedtools
conda install bowtie2
conda install bwa
conda install picard
```

5.6 Bowtie2

5.6.1 Overview

Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) is a short read aligner, that can take a reference genome and map single- or paired-end data to it *[TRAPNELL2009]* (page 65). It requires an indexing step in which one supplies the reference genome and Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) will create an index that in the subsequent steps will be used for aligning the reads to the reference genome. The general command structure of the Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) tools we are going to use are shown below:

```
# bowtie2 help
bowtie2-build

# indexing
bowtie2-build genome.fasta PATH_TO_INDEX_PREFIX

# paired-end mapping
bowtie2 -X 1000 -x PATH_TO_INDEX_PREFIX -1 read1.fq.gz -2 read2.fq.gz -S aln-pe.sam
```

- `-X`: Adjust the maximum fragment size (length of paired-end alignments + insert size) to 1000bp. This might be useful if you do not know the exact insert size of your data. The Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) default is set to 500 which is often considered too short (http://lab.loman.net/2013/05/02/use-x-with-bowtie2-to-set-minimum-and-maximum-insert-sizes-for-nextera-libraries/).

5.6.2 Creating a reference index for mapping

Todo

Create an Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) index for our reference genome assembly. Attention! Remember which file you need to submit to Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml).

Hint: Should you not get it right, try the commands in *Code: Bowtie2 indexing* (page 58).

5.6.3 Mapping reads in a paired-end manner

Now that we have created our index, it is time to map the filtered and trimmed sequencing reads of our evolved line to the reference genome.

Todo

Use the correct `bowtie2` command structure from above and map the reads of the evolved line to the reference genome.

Hint: Should you not get it right, try the commands in *Code: Bowtie2 mapping* (page 58).

5.7 BWA

Attention: If the mapping did not succeed with Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml). We can use the aligner BWA (http://bio-bwa.sourceforge.net/) explained in this section. If the mapping with Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) did work, you can jump this section.

5.7.1 Overview

BWA (http://bio-bwa.sourceforge.net/) is a short read aligner, that can take a reference genome and map single- or paired-end data to it. It requires an indexing step in which one supplies the reference genome and BWA (http://bio-bwa.sourceforge.net/) will create an index that in the subsequent steps will be used for aligning the reads to the reference genome. The general command structure of the BWA (http://bio-bwa.sourceforge.net/) tools we are going to use are shown below:

```
# bwa index help
bwa index

# indexing
bwa index reference-genome.fa

# bwa mem help
bwa mem

# single-end mapping
bwa mem reference-genome.fa reads.fq > aln-se.sam

# paired-end mapping
bwa mem reference-genome.fa read1.fq read2.fq > aln-pe.sam
```

5.7.2 Creating a reference index for mapping

Todo

Create an BWA (http://bio-bwa.sourceforge.net/) index for our reference genome assembly. Attention! Remember which file you need to submit to BWA (http://bio-bwa.sourceforge.net/).

Hint: Should you not get it right, try the commands in *Code: BWA indexing* (page 58).

5.7.3 Mapping reads in a paired-end manner

Now that we have created our index, it is time to map the filtered and trimmed sequencing reads of our evolved line to the reference genome.

Todo

Use the correct `bwa mem` command structure from above and map the reads of the evolved line to the reference genome.

Hint: Should you not get it right, try the commands in *Code: BWA mapping* (page 58).

5.8 The sam mapping file-format

BWA (http://bio-bwa.sourceforge.net/) will produce a mapping file in sam-format. Have a look into the sam-file that was created by BWA (http://bio-bwa.sourceforge.net/). A quick overview of the sam-format can be found here (http://bio-bwa.sourceforge.net/bwa.shtml#4) and even more information can be found here (http://samtools.github.io/hts-specs/SAMv1.pdf). Briefly, first there are a lot of header lines. Then, for each read, that mapped to the reference, there is one line.

The columns of such a line in the mapping file are described in Table 5.1.

Table 5.1: The sam-file format fields.

Col	Field	Description
1	QNAME	Query (pair) NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIAGR	extended CIGAR string
7	MRNM	Mate Reference sequence NaMe ('=' if same as RNAME)
8	MPOS	1-based Mate POSition
9	ISIZE	Inferred insert SIZE
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

One line of a mapped read can be seen here:

```
M02810:197:000000000-AV55U:1:1101:10000:11540   83      NODE_1_length_1419525_cov_
↪15.3898        607378  60      151M    =       607100  -429    ␣
↪TATGGTATCACTTATGGTATCACTTATGGCTATCACTAATGGCTATCACTTATGGTATCACTTATGACTATCAGACGTTATTACTATCAGACGAT
↪FHGHHHHHGGGHHHHHHHHHHHHHHHHHHHGHHHHHHHHHHHGHHHHHGHHHHHHHHHGDHHHHHHHHGHHHHGHHHGHHHHHHFHHHHGHHHHIHH
↪NM:i:0  MD:Z:151        AS:i:151        XS:i:0
```

It basically defines, the read and the position in the reference genome where the read mapped and a quality of the map.

5.9 Mapping post-processing

5.9.1 Fix mates and compress

Because aligners can sometimes leave unusual SAM flag (http://bio-bwa.sourceforge.net/bwa.shtml#4) information on SAM records, it is helpful when working with many tools to first clean up read pairing information and flags with SAMtools (http://samtools.sourceforge.net/). We are going to produce also compressed bam output for efficient storing of and access to the mapped reads.

```
samtools fixmate -O bam mappings/evolved-6.sam mappings/evolved-6.fixmate.bam
```

- `-O bam`: specifies that we want compressed bam output

Attention: The step of sam to bam-file conversion might take a few minutes to finish, depending on how big your mapping file is.

We will be using the SAM flag (http://bio-bwa.sourceforge.net/bwa.shtml#4) information later below to extract specific alignments.

Hint: A very useful tools to explain flags can be found here (http://broadinstitute.github.io/picard/explain-flags.html).

Once we have bam-file, we can also delete the original sam-file as it requires too much space.

```
rm mappings/evolved-6.sam
```

5.9.2 Sorting

We are going to use SAMtools (http://samtools.sourceforge.net/) again to sort the bam-file into coordinate order:

```
# convert to bam file and sort samtools sort -O bam -o mappings/evolved-6.sorted.bam mappings/evolved-6.fixmate.bam
```

- `-o`: specifies the name of the output file.
- `-O bam`: specifies that the output will be bam-format

5.9.3 Remove duplicates

In this step we remove duplicate reads. The main purpose of removing duplicates is to mitigate the effects of PCR amplification bias introduced during library construction.

```
picard MarkDuplicates REMOVE_DUPLICATES=true METRICS_FILE=mappings/evolved-6.marked_dup_metrics.txt INPUT=mappings/evolved-6.sorted.bam OUTPUT=mappings/evolved-6.sorted.dedup.bam
```

Todo

Figure out what “PCR amplification bias” means.

5.10 Mapping statistics

5.10.1 Stats with SAMtools

Lets get an mapping overview:

samtools flagstat mappings/evolved-6.sorted.dedup.bam

Todo

Look at the mapping statistics and understand their meaning (https://www.biostars.org/p/12475/). Discuss your results. Explain why we may find mapped reads that have their mate mapped to a different chromosome/contig? Can they be used for something?

For the sorted bam-file we can get read depth for at all positions of the reference genome, e.g. how many reads are overlapping the genomic position.

samtools depth mappings/evolved-6.sorted.dedup.bam | gzip > mappings/evolved-6.depth.txt.gz

---

Todo

Extract the depth values for contig 20 and load the data into R, calculate some statistics of our scaffold.

---

zcat mappings/evolved-6.depth.txt.gz | egrep ‘^NODE_20_’ | gzip > mappings/NODE_20.depth.txt.gz

Now we quickly use some R (https://www.r-project.org/) to make a coverage plot for contig NODE20. Open a R (https://www.r-project.org/) shell by typing `R` on the command-line of the shell.

```
x <- read.table('mappings/NODE_20.depth.txt.gz', sep='\t', header=FALSE,  strip.
↪white=TRUE)

# Look at the beginning of x
head(x)

# calculate average depth
mean(x[,3])
# std dev
sqrt(var(x[,3]))

# mark areas that have a coverage below 20 in red
plot(x[,2], x[,3], col = ifelse(x[,3] < 20,'red','black'), pch=19, xlab='postion',␣
↪ylab='coverage')

# to save a plot
png('mappings/covNODE20.png', width = 1200, height = 500)
plot(x[,2], x[,3], col = ifelse(x[,3] < 20,'red','black'), pch=19, xlab='postion',␣
↪ylab='coverage')
dev.off()
```

The result plot will be looking similar to the one in Fig. 5.2

Todo

Look at the created plot. Explain why it makes sense that you find relatively bad coverage at the beginning and the end of the contig.

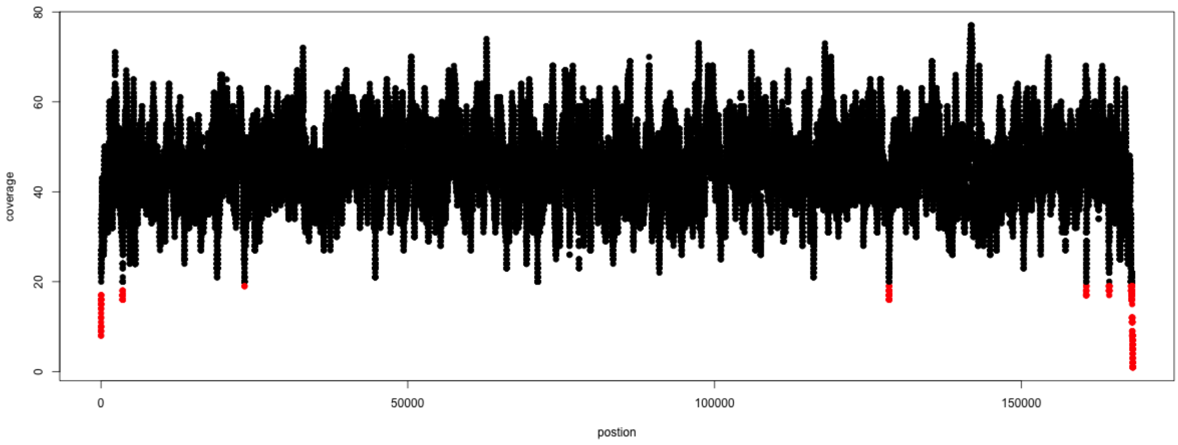


Fig. 5.2: A example coverage plot for a contig with highlighted in red regions with a coverage below 20 reads.

5.10.2 Stats with QualiMap

For a more in depth analysis of the mappings, one can use QualiMap (http://qualimap.bioinfo.cipf.es/).

QualiMap (http://qualimap.bioinfo.cipf.es/) examines sequencing alignment data in SAM/BAM files according to the features of the mapped reads and provides an overall view of the data that helps to the detect biases in the sequencing and/or mapping of the data and eases decision-making for further analysis.

Installation:

```
conda install qualimap
```

Run QualiMap (http://qualimap.bioinfo.cipf.es/) with:

qualimap bamqc -bam mappings/evolved-6.sorted.dedup.bam

This will create a report in the mapping folder. See this webpage (http://qualimap.bioinfo.cipf.es/doc_html/analysis.html#output) to get help on the sections in the report.

Todo

Install QualiMap (http://qualimap.bioinfo.cipf.es/) and investigate the mapping of the evolved sample. Write down your observations.

5.11 Sub-selecting reads

It is important to remember that the mapping commands we used above, without additional parameters to sub-select specific alignments (e.g. for Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) there are options like `--no-mixed`, which suppresses unpaired alignments for paired reads or `--no-discordant`, which suppresses discordant alignments for paired reads, etc.), are going to output all reads, including unmapped reads, multi-mapping reads, unpaired reads, discordant read pairs, etc. in one file. We can sub-select from the output reads we want to analyse further using SAMtools (http://samtools.sourceforge.net/).

Todo

Explain what concordant and discordant read pairs are? Look at the Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) manual.

5.11.1 Concordant reads

Here, we select the reads **we will be using for subsequent analyses**. Frist off, we select reads with a mapping quality of at least 20. Furthermore, we select read-pair that have been mapped in a correct manner (same chromosome/contig, correct orientation to each other).

```
samtools view -h -b -q 20 -f 2 mappings/evolved-6.sorted.dedup.bam > mappings/evolved-6.sorted.concordant.q20.bam
```

- `-h`: Include the sam header
- `-b`: Output will be bam-format
- `-q 20`: Only extract reads with mapping quality >= 20
- `-f 2`: Only extract correctly paired reads. `-f` extracts alignments with the specified SAM flag (http://bio-bwa.sourceforge.net/bwa.shtml#4) set.

Attention: The resulting file of this step will be used in the next section for calling variants.

5.11.2 Unmapped reads

We could decide to use Kraken (https://ccb.jhu.edu/software/kraken/) like in section *NGS - Taxonomic investigation* (page 37) to classify all unmapped sequence reads and identify the species they are coming from and test for contamination.

Lets see how we can get the unmapped portion of the reads from the bam-file:

```
samtools view -b -f 4 mappings/evolved-6.sorted.dedup.bam > mappings/evolved-6.sorted.unmapped.bam
```

```
# count them samtools view -c mappings/evolved-6.sorted.unmapped.bam
```

- `-b`: indicates that the output is BAM.
- `-f INT`: only include reads with this SAM flag (http://bio-bwa.sourceforge.net/bwa.shtml#4) set. You can also use the command `samtools flags` to get an overview of the flags.
- `-c`: count the reads

Lets extract the fastq sequence of the unmapped reads for read1 and read2.

```
bamToFastq -i evolved-6.sorted.unmapped.bam -fq mappings/evolved-6.sorted.unmapped.R1.fastq -fq2 mappings/evolved-6.sorted.unmapped.R2.fastq
```

CHAPTER

SIX

NGS - TAXONOMIC INVESTIGATION

6.1 Preface

We want to investate if there are sequences of other species in our collection of sequenced DNA pieces. We hope that most of them are from our species that we try to study, i.e. the DNA that we have extracted and amplified. This might be a way of quality control, e.g. have the samples been contaminated? Lets investigate if we find sequences from other species in our sequence set.

We will use the tool Kraken (https://ccb.jhu.edu/software/kraken/) to assign taxonomic classifications to our sequence reads. Let us see if we can id some sequences from other species.

Note: You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

6.2 Overview

The part of the workflow we will work on in this section can be viewed in Fig. 6.1.

6.3 Kraken

We will be using a tool called Kraken (https://ccb.jhu.edu/software/kraken/) *[WOOD2014]* (page 66). This tool uses k-mers to assign a taxonomic labels in form of NCBI Taxonomy (https://www.ncbi.nlm.nih.gov/taxonomy) to the sequence (if possible). The taxonomic label is assigned based on similar k-mer content of the sequence in question to the k-mer content of reference genome sequence. The result is a classification of the sequence in question to the most likely taxonomic label. If the k-mer content is not similar to any genomic sequence in the database used, it will not assign any taxonomic label.

6.4 Before we start

Lets see how our directory structure looks so far:

```
cd ~/analysis
ls -1F
```

```
assembly/
data/
mappings/
SolexaQA/
SolexaQA++
trimmed/
```

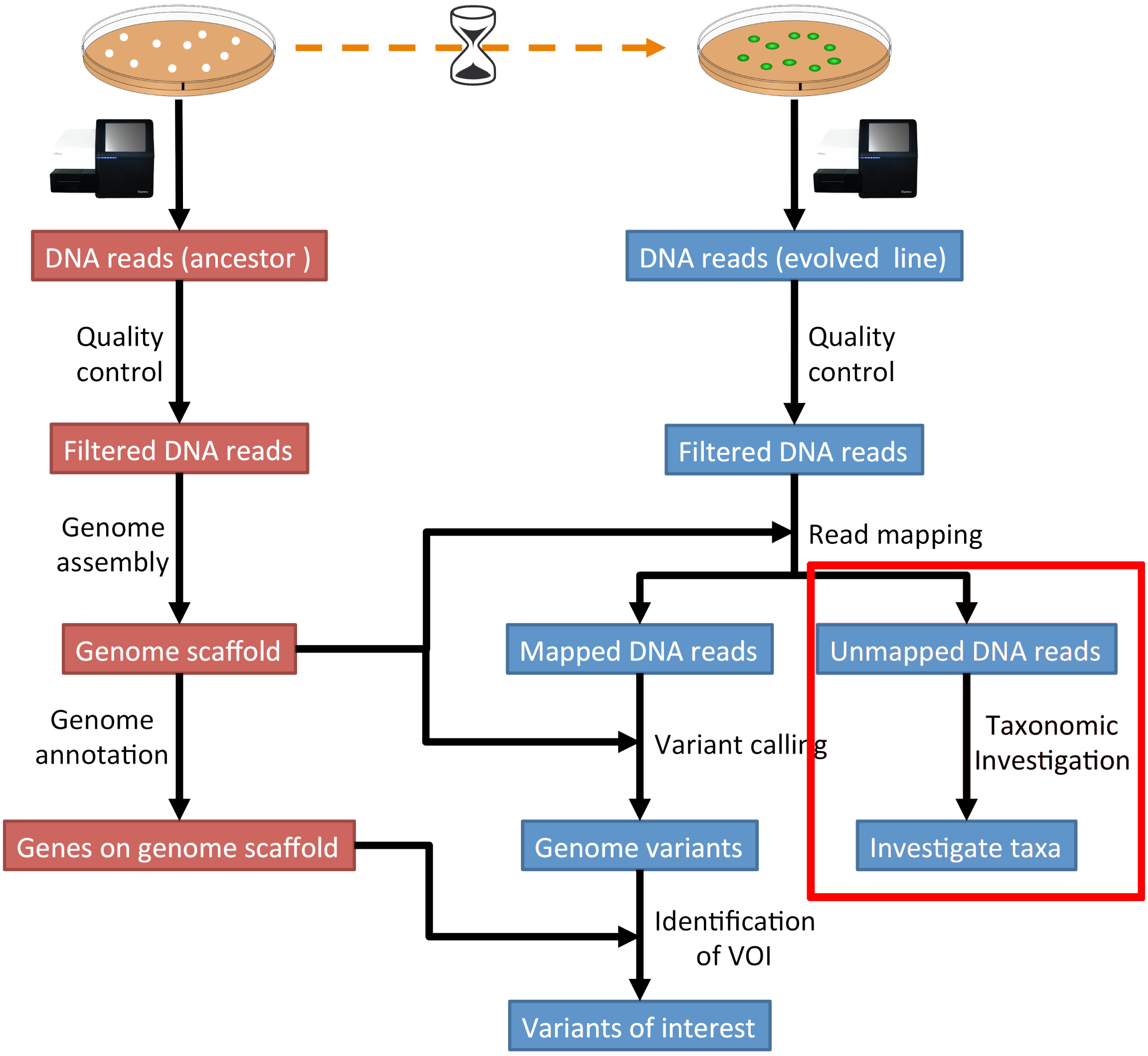

DNA reads (ancestor)
DNA reads (evolved line)
Quality control
Quality control
Filtered DNA reads
Filtered DNA reads
Genome assembly
Read mapping
Genome scaffold
Mapped DNA reads
Unmapped DNA reads
Genome annotation
Variant calling
Taxonomic Investigation
Genes on genome scaffold
Genome variants
Investigate taxa
Identification of VOI
Variants of interest

Fig. 6.1: The part of the workflow we will work on in this section marked in red.

```
trimmed-fastqc/
trimmed-solexaqa/
```

6.5 Installation

Use conda in the same fashion as before to install Kraken (https://ccb.jhu.edu/software/kraken/):

```
source activate ngs
conda install kraken-all
```

Now we create a directory where we are going to do the analysis and we will change into that directory too.

```
# make sure you are in your analysis root folder
cd ~/analysis

# create dir
mkdir kraken
cd kraken
```

Now we need to create or download a Kraken (https://ccb.jhu.edu/software/kraken/) database that can be used to assign the taxonomic labels to sequences. We opt for downloading the pre-build "minikraken" database from the Kraken (https://ccb.jhu.edu/software/kraken/) website:

```
curl -O https://ccb.jhu.edu/software/kraken/dl/minikraken.tgz

# alternatively we can use wget
wget https://ccb.jhu.edu/software/kraken/dl/minikraken.tgz

# once the download is finished, we need to extract the archive content
# it will create a directory: "minikraken_20141208/"
tar -xvzf minikraken.tgz
```

Attention: Should the download fail. Please find links to alternative locations on the *Downloads* (page 59) page.

Note: The "minikraken" database was created from bacteria, viral and archaea sequences. What are the implications for us when we are trying to classify our sequences?

6.6 Usage

Now that we have installed Kraken (https://ccb.jhu.edu/software/kraken/) and downloaded and extracted the minikraken database, we can attempt to investigate the sequences we got back from the sequencing provider for other species as the one it should contain. We call the Kraken (https://ccb.jhu.edu/software/kraken/) tool and specify the database and fasta-file with the sequences it should use. The general command structure looks like this:

kraken –only-classified-output –db minikraken_20141208 example.fa > example.kraken

However, we may have fastq-files, so we need to use `--fastq-input` which tells Kraken (https://ccb.jhu.edu/software/kraken/) that it is dealing with fastq-formated files. Here, we are investigating one of the unmapped paired-end read files of the evolved line.

kraken –only-classified-output –db minikraken_20141208 –fastq-input ../mappings/evolved-6-R1.sorted.unmapped.R1.fastq > evolved-6-R1.kraken

---

Note: We are for now only interested in the portion of sequeuces that can be classified, that is why we supplied the option `--only-classified-output` to the Kraken (https://ccb.jhu.edu/software/kraken/) command.

This classification may take a while, depending on how many sequences we are going to classify. The resulting content of the file "evolved-6-R1.kraken" looks similar to the following example:

```
C       M02810:197:000000000-AV55U:1:1101:10078:18384/1 2157    151     0:99 2157:1 0:1
C       M02810:197:000000000-AV55U:1:1101:10573:27304/1 364745  150     0:43 364745:1 0:
C       M02810:197:000000000-AV55U:1:1101:10852:5722/1  37665   151     0:33 37665:1 0:
C       M02810:197:000000000-AV55U:1:1101:11429:10330/1 374840  101     0:17 374840:1 0:
C       M02810:197:000000000-AV55U:1:1101:11705:24355/1 2157    151     0:1 2157:1 0:11
```

Each sequence classified by Kraken (https://ccb.jhu.edu/software/kraken/) results in a single line of output. Output lines contain five tab-delimited fields; from left to right, they are:

1. `C/U`: one letter code indicating that the sequence was either classified or unclassified.
2. The sequence ID, obtained from the FASTA/FASTQ header.
3. The taxonomy ID Kraken used to label the sequence; this is **0** if the sequence is unclassified and otherwise should be the NCBI Taxonomy (https://www.ncbi.nlm.nih.gov/taxonomy) identifier.
4. The length of the sequence in bp.
5. A space-delimited list indicating the lowest common ancestor (in the taxonomic tree) mapping of each k-mer in the sequence. For example, `562:13 561:4 A:31 0:1 562:3` would indicate that:
    - the first 13 k-mers mapped to taxonomy ID #562
    - the next 4 k-mers mapped to taxonomy ID #561
    - the next 31 k-mers contained an ambiguous nucleotide
    - the next k-mer was not in the database
    - the last 3 k-mers mapped to taxonomy ID #562

Note: The Kraken (https://ccb.jhu.edu/software/kraken/) manual can be accessed here (http://ccb.jhu.edu/software/kraken/MANUAL.html).

6.7 Investigate taxa

We can use the webpage NCBI TaxIdentifier (https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi) to quickly get the names to the taxonomy identifier. However, this is impractical as we are dealing potentially with many sequences. Kraken (https://ccb.jhu.edu/software/kraken/) has some scripts that help us understand our results better.

6.7.1 kraken-report

First, we generate a sample-wide report of all taxa found. This can be achieved with the tool `kranken-report`.

kraken-report –db minikraken_20141208 evolved-6-R1.kraken > evolved-6-R1.kraken.report

The first few lines of an example report are shown below.

```
  0.00  0       0       U       0       unclassified
100.00  2665    47      -       1       root
 54.56  1454    0       -       131567    cellular organisms
 38.50  1026    1023    D       2157        Archaea
```

```
0.08  2       0       P       651137        Thaumarchaeota
0.08  2       0       -       651142          unclassified Thaumarchaeota
0.08  2       0       G       1048752           Candidatus Caldiarchaeum
0.08  2       2       S       311458              Candidatus Caldiarchaeum subterraneu
0.04  1       0       P       28890         Euryarchaeota
0.04  1       0       C       183967          Thermoplasmata
```

The output of kraken-report is tab-delimited, with one line per taxon. The fields of the output, from left-to-right, are as follows:

1. **Percentage** of reads covered by the clade rooted at this taxon
2. **Number of reads** covered by the clade rooted at this taxon
3. **Number of reads** assigned directly to this taxon
4. A rank code, indicating **(U)nclassified, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies**. All other ranks are simply **"-"**.
5. NCBI Taxonomy (https://www.ncbi.nlm.nih.gov/taxonomy) ID
6. indented scientific name

Note: If you want to compare the taxa content of different samples to another, one can create a report whose structure is always the same for all samples, disregarding which taxa are found (obviously the percentages and numbers will be different).

We can cerate such a report using the option `--show-zeros` which will print out all taxa (instead of only those found). We then sort the taxa according to taxa-ids (column 5), e.g. `sort -n -k5`.

kraken-report –*show-zeros* –db minikraken_20141208 evolved-6-R1.kraken | **sort -n -k5** > evolved-6-R1.kraken.report.sorted

The report is not ordered according to taxa ids and contains all taxa in the database, even if they have not been found in our sample and are thus zero. The columns are the same as in the former report, however, we have more rows and they are now differently sorted, according to the NCBI Taxonomy (https://www.ncbi.nlm.nih.gov/taxonomy) id.

6.7.2 kraken-translate

For every sequence in our sample and its predicted taxonomic identifier, we can attach the taxonomic names with `kraken-translate`.

kraken-translate –mpa-format –db minikraken_20141208 evolved-6-R1.kraken > evolved-6-R1.kraken.names

An example output looks like this:

```
M02810:197:000000000-AV55U:1:1101:10078:18384/1 d__Archaea
M02810:197:000000000-AV55U:1:1101:10573:27304/1 d__Viruses|f__Baculoviridae|g__Betabacul
M02810:197:000000000-AV55U:1:1101:10852:5722/1  d__Viruses|f__Phycodnaviridae|g__Phaeovi
M02810:197:000000000-AV55U:1:1101:11429:10330/1 d__Viruses|f__Microviridae|g__Microvirus
M02810:197:000000000-AV55U:1:1101:11705:24355/1 d__Archaea
M02810:197:000000000-AV55U:1:1101:12206:13333/1 d__Viruses|o__Herpesvirales|f__Alloherpe
M02810:197:000000000-AV55U:1:1101:12336:11626/1 d__Viruses|f__Baculoviridae|g__Betabacul
M02810:197:000000000-AV55U:1:1101:12707:5809/1  d__Archaea
M02810:197:000000000-AV55U:1:1101:12741:21685/1 d__Viruses|f__Baculoviridae|g__Betabacul
M02810:197:000000000-AV55U:1:1101:12767:27821/1 d__Archaea
```

Here, each sequence that got classified is present in one row. The nomenclature for the names is preceded with an letter according to its rank, e.g. **(d)omain, (k)ingdom, (p)hylum, (c)lass, (o)rder, (f)amily (g)enus, or (s)pecies**. Taxonomy assignments above the superkingdom (**d**) rank are represented as just **root**.

6.7.3 Visualisation

We use the Krona (https://github.com/marbl/Krona/wiki) tools to create a nice interactive visualisation of the taxa content of our sample *[ONDOV2011]* (page 66). Fig. 6.2 shows an example (albeit an artificial one) snapshot of the visualisation Krona (https://github.com/marbl/Krona/wiki) provides. Fig. 6.2 is a snapshot of the interactive web-page similar to the one we try to create.

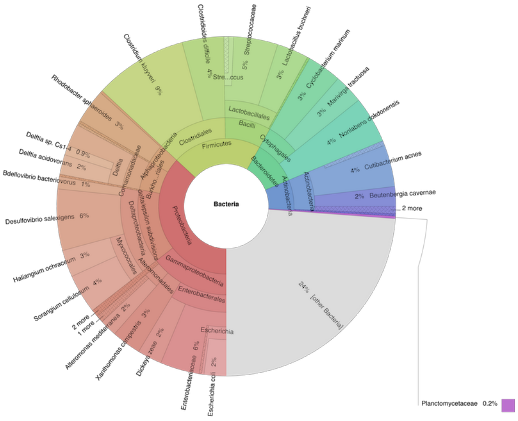


Fig. 6.2: Example of an Krona output webpage.

Install Krona (https://github.com/marbl/Krona/wiki) with:

```
source activate ngs
conda install krona
```

First some house-keeping to make the Krona (https://github.com/marbl/Krona/wiki) installation work. Do not worry to much about what is happening here.

```
# we delete a symbolic link that is not correct
rm -rf ~/miniconda3/envs/ngs/opt/krona/taxonomy

# we create a directory in our home where the krona database will live
mkdir -p ~/krona/taxonomy

# now we make a symbolic link to that directory
ln -s ~/krona/taxonomy ~/miniconda3/envs/ngs/opt/krona/taxonomy

# now we copy some scripts around, this is necessary as krona installation fails␣
↪to do this
cp -r ~/miniconda3/envs/ngs/opt/krona/scripts ~/miniconda3/envs/py3-kraken/bin/
```

We need to build a taxonomy database for Krona (https://github.com/marbl/Krona/wiki). However, if this fails we will skip this step and just download a pre-build one. Lets first try to build one.

```
ktUpdateTaxonomy.sh ~/krona/taxonomy
```

Now, if this fails, we download a pre-build taxonomy database for krona.

```
# Download pre-build database
curl -O http://compbio.massey.ac.nz/data/taxonomy.tab.gz

# we unzip the file
gzip -d taxonomy.tab.gz

# we move the unzipped file to our taxonomy directory we specified in the step␣
↪before.
mv taxonomy.tab ~/krona/taxonomy
```

Attention: Should this also fail we can download a pre-build database on the *Downloads* (page 59) page via a browser.

Now we use the tool `ktImportTaxonomy` from the Krona (https://github.com/marbl/Krona/wiki) tools to crate the html web-page:

```
cat evolved-6-R1.kraken | cut -f 2,3 > evolved-6-R1.kraken.krona ktImportTaxonomy evolved-6-R1.kraken.krona
firefox taxonomy.krona.html
```

What happens here is that we extract the second and third column from the Kraken (https://ccb.jhu.edu/software/kraken/) results. Afterwards, we input these to the Krona (https://github.com/marbl/Krona/wiki) script, and open the resulting web-page in a bowser. Done!

CHAPTER

SEVEN

NGS - VARIANT CALLING

7.1 Preface

In this section we will use our genome assembly based on the ancestor and call genetic variants in the evolved line *[NIELSEN2011]* (page 66).

Note: You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

7.2 Overview

The part of the workflow we will work on in this section can be viewed in Fig. 7.1.

7.3 Learning outcomes

After studying this tutorial section you should be able to:

#. Use tools to call variants based on a reference genome. #, Be able to describe what influences the calling of variants.

7.4 Before we start

Lets see how our directory structure looks so far:

```
cd ~/analysis
ls -1F
```

```
annotation/
assembly/
data/
kraken/
mappings/
SolexaQA/
SolexaQA++
trimmed/
trimmed-fastqc/
trimmed-solexaqa/
```

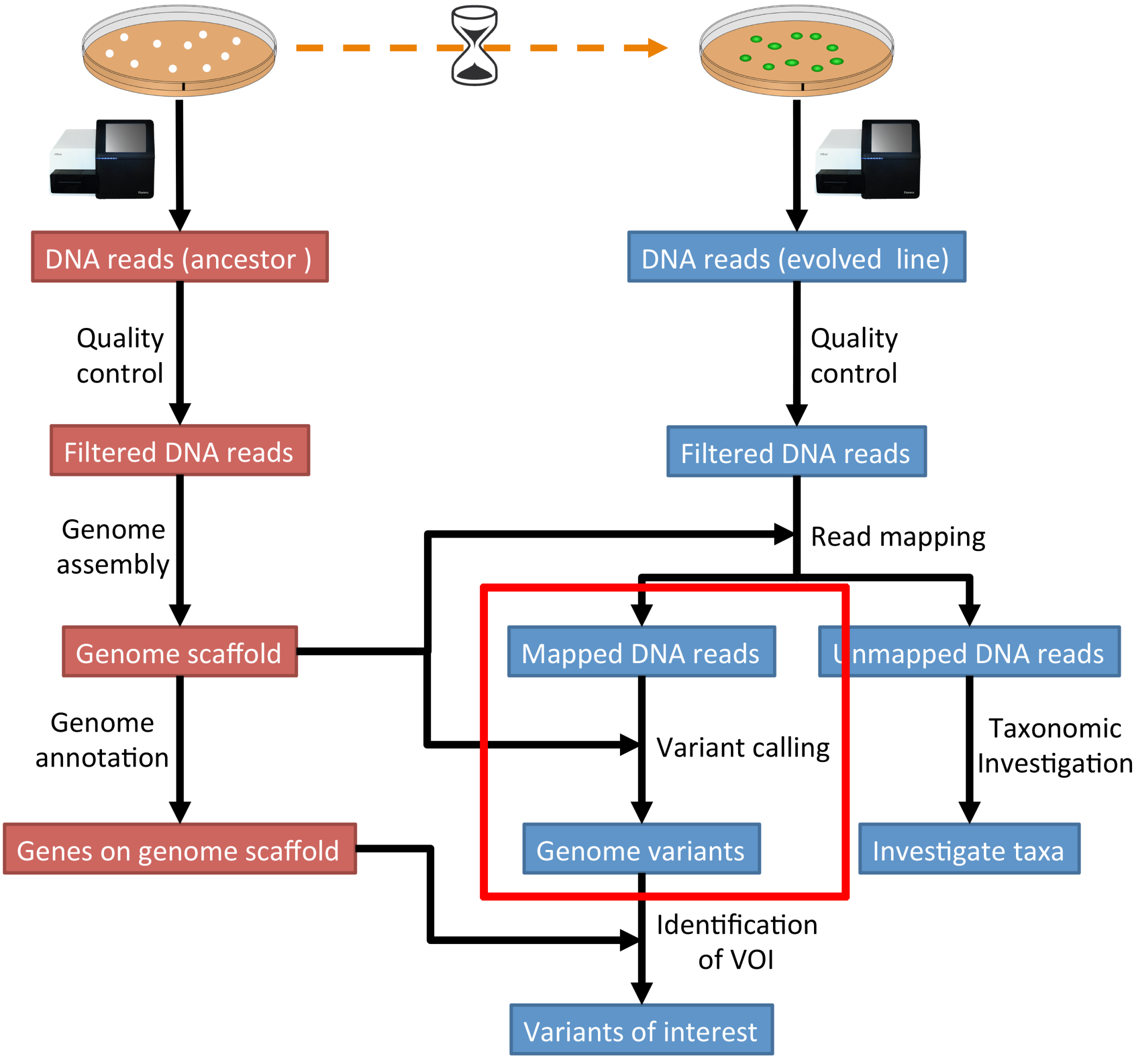

DNA reads (ancestor)
Quality control
Filtered DNA reads
Genome assembly
Genome scaffold
Genome annotation
Genes on genome scaffold
DNA reads (evolved line)
Quality control
Filtered DNA reads
Read mapping
Mapped DNA reads
Unmapped DNA reads
Variant calling
Genome variants
Taxonomic Investigation
Investigate taxa
Identification of VOI
Variants of interest

Fig. 7.1: The part of the workflow we will work on in this section marked in red.

7.5 Installing necessary software

Tools we are going to use in this section and how to intall them if you not have done it yet.

```
# activate the env
source activate ngs

# Install these tools into the conda environment
# if not already installed
conda install samtools
conda install bamtools
conda install bcftools
conda install freebayes
conda install rtg-tools
conda install bedtools
```

7.6 Preprocessing

We first need to make an index of our reference genome as this is required by the SNP caller. Given a scaffold/contig file in fasta-format, e.g. `scaffolds.fasta` which is located in the directory `assembly/spades_final`, use SAMtools (http://samtools.sourceforge.net/) to do this:

```
samtools faidx assembly/spades-final/scaffolds.fasta
```

Furthermore we need to pre-process our mapping files a bit further and create a bam-index file (`.bai`) for the bam-file we want to work with:

bamtools index -in mappings/evolved-6.sorted.concordant.q20.bam

Lets also create a new directory for the variants:

```
mkdir variants
```

7.7 Calling variants

7.7.1 SAMtools mpileup

We use the sorted filtered bam-file that we produced in the mapping step before.

# We first pile up all the reads and then call variants samtools mpileup -u -g -f assembly/spades-final/scaffolds.fasta mappings/evolved-6.sorted.concordant.q20.bam | bcftools call -v -m -O z -o variants/evolved-6.mpileup.vcf.gz

SAMtools (http://samtools.sourceforge.net/) mpileup parameter:

- `-u`: uncompressed output
- `-g`: generate genotype likelihoods in BCF format
- `-f FILE`: faidx indexed reference sequence file

BCFtools (http://www.htslib.org/doc/bcftools.html) view parameter:

- `-v`: output variant sites only
- `-m`: alternative model for multiallelic and rare-variant calling
- `-o`: output file-name
- `-O z`: output type: ‘z’ compressed VCF

7.7.2 Freebayes

As an alternative we can do some variant calling with another tool called freebayes (https://github.com/ekg/freebayes). Given a reference genome scaffold file in fasta-format, e.g. `scaffolds.fasta` and the index in `.fai` format and a mapping file (.bam file) and a mapping index (.bai file), we can call variants with freebayes (https://github.com/ekg/freebayes) like so:

# Now we call variants and pipe the results into a new file freebayes -f assembly/spades-final/scaffolds.fasta mappings/evolved-6.sorted.concordant.q20.bam | gzip > variants/evolved-6.freebayes.vcf.gz

7.8 Post-processing

7.8.1 Understanding the output files (.vcf)

Lets look at a vcf-file:

# first 10 lines, which are part of the header zcat variants/evolved-6.mpileup.vcf.gz | head

```
##fileformat=VCFv4.2
##FILTER=<ID=PASS,Description="All filters passed">
##samtoolsVersion=1.3.1+htslib-1.3.1
##samtoolsCommand=samtools mpileup -g -f assembly/spades-final/scaffolds.fasta -o␣
↪variants/evolved-6.mpileup.bcf mappings/evolved-6.sorted.concordant.q20.bam
##reference=file://assembly/spades-final/scaffolds.fasta
##contig=<ID=NODE_1_length_1419525_cov_15.3898,length=1419525>
##contig=<ID=NODE_2_length_1254443_cov_15.4779,length=1254443>
##contig=<ID=NODE_3_length_972329_cov_15.3966,length=972329>
##contig=<ID=NODE_4_length_951685_cov_15.4231,length=951685>
##contig=<ID=NODE_5_length_925222_cov_15.39,length=925222>
##contig=<ID=NODE_6_length_916533_cov_15.4426,length=916533>
```

Lets look at the variants:

# remove header lines and look at top 4 entires zcat variants/evolved-6.mpileup.vcf.gz | egrep -v '##' | head -4

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  mappings/
↪evolved-6.sorted.concordant.q20.bam
NODE_1_length_1419525_cov_15.3898      24721   .       T       C       164     .␣
↪      DP=12;VDB=0.205941;SGB=-0.680642;MQ0F=0;AC=2;AN=2;DP4=0,0,12,0;MQ=40    ␣
↪GT:PL   1/1:191,36,0
NODE_1_length_1419525_cov_15.3898      157033  .       AAGAGAGAGAGAGAGAGAGAGAGA ␣
↪      AAGAGAGAGAGAGAGAGAGAGA  39.3328 .       INDEL;IDV=6;IMF=0.146341;DP=41;
↪VDB=0.0813946;SGB=-0.616816;MQSB=1;MQ0F=0;ICB=1;HOB=0.5;AC=1;AN=2;DP4=13,17,3,3;
↪MQ=42     GT:PL   0/1:75,0,255
NODE_1_length_1419525_cov_15.3898      162469  .       T       C       19.609  .␣
↪      DP=16;VDB=0.045681;SGB=-0.511536;RPB=0.032027;MQB=0.832553;BQB=0.130524;
↪MQ0F=0;ICB=1;HOB=0.5;AC=1;AN=2;DP4=13,0,3,0;MQ=39        GT:PL   0/1:54,0,155
```

The fields in a vcf-file are described in he table (Table 7.1) below:

Table 7.1: The vcf-file format fields.

Col	Field	Description
1	CHROM	Chromosome name
2	POS	1-based position. For an indel, this is the position preceding the indel.
3	ID	Variant identifier. Usually the dbSNP rsID.
4	REF	Reference sequence at POS involved in the variant. For a SNP, it is a single base.
5	ALT	Comma delimited list of alternative seuqence(s).
6	QUAL	Phred-scaled probability of all samples being homozygous reference.
7	FILTER	Semicolon delimited list of filters that the variant fails to pass.
8	INFO	Semicolon delimited list of variant information.
9	FORMAT	Colon delimited list of the format of individual genotypes in the following fields.
10+	Sample(s)	Individual genotype information defined by FORMAT.

7.8.2 Statistics

Now we can use it to do some statistics and filter our variant calls.

First, to prepare out vcf-file for querying we need to index it with `tabix`:

tabix -p vcf variants/evolved-6.mpileup.vcf.gz

- `-p vcf`: input format

We can get some quick stats with `rtg vcfstats`:

rtg vcfstats variants/evolved-6.mpileup.vcf.gz

Example output from `rtg vcfstats`:

```
Location                      : variants/evolved-6.mpileup.vcf.gz
Failed Filters                : 0
Passed Filters                : 516
SNPs                          : 399
MNPs                          : 0
Insertions                    : 104
Deletions                     : 13
Indels                        : 0
Same as reference             : 0
SNP Transitions/Transversions: 1.87 (286/153)
Total Het/Hom ratio           : 3.20 (393/123)
SNP Het/Hom ratio             : 8.98 (359/40)
MNP Het/Hom ratio             : - (0/0)
Insertion Het/Hom ratio       : 0.30 (24/80)
Deletion Het/Hom ratio        : 3.33 (10/3)
Indel Het/Hom ratio           : - (0/0)
Insertion/Deletion ratio      : 8.00 (104/13)
Indel/SNP+MNP ratio           : 0.29 (117/399)
```

However, we can also run BCFtools (http://www.htslib.org/doc/bcftools.html) to extract more detailed statistics about our variant calls:

bcftools stats -F assembly/spades-final/scaffolds.fasta -s - variants/evolved-6.mpileup.vcf.gz > variants/evolved-6.mpileup.vcf.gz.stats

- `-s -`: list of samples for sample stats, “-” to include all samples
- `-F FILE`: faidx indexed reference sequence file to determine INDEL context

Now we take the stats and make some plots (e.g. Fig. 7.2) which are particular of interest if having multiple samples, as one can easily compare them. However, we are only working with one here:

mkdir variants/plots plot-vcfstats -p variants/plots/ variants/evolved-6.mpileup.vcf.gz.stats

- `-p`: The output files prefix, add a slash at the end to create a new directory.

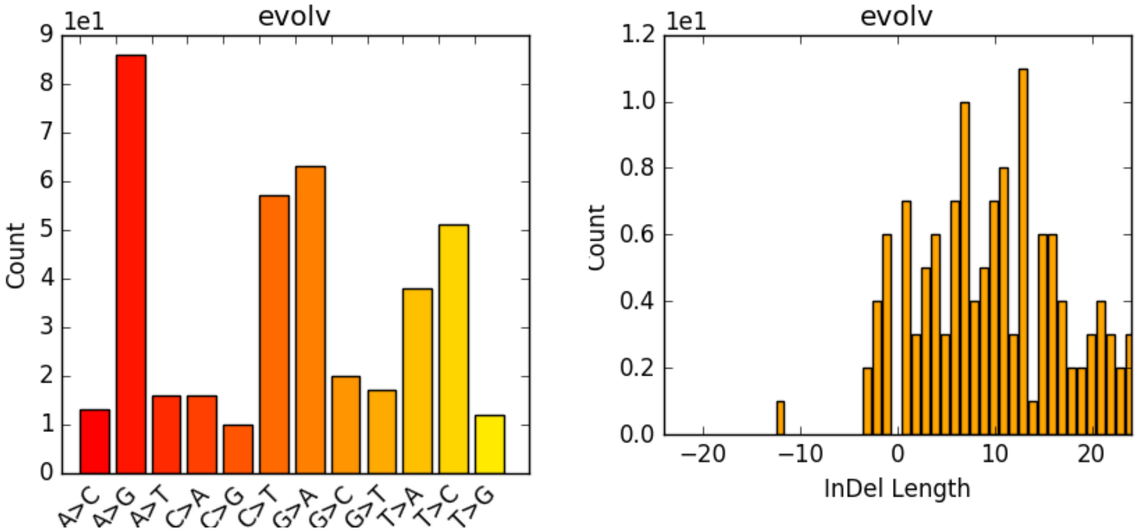


Fig. 7.2: Example of `plot-vcfstats` output.

7.8.3 Variant filtration

Variant filtration is a big topic in itself *[OLSEN2015]* (page 66). There is no consens yet and research on how to best filter variants is ongoing.

We will do some simple filtration procedures here. For one, we can filter out low quality reads.

Here, we only include variants that have quality > 30.

# use rtg vcfffilter rtg vcffilter -q 30 -i variants/evolved-6.mpileup.vcf.gz -o variants/evolved-6.mpileup.q30.vcf.gz

- `-i FILE`: input file
- `-o FILE`: output file
- `-q FLOAT`: minimal allowed quality in output.

or use BCFtools (http://www.htslib.org/doc/bcftools.html):

# or use bcftools bcftools filter -O z -o variants/evolved-6.mpileup.q30.vcf.gz -i'%QUAL>=30' variants/evolved-6.mpileup.vcf.gz # bcftools filter does not index output, so we need to do it again tabix -p vcf variants/evolved-6.mpileup.q30.vcf.gz

- `-i'%QUAL>=30'`: we only include variants that have been called with quality >= 30.

Quick stats for the filtered variants:

# look at stats for filtered rtg vcfstats variants/evolved-6.mpileup.q30.vcf.gz

---

Todo

Look at the statistics. One ratio that is mentioned in the statistics is transition transversion ratio (*ts/tv*). Explain what this ratio is and why the observed ratio makes sense.

This strategy used here will do for our purposes. However, several more elaborate filtering strategies have been explored, e.g. here (https://github.com/ekg/freebayes#observation-filters-and-qualities).

CHAPTER

EIGHT

NGS - GENOME ANNOTATION

Warning: THIS PART OF THE TUTORIAL IS CURRENTLY UNDER ACTIVE DEVELOPMENT SO EXPECT CONTENT TO CHANGE

CHAPTER

NINE

NGS - VARIANTS-OF-INTEREST

Warning: THIS PART OF THE TUTORIAL IS CURRENTLY UNDER ACTIVE DEVELOPMENT SO EXPECT CONTENT TO CHANGE

CHAPTER

TEN

QUICK COMMAND REFERENCE

10.1 Shell commands

```
# Where in the directory tree am I?
pwd

# List the documents and sub-directories in the current directory
ls

# a bit nicer listing with more information
ls -laF

# Change into your home directory
cd ~

# Change back into the last directory
cd -

# Change one directory up in the tree
cd ..

# Change explicitly into a directory "temp"
cd temp

# Quickly show content of a file "temp.txt"
# exist the view with "q", navigate line up and down with "k" and "j"
less temp.text

# Show the beginning of a file "temp.txt"
head temp.txt

# Show the end of a file "temp.txt"
tail temp.txt
```

10.2 General conda commands

```
# To update all packages
conda update --all --yes

# List all packages installed
conda list [-n env]

# conda list environments
conda env list

# create new env
```

```
conda create -n [name] package [package] ...

# activate env
source activate [name]

# deavtivate env
source deactivate
```

CHAPTER

ELEVEN

CODING SOLUTIONS

11.1 QC

11.1.1 Code: FastQC

Create directory:

```
mkdir trimmed-fastqc
```

Run FastQC:

```
fastqc -o trimmed-fastqc trimmed/ancestor-R1.fastq.trimmed.gz trimmed/ancestor-R2.fastq.trimmed.gz trimmed/evolved-6-R1.fastq.trimmed.gz trimmed/evolved-6-R2.fastq.trimmed.gz
```

Open html webpages:

```
firefox trimmed-fastqc/*.html
```

11.1.2 Code: SolexaQA++ trimming

Create directory for result-files:

```
mkdir trimmed
```

Run SolexaQA++:

```
./SolexaQA++ dynamictrim -p 0.01 -d trimmed/ data/ancestor-R1.fastq.gz
```

```
./SolexaQA++ dynamictrim -p 0.01 -d trimmed/ data/ancestor-R2.fastq.gz
```

```
./SolexaQA++ dynamictrim -p 0.01 -d trimmed/ data/evolved-6-R1.fastq.gz
```

```
./SolexaQA++ dynamictrim -p 0.01 -d trimmed/ data/evolved-6-R2.fastq.gz
```

11.1.3 Code: SolexaQA++ qc

Create directory for result-files:

```
mkdir trimmed-solexaqa/
```

Run SolexaQA++:

```
./SolexaQA++ analysis -d trimmed-solexaqa trimmed/ancestor-R1.fastq.trimmed.gz
```

```
./SolexaQA++ analysis -d trimmed-solexaqa trimmed/ancestor-R2.fastq.trimmed.gz
```

```
./SolexaQA++ analysis -d trimmed-solexaqa trimmed/evolved-6-R1.fastq.trimmed.gz
```

```
./SolexaQA++ analysis -d trimmed-solexaqa trimmed/evolved-6-R2.fastq.trimmed.gz
```

11.2 Assembly

11.2.1 Code: SPAdes assembly

```
spades.py -o assembly/spades-150/ -k 21,33,55,77 –careful -1 trimmed/ancestor-R1.fastq.trimmed.gz -2 trimmed/ancestor-R2.fastq.trimmed.gz
```

11.3 Mapping

11.3.1 Code: Bowtie2 indexing

Build the index:

```
bowtie2-build assembly/spades-final/scaffolds.fasta assembly/spades-final/scaffolds
```

11.3.2 Code: Bowtie2 mapping

Map to the genome. Use a max fragemnt length of 1000 bp:

```
bowtie2 -X 1000 -x assembly/spades-final/scaffolds -1 trimmed/evolved-6-R1.fastq.trimmed.gz -2 trimmed/evolved-6-R2.fastq.trimmed.gz -S mappings/evolved-6.sam
```

11.3.3 Code: BWA indexing

Index the genome assembly:

```
bwa index assembly/spades-final/scaffolds.fasta
```

11.3.4 Code: BWA mapping

Run bwa mem:

```
bwa mem assembly/spades-final/scaffolds.fasta trimmed/evolved-6-R1.fastq.trimmed.gz trimmed/evolved-6-R2.fastq.trimmed.gz > mappings/evolved-6.sam
```

CHAPTER

TWELVE

DOWNLOADS

12.1 Tools

- Miniconda installer [EXTERNAL (https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh) | MASSEY (http://compbio.massey.ac.nz/data/203341/Miniconda3-latest-Linux-x86_64.sh) | DROPBOX (https://www.dropbox.com/s/tz2wocdzjr4grdy/Miniconda3-latest-Linux-x86_64.sh?dl=0)]
- Minikraken database [EXTERNAL (http://ccb.jhu.edu/software/kraken/dl/minikraken.tgz) | MASSEY (http://compbio.massey.ac.nz/data/203341/minikraken.tgz) | DROPBOX (https://www.dropbox.com/s/lje0ykzdxtq3rpk/minikraken.tgz?dl=0)]
- Krona (https://github.com/marbl/Krona/wiki) taxonomy database [MASSEY (http://compbio.massey.ac.nz/data/203341/taxonomy.tab.gz) | DROPBOX (https://www.dropbox.com/s/cwf1qc5zyq65yvn/taxonomy.tab.gz?dl=0)]
- SolexaQA++ [MASSEY (http://compbio.massey.ac.nz/data/203341/SolexaQA.tar.gz) | DROPBOX (https://www.dropbox.com/s/r9a7hg0tlwe6pk4/SolexaQA.tar.gz?dl=0)]

12.2 Data

- Raw data-set [MASSEY (http://compbio.massey.ac.nz/data/203341/data.tar.gz) | DROPBOX (https://www.dropbox.com/s/70gcfqzrqugwcn5/data.tar.gz?dl=0)]
- Trimmed data-set [MASSEY (http://compbio.massey.ac.nz/data/203341/trimmed.tar.gz) | DROPBOX (https://www.dropbox.com/s/o6ioadoxfppbjrv/trimmed.tar.gz?dl=0)]
- Assembled data-set [MASSEY (http://compbio.massey.ac.nz/data/203341/assembly.tar.gz) | DROPBOX (https://www.dropbox.com/s/vlyn2fxgkiml5m8/assembly.tar.gz?dl=0)]

12.3 Software

```
conda install ea-utils
conda install sickle-trim
conda install fastqc

conda install spades
conda install quast

conda install samtools
conda install bamtools
conda install bedtools
conda install bowtie2
conda install bwa
conda install picard
conda install qualimap
conda install kraken-all
conda install krona

conda install bcftools
conda install freebayes
conda install rtg-tools
```

LIST OF FIGURES

1.1 The tutorial will follow this workflow. 4

3.1 The part of the workflow we will work on in this section marked in red. 8
3.2 Illustration of single-end (SE) versus paired-end (PE) sequencing. 9
3.3 SolexaQA++ example quality plot along reads of a bad MiSeq run 14
3.4 SolexaQA++ example histogram plot of a bad MiSeq run. 15
3.5 SolexaQA++ example cumulative plot of a bad MiSeq run. 15
3.6 SolexaQA++ example quality heatmap of a bad MiSeq run. 16
3.7 Quality score across bases. 17
3.8 Quality per tile. 18
3.9 GC distribution over all sequences. 19

4.1 The part of the workflow we will work on in this section marked in red. 22

5.1 The part of the workflow we will work on in this section marked in red. 28
5.2 A example coverage plot for a contig with highlighted in red regions with a coverage below 20 reads. 34

6.1 The part of the workflow we will work on in this section marked in red. 38
6.2 Example of an Krona output webpage. 42

7.1 The part of the workflow we will work on in this section marked in red. 46
7.2 Example of `plot-vcfstats` output. 50

LIST OF TABLES

5.1 The sam-file format fields. 31

7.1 The vcf-file format fields. 49

BIBLIOGRAPHY

[KAWECKI2012] Kawecki TJ et al. Experimental evolution. Trends in Ecology and Evolution (2012) 27:10 (http://dx.doi.org/10.1016/j.tree.2012.06.001)

[ZEYL2006] Zeyl C. Experimental evolution with yeast. FEMS Yeast Res, 2006, 685–691 (http://doi.org/10.1111/j.1567-1364.2006.00061.x)

[COX2010] Cox MP, Peterson DA and Biggs PJ. SolexaQA: At-a-glance quality assessment of Illumina (http://illumina.com) second-generation sequencing data. BMC Bioinformatics, 2010, 11:485. DOI: 10.1186/1471-2105-11-485 (http://doi.org/10.1186/1471-2105-11-485)

[GLENN2011] Glenn T. Field guide to next-generation DNA sequencers. Molecular Ecology Resources (2011) 11, 759–769 doi: 10.1111/j.1755-0998.2011.03024.x (http://doi.org/10.1111/j.1755-0998.2011.03024.x)

[KIRCHNER2014] Kirchner et al. Addressing challenges in the production and analysis of Illumina sequencing data. BMC Genomics (2011) 12:382 (http://doi.org/10.1186/1471-2164-12-382)

[MUKHERJEE2015] Mukherjee S, Huntemann M, Ivanova N, Kyrpides NC and Pati A. Large-scale contamination of microbial isolate genomes by Illumina PhiX control. Standards in Genomic Sciences, 2015, 10:18. DOI: 10.1186/1944-3277-10-18 (http://doi.org/10.1186/1944-3277-10-18)

[ROBASKY2014] Robasky et al. The role of replicates for error mitigation in next-generation sequencing. Nature Reviews Genetics (2014) 15, 56-62 (http://doi.org/10.1038/nrg3655)

[ABBAS2014] Abbas MM, Malluhi QM, Balakrishnan P. Assessment of de novo assemblers for draft genomes: a case study with fungal genomes. BMC Genomics. 2014;15 Suppl 9:S10. (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4290589/) doi: 10.1186/1471-2164-15-S9-S10. Epub 2014 Dec 8.

[COMPEAU2011] Compeau PE, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. Nat Biotechnol. 2011 Nov 8;29(11):987-91 (http://dx.doi.org/10.1038/nbt.2023)

[GUREVICH2013] Gurevich A, Saveliev V, Vyahhi N and Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics 2013, 29(8), 1072-1075 (http://bioinformatics.oxfordjournals.org/content/29/8/1072)

[NAGARAJAN2013] Nagarajan N, Pop M. Sequence assembly demystified. Nat Rev Genet. 2013 Mar;14(3):157-67 (http://dx.doi.org/10.1038/nrg3367)

[SALZBERG2012] Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M, Marçais G, Pop M, Yorke JA. GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Res. 2012 Mar;22(3):557-67 (http://genome.cshlp.org/content/22/3/557.full?sid=59ea80f7-b408-4a38-9888-3737bc670876)

[WICK2015] Wick RR, Schultz MB, Zobel J and Holt KE. Bandage: interactive visualization of de novo genome assemblies. Bioinformatics 2015, 10.1093/bioinformatics/btv383 (http://bioinformatics.oxfordjournals.org/content/early/2015/07/11/bioinformatics.btv383.long)

[TRAPNELL2009] Trapnell C, Salzberg SL. How to map billions of short reads onto genomes. Nat Biotechnol. (2009) 27(5):455-7. doi: 10.1038/nbt0509-455. (http://doi.org/10.1038/nbt0509-455)

[ONDOV2011] Ondov BD, Bergman NH, and Phillippy AM. Interactive metagenomic visualization in a Web browser. BMC Bioinformatics, 2011, 12(1):385. (http://www.ncbi.nlm.nih.gov/pubmed/21961884)

[WOOD2014] Wood DE and Steven L Salzberg SL. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biology, 2014, 15:R46. DOI: 10.1186/gb-2014-15-3-r46 (http://doi.org/10.1186/gb-2014-15-3-r46).

[NIELSEN2011] Nielsen R, Paul JS, Albrechtsen A, Song YS. Genotype and SNP calling from next-generation sequencing data. Nat Rev Genetics, 2011, 12:433-451 (http://doi.org/10.1038/nrg2986)

[OLSEN2015] Olsen ND et al. Best practices for evaluating single nucleotide variant calling methods for microbial genomics. Front. Genet., 2015, 6:235. (https://doi.org/10.3389/fgene.2015.00235)