

# 246.201 Systems and Models - Week 1

Sebastian Schmeier

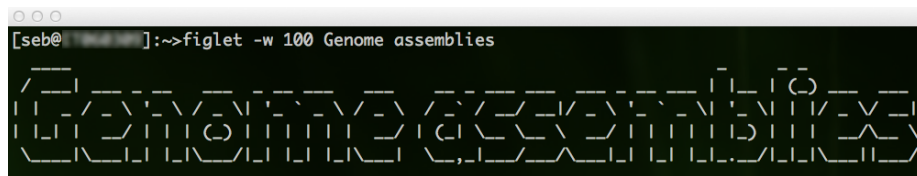
2014/09/16

## Contents

<b>246.201 Systems and Models - Week 1</b>	<b>2</b>
0. Learning outcomes . . . . .	2
1. Linux/Unix command-line intro . . . . .	2
1.1 Introduction . . . . .	3
1.2 Some words regarding the linux file-system . . . . .	3
1.3 Let's get started . . . . .	5
1.4 File-handling . . . . .	7
1.5 Investigate files . . . . .	8
1.6 Compression magic . . . . .	9
2. Get the data . . . . .	10
2.1 Investigate the data . . . . .	10
3. Quality assessment sequencing reads . . . . .	10
3.1 Download/install SolexaQA++ (hopefully not necessary) . . . . .	10
3.2 Understand SolexaQA++ . . . . .	11
3.2 Run SolexaQA++ on untrimmed data . . . . .	11
3.3 Dynamic trim the data . . . . .	14
3.4 Run SolexaQA++ on trimmed data . . . . .	14
3.5 Download/install FastQC (hopefully not necessary) . . . . .	15
3.6 Understand FastQC . . . . .	15
3.7 Run FastQC on the untrimmed and trimmed data . . . . .	16
4. Whole genome assembly . . . . .	18

4.1 Download/install Velvet (hopefully not necessary) . . . . .	19
4.2 Understanding Velvet . . . . .	19
4.3 Run Velvet with untrimmed data . . . . .	20
4.4 Run Velvet with trimmed data . . . . .	20
4.5 Evaluate assemblies . . . . .	20
5. What's next? . . . . .	21

## 246.201 Systems and Models - Week 1



### 0. Learning outcomes

1. Being able to operate comfortably the Linux command-line.
2. Being able to compute, investigate and evaluate the quality data from a sequencing experiment.
3. Being able to compute, interpret and evaluate a whole genome assembly.

### 1. Linux/Unix command-line intro

This tutorial is based on a Linux/Unix *command-line*. Using the *command-line* requires a Linux/Unix operating system. The easiest way to try out a Linux system without actually installing it on your computer is a [LiveCD](#). A LiveCD is a DVD that you prepare (e.g. burn a Linux distribution on it) and insert in your computer. You would restart your computer and can run Linux from the DVD without any installation requirements. This is helpful for trying out a distribution of Linux not for actual work.

Another route would be to use a virtual machine. Software to create a virtual machine is free, e.g. [VirtualBox](#).

Common flavors of Linux ready for download are e.g. [Ubuntu](#) or if you are thinking of going the bioinformatics route, [BioLinux](#), which includes many pre-installed bioinformatics tools.

## 1.1 Introduction

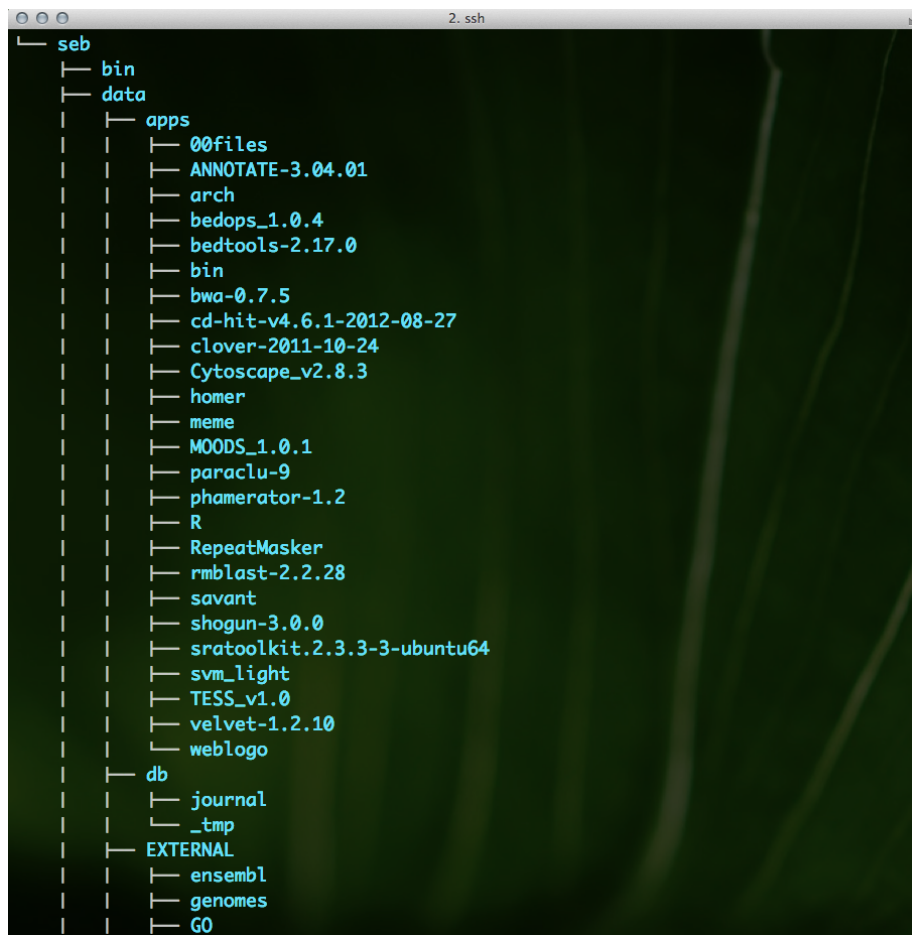
This is a collection of commands and programs I put together for working under Linux/Unix shells. It is not comprehensive. It includes very basic stuff. Tutorial style. This is bash syntax but most of it will work on other shells (tcsh, sh) as well.

You need a editor use nedit, gedit, emacs. Editing on the shell: emacs -nw or vi.

Hint! If you see a *grey* box, this means this is code and you can paste it into the command-line and hit “Enter” to run it. If you see a “#” at the start of a line, this denotes a comment.

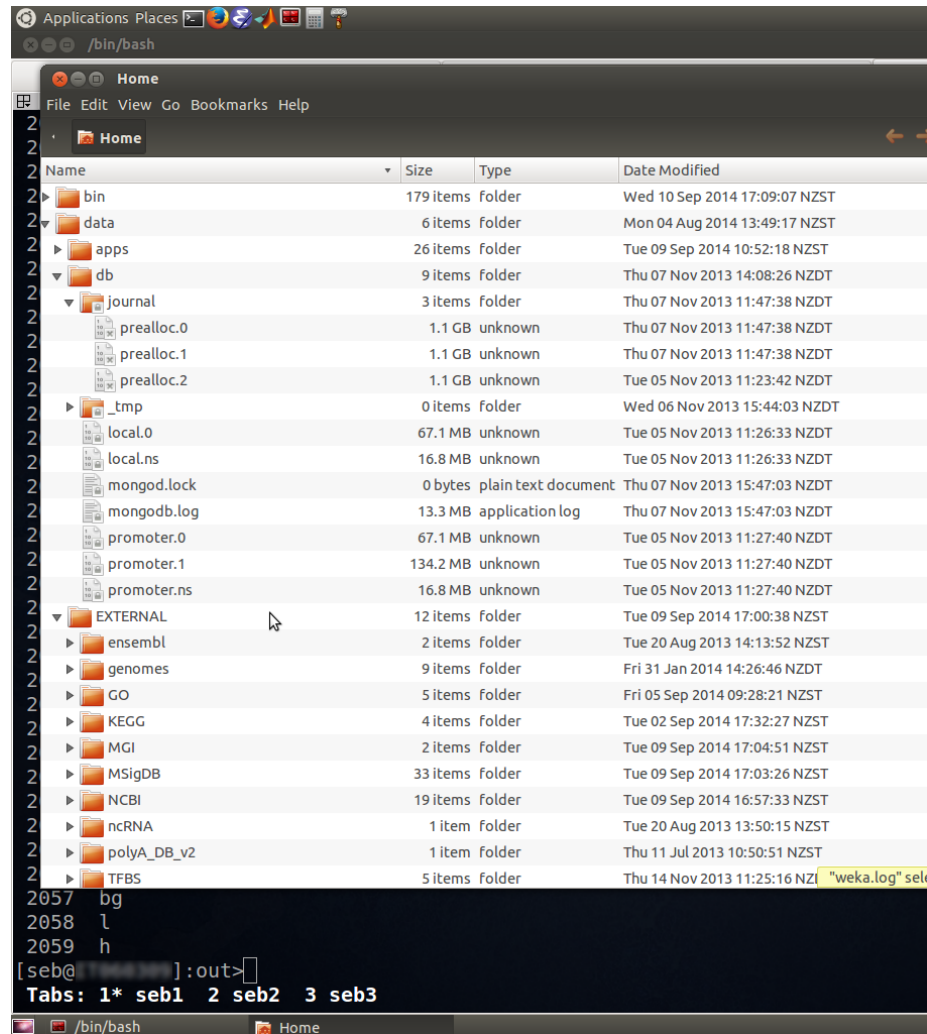
Open a terminal window and you are are ready to go.

## 1.2 Some words regarding the linux file-system



```
seb
├── bin
├── data
│   ├── apps
│   │   ├── 00files
│   │   ├── ANNOTATE-3.04.01
│   │   ├── arch
│   │   ├── bedops_1.0.4
│   │   ├── bedtools-2.17.0
│   │   ├── bin
│   │   ├── bwa-0.7.5
│   │   ├── cd-hit-v4.6.1-2012-08-27
│   │   ├── clover-2011-10-24
│   │   ├── Cytoscape_v2.8.3
│   │   ├── homer
│   │   ├── meme
│   │   ├── MOODS_1.0.1
│   │   ├── paraclu-9
│   │   ├── phamerator-1.2
│   │   ├── R
│   │   ├── RepeatMasker
│   │   ├── rmblast-2.2.28
│   │   ├── savant
│   │   ├── shogun-3.0.0
│   │   ├── sratoolkit.2.3.3-3-ubuntu64
│   │   ├── svm_light
│   │   ├── TESS_v1.0
│   │   ├── velvet-1.2.10
│   │   └── weblogo
│   ├── db
│   │   ├── journal
│   │   └── _tmp
│   └── EXTERNAL
│       ├── ensembl
│       ├── genomes
│       └── GO
```

The directory structure in a Linux system is not much different from an other system you worked with. It is essentially a tree structure. The way you navigate in the file-system can be via a file-manager e.g. Nautilus.



The difference is how you navigate the directory structure on the command-line. Why is this necessary? Strictly speaking it is not, if you do not want to make use of programs on the command-line. However, the power of the Linux system becomes only obvious once we learn to make use of the command-line, thus navigating the directory structure via commands is an important skill to know.

### 1.3 Let's get started

Help about a program e.g. 'ls':

```
man pwd
pwd -h
```

Another very helpful resource is the [explainshell.com](http://explainshell.com) webpage, that lets you write down a *command-line* to see the help text that matches each argument.

Investigate directory / list directory:

```
# what directory am I in?
pwd
# you should see something like /home/seb

# list the current directory elements implicitly
ls
# the same in a nicer format
ls -l
```

```

[seb@~]:~>ls -l
total 96604
drwxrwxr-x 2 seb seb 12288 Sep 10 17:09 bin
drwxrwxrwx 8 seb seb 4096 Aug 4 13:49 data
drwxr-xr-x 2 seb seb 4096 Aug 21 13:01 Desktop
drwxr-xr-x 3 seb seb 4096 Sep 24 2013 Documents
drwxr-xr-x 2 seb seb 4096 Jul 4 2013 Downloads
drwx----- 14 seb seb 4096 Sep 16 16:13 Dropbox
drwxrwxr-x 3 seb seb 4096 Oct 25 2013 gsea_home
drwxrwxr-x 3 seb seb 4096 Feb 5 2013 igv
drwxrwxr-x 3 seb seb 4096 Jul 29 16:52 Mail
drwxrwxr-x 3 seb seb 4096 Sep 23 2013 PERL
drwxrwxr-x 12 seb seb 4096 Jul 15 13:36 projects
drwxr-xr-x 2 seb seb 4096 Feb 1 2013 Public
drwxrwxr-x 4 seb seb 4096 Oct 17 2013 R
-rw-rw-r-- 1 seb seb 34104229 Sep 16 02:55 rsync_backup_it.err
-rw-rw-r-- 1 seb seb 63232963 Sep 16 02:55 rsync_backup_it.log
-rw-rw-r-- 1 seb seb 244 Sep 16 05:00 rsync_vm_natsci_2it.log
-rw-rw-r-- 1 seb seb 2585 Sep 16 05:00 rsync_vm_www-home_2it.log
-rw-rw-r-- 1 seb seb 1494632 Sep 16 16:14 Screenshot from 2014-09-16 16:14:28.png
drwxrwxr-x 20 seb seb 4096 Sep 16 15:20 temp
-rw-rw-r-- 1 seb seb 1997 Mar 14 2014 weka.log
[seb@~]:~>ls -l data/
total 24
drwxrwxrwx 27 seb seb 4096 Sep 9 10:52 apps
drwxrwxr-x 4 seb seb 4096 Nov 7 2013 db
drwxrwxrwx 14 seb seb 4096 Sep 9 17:00 EXTERNAL
drwxrwxr-x 3 seb seb 4096 Aug 18 15:51 mysql
drwxrwxrwx 4 seb seb 4096 Nov 25 2013 projects
drwxrwxr-x 3 seb seb 4096 Feb 2 2014 temp
[seb@~]:~>ls -l data/db
total 307908
drwxr-xr-x 2 root root 4096 Nov 7 2013 journal
-rw----- 1 root root 67108864 Nov 5 2013 local.0
-rw----- 1 root root 16777216 Nov 5 2013 local.ns
-rw-r--r-- 1 root root 13289399 Nov 7 2013 mongodb.log
-rwxr-xr-x 1 root root 0 Nov 7 2013 mongod.lock
-rw----- 1 root root 67108864 Nov 5 2013 promoter.0
-rw----- 1 root root 134217728 Nov 5 2013 promoter.1
-rw----- 1 root root 16777216 Nov 5 2013 promoter.ns
drwxr-xr-x 2 root root 4096 Nov 6 2013 _tmp
[seb@~]:~>ls -l data/db/journal/
total 3145740
-rw----- 1 root root 1073741824 Nov 7 2013 prealloc.0
-rw----- 1 root root 1073741824 Nov 7 2013 prealloc.1
-rw----- 1 root root 1073741824 Nov 5 2013 prealloc.2

```

### Moving around in the file system

```

# Lets create a directory
mkdir temp

# List a particular directory (e.g. temp/) explicitly
ls temp/

```

```
# change into directory "temp" with command "cd" (change directory)
cd temp/
pwd
# you should see something like /home/seb/temp

# Go one directory up in the directory tree
cd ..
pwd
# back to /home/seb

# Go to your home directory from any position in the directory tree
cd

# A shortcut for the home directory is ~/
# This command will change to /home/user/temp from any position in the directory tree
cd ~/temp
```

## 1.4 File-handling

Create a new empty text-file:

```
# first change into the temp directory
cd temp
# now create empty file
touch file1.txt
```

Delete a file (*caution*)

```
rm file1.txt
```

Warning! Avoid using "rm \*". This will erase all files in the directory.

Copy a file (file1.txt) to another location or file

```
# create empty file again
touch file1.txt
cp file1.txt file2.txt
```

Copy directories

```
cp -r dir1 dir2
# will not work because there is not a directory "dir1"
```

### Move a file/directory

```
# move files
mv file1.txt file2.txt
# move directories
mv dir1 dir2
# again will not work because we miss "dir1"
```

### Delete a dir:

```
rm -r temp/
```

## 1.5 Investigate files

Note! Download two sample-files [here](#) and [here](#).

### Look into files

```
less file1.txt
# move line down with "j", up with "k", you can get out of it with "q"
less file2.txt
```

### Print head/tail of files

```
# first 15 lines:
head -15 file1.txt
# last 15 lines:
tail -15 file1.txt
```

### Concatenate content of files -> will print on stdout:

```
cat file1.txt file2.txt
# all files starting with "file":
cat file*
# print content from one file to stdout:
cat file1.txt
```

### Count number of rows of a file:

```
wc -l file1.txt
```

### Sorting files



```
# sort on complete line:
sort file1.txt

# sort a comma-seperated file on second field:
sort -t ',' -k2,2 file1.txt

# sort a comma-seperated file on second field according to numbers
sort -t ',' -k2,2n file1.txt
```

### Extract columns of a file

```
# cut -d'seperator' -fCOLUMN,COLUMN,... file.txt, e.g.
cut -d ',' -f 1,3-5 file1.txt
```

### Search for pattern in a file. grep/egrep

```
# print only lines of a file that contain a pattern:
grep 'AAA' file1.txt

# print only lines that do _not_ contain the pattern:
grep -v 'AAA' file1.txt
```

## 1.6 Compression magic

First we compress a single file:

```
gzip file1.txt
# will produce a file called file1.txt.gz in gzip format, and delete file1.txt
```

We do not need to decompress a file to use look at its content (most of my text files are stored in gzip format):

```
zless file1.txt.gz
zcat file1.txt.gz
zcat file1.txt.gz
```

Extract a single zipped-file:

```
gzip -d file1.txt.gz
```

Compress using zip:

```
zip file.zip file1.txt
```

Extract a zipped-file:

```
unzip file1.zip
```

## 2. Get the data

You can download the data-file [here](#). I will also bring the data on a USB drive, please copy it onto your system should the download not work.

The data is a down-sampled (randomly selected) small portion of the original sequencing data-set. This has been done because the amount of data produced was too high for this exercises today. Also, the original data was paired-end data, thus we had two files, one for each end. The paired-data here was already combined into one file.

### 2.1 Investigate the data

Make use of your newly developed skills on the command-line to investigate the files in two folders.

**To-do:**

1. Unzip the data using `gzip`.
2. What kind of files are we dealing with?
3. How many sequence reads are in the file?

## 3. Quality assessment sequencing reads

To assess the sequence read quality of the Illumina run we make use of a program called [SolexaQA](#). This was originally developed to work with Solexa data (since bought by Illumina), but long since working with Illumina 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.

### 3.1 Download/install SolexaQA++ (hopefully not necessary)

Download [SolexaQA](#) from the developer webpage [here](#).

Note! Emergency link [here](#).

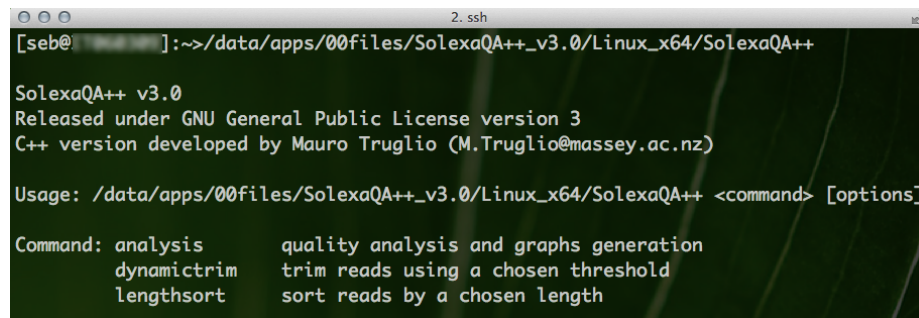
```
unzip SolexaQA++_v3.0.zip
chmod a+x SolexaQA++_v3.0/Linux_x64/SolexaQA++
./SolexaQA++_v3.0/Linux_x64/SolexaQA++
# should now run the program
```

## 3.2 Understand SolexaQA++

SolexaQA++ has three modes that can be run. Type:

**SolexaQA++**

This results in:



```
[seb@ ~]$:~/data/apps/00files/SolexaQA++_v3.0/Linux_x64/SolexaQA++
SolexaQA++ v3.0
Released under GNU General Public License version 3
C++ version developed by Mauro Truglio (M.Truglio@massey.ac.nz)

Usage: /data/apps/00files/SolexaQA++_v3.0/Linux_x64/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, 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.

## 3.2 Run SolexaQA++ on untrimmed data

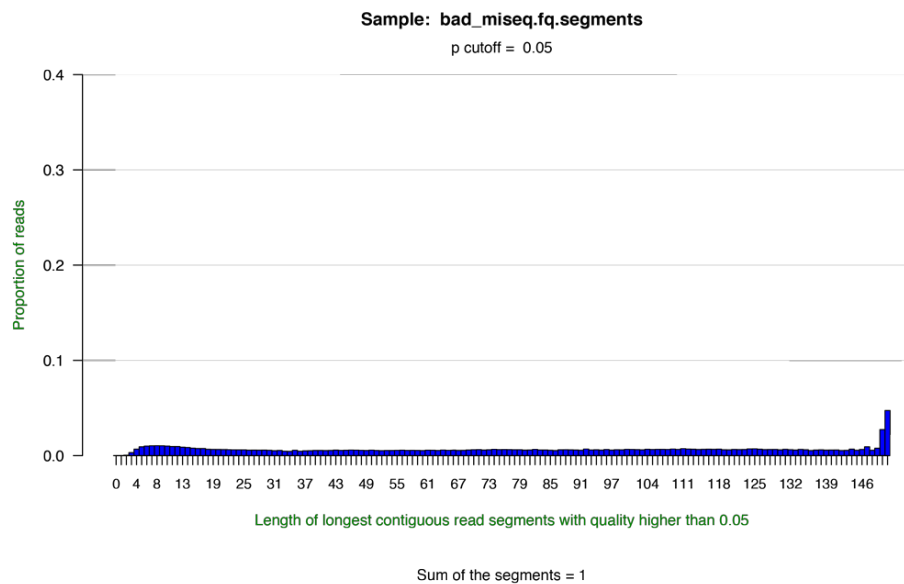
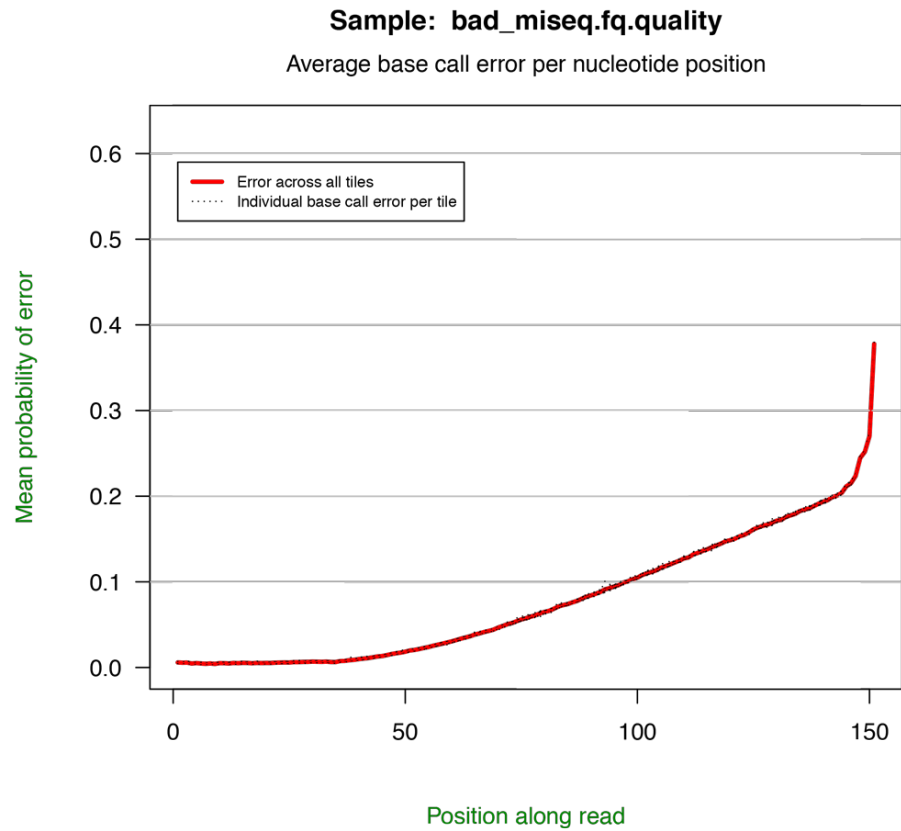
**To-do:**

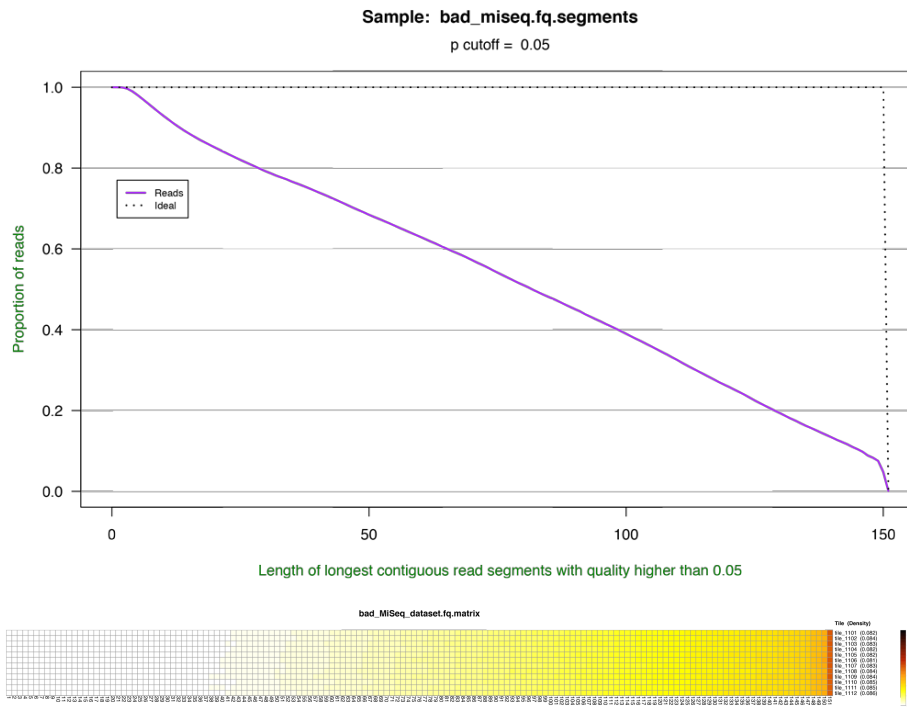
1. Create a directory for the result-files -> **qa\_untrimmed/**
2. Run SolexaQA++ with the untrimmed data, and submit result-directory **qa\_untrimmed/**.
3. Investigate the result-files in **qa\_untrimmed/**.

Hint! Should you not get 1 and/or 2 it right, try these commands [here](#).

Compare your results to these examples of a particularly bad run (taken from <http://solexaqa.sourceforge.net/>).

**What can we say about our data?**





### 3.3 Dynamic trim the data

Despite what you may have found out about the untrimmed data, it is a good idea to trim the data before further analyses.

#### To-do:

1. Create a directory for the result-files -> `qa_toTrimmed/`
2. Use SolexaQA++ to trim the reads based on quality and a probability cutoff of 0.01.

Hint! Should you not get 1 and/or 2 it right, try these commands [here](#).

### 3.4 Run SolexaQA++ on trimmed data

#### To-do:

1. Create a directory for the result-files -> `qa_trimmed/`.
2. Do the quality assessment again with the trimmed data-set.

3. Compare the results in `qa_trimmed/` to the untrimmed results in `qa_untrimmed/`.

Hint! Should you not get 1 and/or 2 it right, try these commands [here](#).

### 3.5 Download/install FastQC (hopefully not necessary)

Download [FastQC](#) from the developer webpage [here](#).

Note! Emergency link [here](#).

```
unzip fastqc_v0.11.2.zip
./FastQC/fastqc --help
# should now run the program
```

### 3.6 Understand FastQC

FastQC is a very simple program to run that provides similar and additional information to SolexaQA++.

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.

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

### 3.7 Run FastQC on the untrimmed and trimmed data

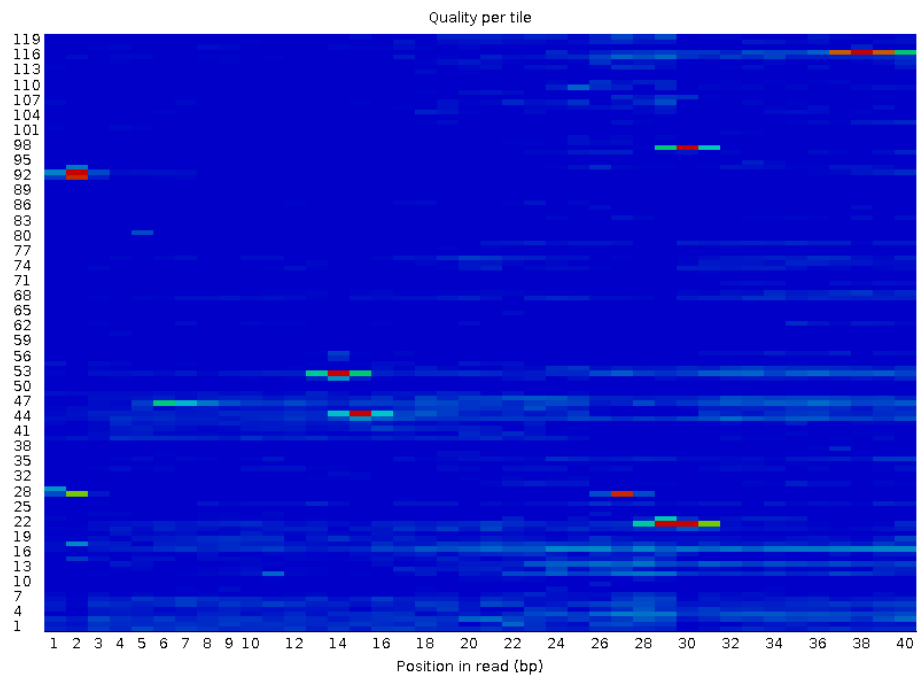
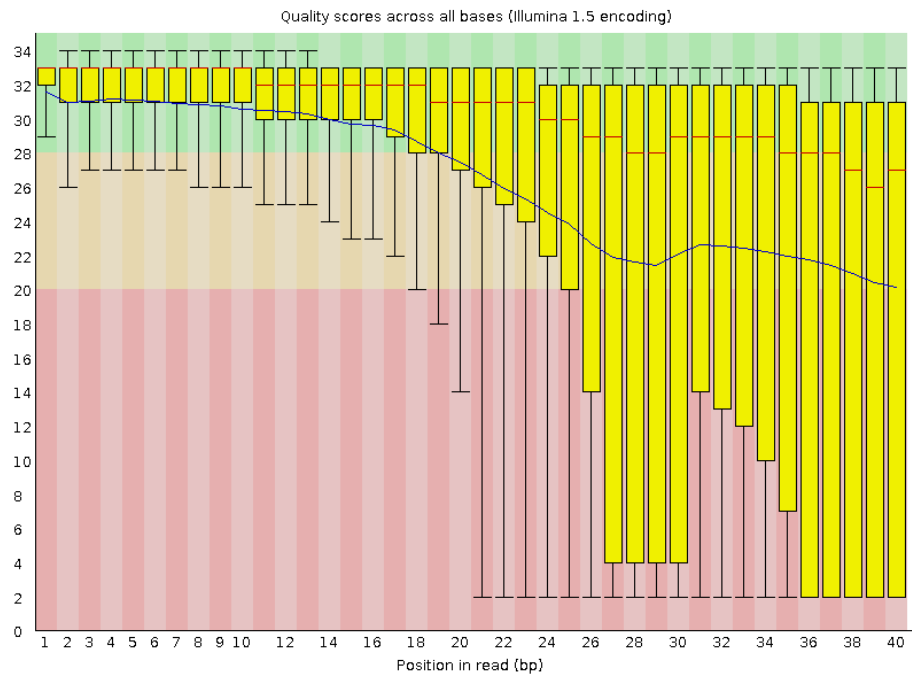
#### To-do

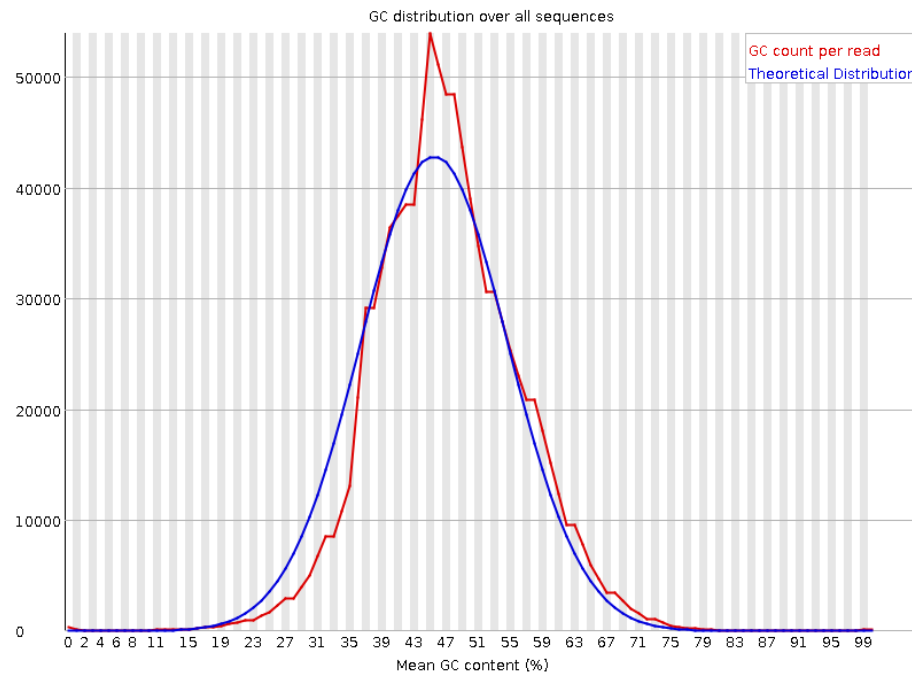
1. Create a directory for the results -> **fastqc/**
2. Run FastQC with both files
3. Compare the two result-files in a browser

Hint! Should you not get it right, try these commands [here](#).

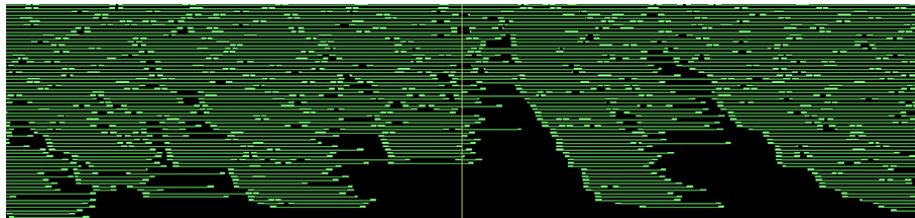
Compare your results to these examples of a particularly bad run (taken from <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>).







#### 4. Whole genome assembly



[Velvet](#) is a de Bruijn graph-based assembly program. It is developed with the aim of assembling very short reads like in our case.

From the webpage:

Velvet is a de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454.

Velvet currently takes in short read sequences, removes errors then produces high quality unique contigs. It then uses paired-end read and long read information, when available, to retrieve the repeated areas between contigs.

## 4.1 Download/install Velvet (hopefully not necessary)

Download [Velvet](#) from the developer website [here](#).

Note! Emergency download-link [here](#).

```
# unzip
gunzip velvet_1.2.10.tgz
# extract the archive
tar xvf velvet_1.2.10.tar
# change directory
cd velvet_1.2.10/
# compile program
make 'MAXKMERLENGTH=63'
# this will take a few seconds
# we have now two programs: velveth and velvetg
```

## 4.2 Understanding Velvet

Velvet uses two different programs sequentially to achieve the assembly. You have to use them one at a time. You should familiarize yourself with the structure of the program calls. Once you understand this it is very easy to change things around.

### velveth

This program is a hashing program. It basically prepares your data for the assembler **velvetg**. An example program call looks like:

```
# Running velveth without parameters gives you a help page
./velveth
# Here a complete example call
./velveth results_directory/ 63 -fastq -shortPaired input_seqs.fastq
```

- **./velveth** the program including the full path where the program is located
- **result\_directory/** the directory where you want the results to be saved
- **63** the kmer value
- **-fastq** specifies the input sequence-file type
- **-shortPaired** the type of the reads (here short paired reads)
- **input\_seqs.fastq** the file-name of the input data

### velvetg

This program is the core of velvet and the actual assembler. An example program call looks like this:

```
./velvetg result_directory/ -cov_cutoff 10 -exp_cov 30 -min_contig_lgth 500 -ins_length 300
```

- `./velvetg` the program including the full path where the program is located
- `result_directory/` the directory where you want the results to be saved
- `-cov_cutoff 10` removal of low coverage nodes AFTER tour bus or allow the system to infer it [do not worry about this]
- `-exp_cov 30` the expected coverage of unique regions or allow the system to infer it [do not worry about this]
- `-min_contig_lgth 500` the minimum size of contig to report back (to exclude tiny ones we only look at contigs >500bp)
- `-ins_length 300` expected distance between two paired-end reads in the respective short-read dataset

### 4.3 Run Velvet with untrimmed data

#### To-do:

1. Make sure you have the untrimmed data and you know where it is.
2. Create a directory for the results -> `velvet_untrimmed/`
3. Run velvet with the data `eli.low10paired.fastq`
4. Run velvetg

Hint! Should you not get it right, try these commands [here](#).

### 4.4 Run Velvet with trimmed data

#### To-do:

1. Make sure you have the trimmed data and you know where it is.
2. Create a directory for the results -> `velvet_trimmed/`
3. Run velvet with the data `qa_toTrimmed/eli.low10paired.fastq.trimmed`
4. Run velvetg

Hint! Should you not get it right, try these commands [here](#).

### 4.5 Evaluate assemblies

#### To-do:

1. Look at the Log files from trimmed and untrimmed assemblies.

2. Look at the stat.txt files from trimmed and untrimmed assemblies.
3. Look at the contigs.fa files of the trimmed and untrimmed assemblies.
4. What can we say about the the assemblies?
5. How does untrimmed and trimmed compare?
6. What can you say about the trimming procedure in light of assembling sequences?

## 5. What's next?

Next steps could include:

- map all reads to the “new” genome
- look at the aligned reads in a genome viewer
- predict genes in the “new” genome
- overlay gene information with aligned reads in the genome viewer
- etc.

*File: index.md - Sebastian Schmeier - Last update: Tue Sep 16 18:21:47 NZST 2014*