Genomics and Transcriptomics

Class 04 - Bash scripting and read processing

INSTRUCTOR:
Aureliano Bombarely
Department of Bioscience
Universita degli Studi di Milano
aureliano.bombarely@unimi.it

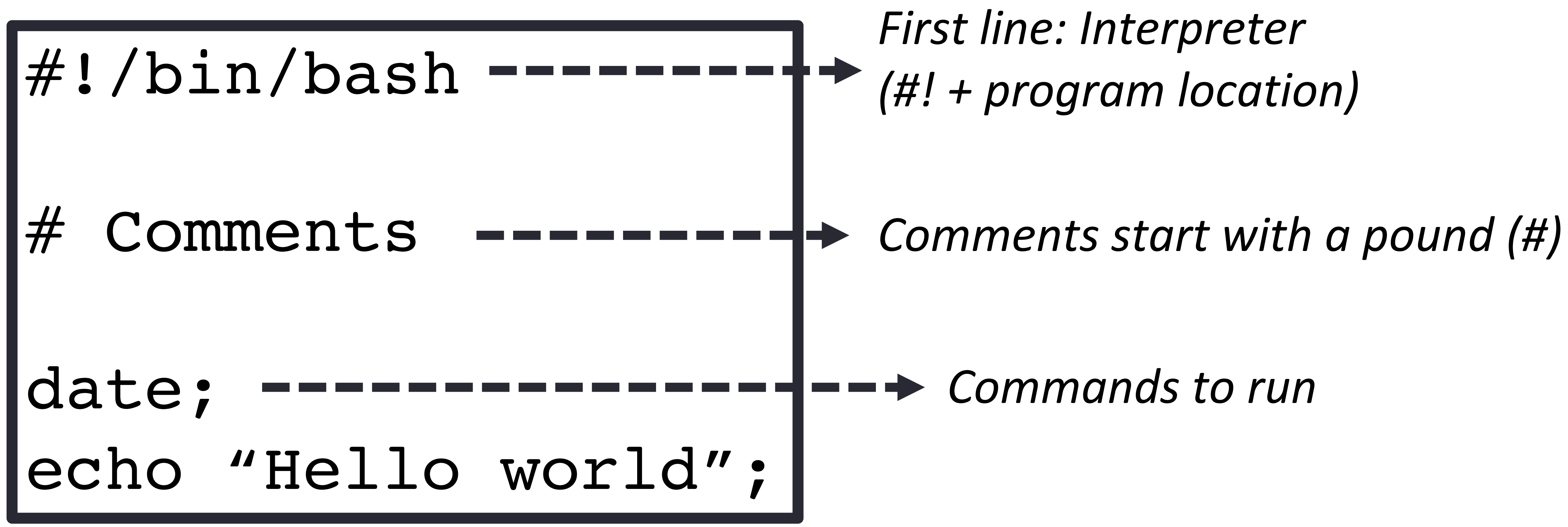
Outline of Topics

Outline of Topics

1.1. Basics about bash scripting

What is a script ?

A script (computing) is a small non-compiled program written for a scripting language or command interpreter. E.g. Bash scripts are interpreted by Bash (terminal), Perl scripts are interpreted by Perl…



```
#!/bin/bash

# Comments

date;
echo “Hello world”;
```

First line: Interpreter (#! + program location)
Comments start with a pound (#)
Commands to run

1.1. Basics about bash scripting

How to run a script

- If it is executable (check permisions with ls -lh) you can just run it as ./myscript.sh. You can do it executable with chmod 755 myscript.sh.

```
chmod 755 myscript.sh
./myscript.sh
```

- If it is not executable, you can use the interpreter (e.g. bash myscript.sh)

```
bash myscript.sh
```

1.1. Basics about bash scripting

What is a bash script ?

Usually is a script that is going to be interpreted by the Linux shell (commonly bash).

Because it is going to be interpreted by the bash you can run any command from your system (e.g. cd, pwd…).

1.1. Basics about bash scripting

Remember, special characters have special meanings

Character	Meaning
SPACE	Separate commands and arguments
# HASH	Comment
; SEMICOLON	Command separator to run multiple commands
. DOT	Source command OR filename component OR current directory
.. DOUBLE DOTS	Parent directory
‘ SINGLE QUOTES	Use expression between quotes
, COMMA	Concatenate strings
\ BACKSLASH	Escape for single character
/ SLASH	Filename path separator
* ASTERISK	Wildcard for filename expansion
>,<,>> CHARACTERS	Redirection input/outputs
| PIPE	Pipe outputs between commands
! BANG	Run a command

Commands to use in Bash scripts

- Software specific (e.g. cufflinks)
- Basic Linux/Unix command (e.g. echo)

COMMAND	USE	EXAMPLE
grep	Search a pattern and print it	grep ‘A’ file.txt
cut	Get sections of a file by column	cut -f1, 3 file.txt
wc	Count characters and lines	wc -l file.txt
sed	Execute a script over the file (e.g., replace)	sed -r ’s/A/a’ file.txt
sort	Sort lines of a text file	sort file.txt
uniq	Find unique lines	uniq -c file.txt
paste	Concatenates files as columns	paste *file1.txt file2.txt*

1.1. Basics about bash scripting

Write a Bash script to print date and working dir.

1. Open a file with the text editor (nano)

```
nano script01.sh
```

2. Write the first line specifying the interpreter (#!/bin/bash)

```
#!/bin/bash
```

3. Add the commands that are going to be executed
    3.1. **date** to print the date
    3.2. **pwd** to print the working directory.

```
#!/bin/bash

date;
pwd;
```

Write a Bash script to print date and working dir.

4. Save the file and exit (CTR+O; CTR+X).

5. Change the permissions of the script.

```
chmod755 script01.sh
```

6. Execute the script

```
./script01.sh
```

1.1. Basics about bash scripting

Exercise 4.1 - Write a Bash script and execute to

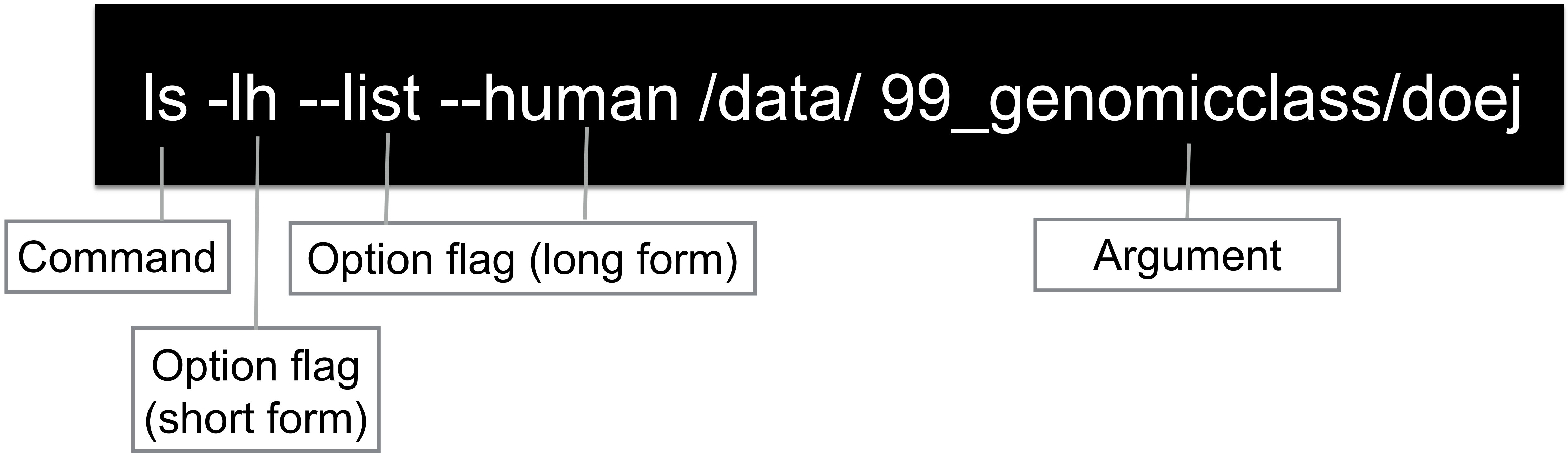
1. **Download** the following file: ftp://ftp.solgenomics.net/genomes/Solanum_lycopersicum/Heinz1706/annotation/ITAG4.1_release/ITAG4.1_proteins.fasta

2. **Count** how many reads are in the file.

3. **Calculate** the total number of aminoacids in the file (Non-ID characters) using grep and wc (check the manuals if necessary)

Outline of Topics

1.2. Arguments



```
ls -lh --list --human /data/ 99_genomicclass/doej
```

Arguments can be used inside the script with the default variables $1, $2, $3….

```
Itagreetings.sh [name] [place]
```

```
#!/bin/bash
echo “Mi chiamo $1”;
echo “Io sono di $2”;
```

Outline of Topics

1.3. Variables

A variable is a storage location paired with a SYMBOLIC NAME with contains some quantity of information defined as VALUE

```
A=2;
B=“Solanum lycopersicum is tomato”
```

1.3. Variables

Two types of variables:

- System variables (CAPITAL LETTER).
  E.g. HOME contains the home dir.
  E.g. OSTYPE contains the operating system info.

```
echo $HOME
```

- User variables (defined with lower letters) defined as id=value (No spaces, case sensitive) (e.g. counter=10;)

```
A=“Petunia hybrida”
echo “Variable A is $A”
```

Note: To print variables you need to use $ as prefix.

1.3. Variables

Special variables:

• Exit status: **$?** (prints if a command has been successful or it failed).

• Variables passed to the script: **$1, $2** … E.g. if a script is run as script02.sh filename1, inside the script ‘filename1’ can be passed as $1.

• Command outputs can be captured in user variables using **$()**. E.g. count_files=$(ls l wc -l)

1.3. Variables

Exercise 4.2 - Write a Bash script to count sequences and characters

1. The script should use an **argument** as input to get the **fasta_file_name** as an internal variable.

2. The script should use $() to run commands to calculate: 1- Number of sequences; 2- Number of characters that are not SeqIDs

3. The script should print an output that says:

   "The number of sequences of the file <file> is <seq_n>"

   "The total number of NT or AA in the file is <total_characters>"

Outline of Topics

1.4. Loops

A loop is a CONTROL STATEMENT for a specific iteration allowing a code being executed repetitively.

There are two important loops for Bash:

1. FOR
2. WHILE

1.4. Loops

A for-loop in a Bash script has three parts.

1. Define the interacting variable (x): **`for x in y`**

2. Define the consequence: **`do z`**

3. End the loop with **`done`**.

```
dir=$(ls $1 | grep fasta);
for file in $dir
  do
     seqcount=$(grep –c ‘>’ $file);
     echo “$file has $seqcount sequences”
  done
```

1.4. Loops

A while-loop in a Bash script has three parts.

1. Define the interacting variable (x): **`while[x]`**

2. Define the consequence: **`do z`**

3. End the loop with **`done`**.

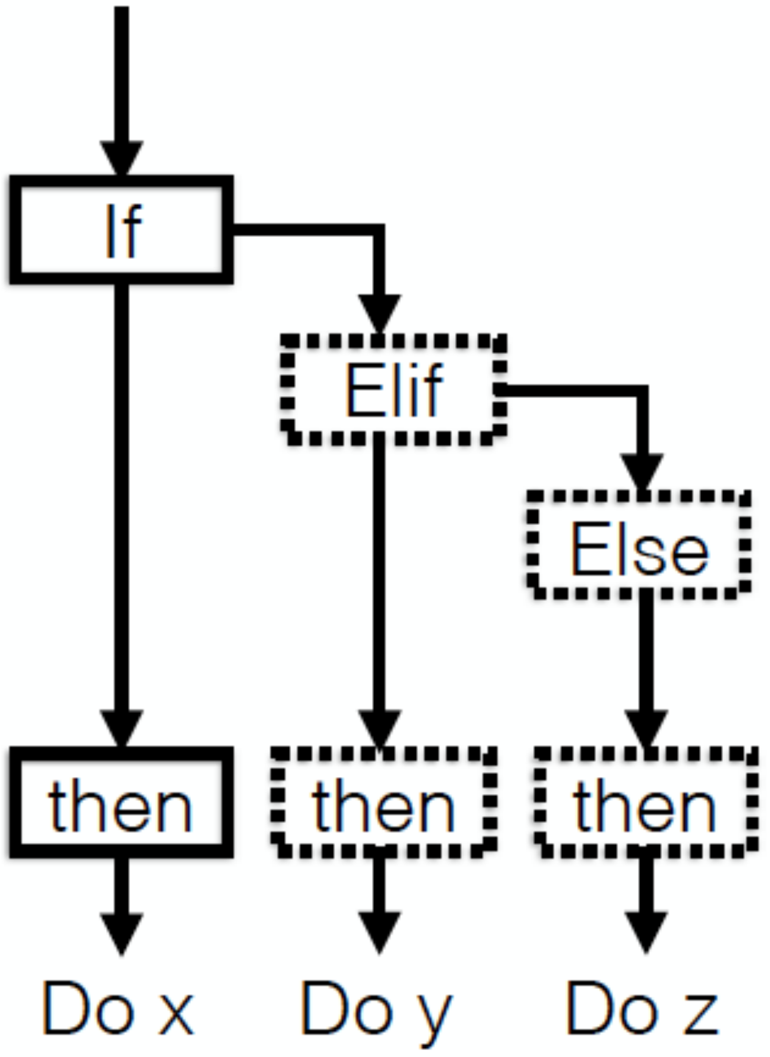
```
x=0;
While [$x -le 10]
  do
     echo “variable x value is $x”;
     x=$(($x + 1));
  done
```

Outline of Topics

1.5. Conditionals

A conditional is a CONTROL STATEMENT where a statement is tested for TRUE or FALSE. Depending of the result, a different action is performed.

CONDITIONALS	RESULT
if	First condition to meet
then	If the condition is met, do this
elif	Alternative condition to meet
else	If none is met, then do this



1.5. Conditionals

Testing a condition.

I- Square brackets: **[<space><condition?<space>]**

```
if [ $1 –gt $2 ]
   then
      echo “$1 is greater than $2”;
   else
      echo “$2 is greater than $1”;
   fi
```

II- Test command: **test <condition>**

```
if test –f $1
   then
      echo “Running the analysis on $1”;
   else
      echo “ERROR: $1 file does not exists”;
   fi
```

1.5. Conditionals

Numeric operators used in conditionals

OPERATOR	MEANING	EXAMPLE
-eq	EQUAL	`[1 -eq 1]`
-ne	NON EQUAL	`[3 -ne 1]`
-gt	GREATER THAN	`[3 -gt 1]`
-ge	GREATER EQUAL THAN	`[3 -ge 3]`
-lt	LESS THAN	`[2 -lt 5]`
-le	LESS EQUAL THAN	`[2 -le 3]`

1.5. Conditionals

File tests used in conditionals

OPERATOR	MEANING	EXAMPLE
-e	File exists	`test —e $file`
-d	File exists and it is a dir	`test —e $file`
-f	File exists and it is regular	`test —e $file`
-s	File exists with > 0 bytes	`test —e $file`
-r/-w/-x	File exists and it is r/w/x	`test —e $file`
-nt/ot	File exists and it is older/newer than	`test —e $file`

Regular expression in Bash

1. Use double square brackets: `[[ regexp ]]`

2. Use the operator `=~` for match or `!~` for doesn't match

3. Use `^` as line starting or `$` as line ending

4. Use `[0-9]` for any number or `[a-z]` for any letter

1.5. Conditionals

Regular expression in Bash

MODIFIER	DESCRIPTION
. (dot)	match any character once
*	match occurrence, zero or more OR multiple characters
+	match occurrence, one or more
?	match occurrence, zero or one
{n}	match n number of occurrences
\	escape (to match special characters)
[0-9]	any number
[a-z] or [A-Z]	any letter

http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_03

More at:

http://www.cheatography.com/davechild/cheat-sheets/regular-expressions/

1.5. Conditionals

Regular expression in Bash

```
source=$1;

if [[ $source =~ ^ftp ]];
        then
                echo "Source is a ftp url”
                echo "Downloading the file";
                countseq=$(curl $source | grep -c '>');
                echo "File has $countseq sequences";
        else
                grep -c '>' $source;
        fi
```

Outline of Topics

1.6. Debugging a Bash script

Debugging is the process of FINDING and RESOLVING defects in the PROGRAMMING CODE that prevents it of being run correctly

How to debug a script ?

http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_02_03.html

1. Read the error and think about it.

2. Print status messages in each of the steps.

3. Check variables using the if.

4. Check command results errors using $!

5. Run the script in debugging mode (bash -x or bash --debugger)

Best practices writing scripts

1. Write comments.

2. Break the conditionals and loops into different lines using tabs as indentations.

3. Use semicolons at the end of each command.

4. Define variables.

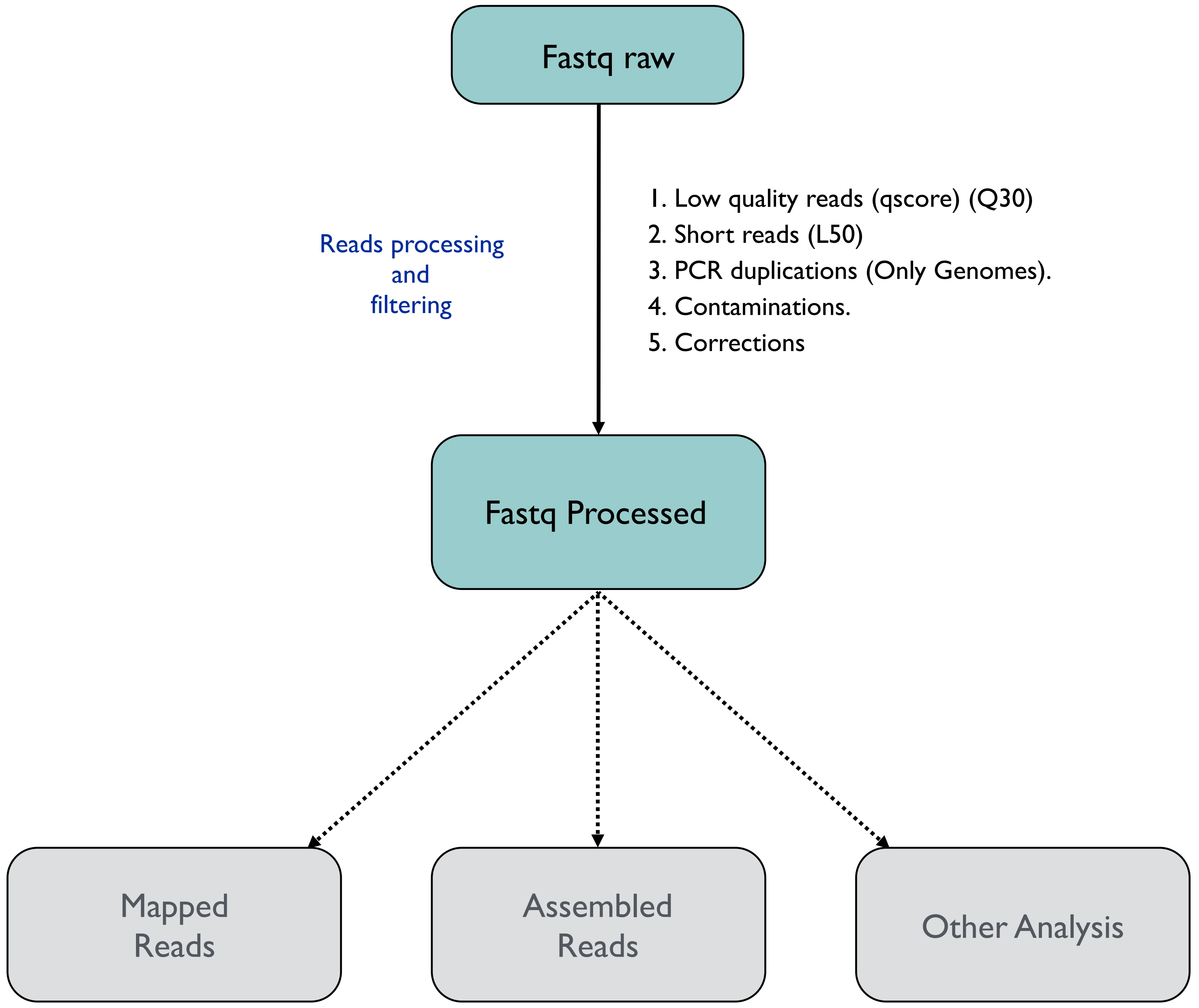
5. Use if’s to check variables.

6. Print status messages.

7. Create a small file to test the script

Outline of Topics

2. Read processing

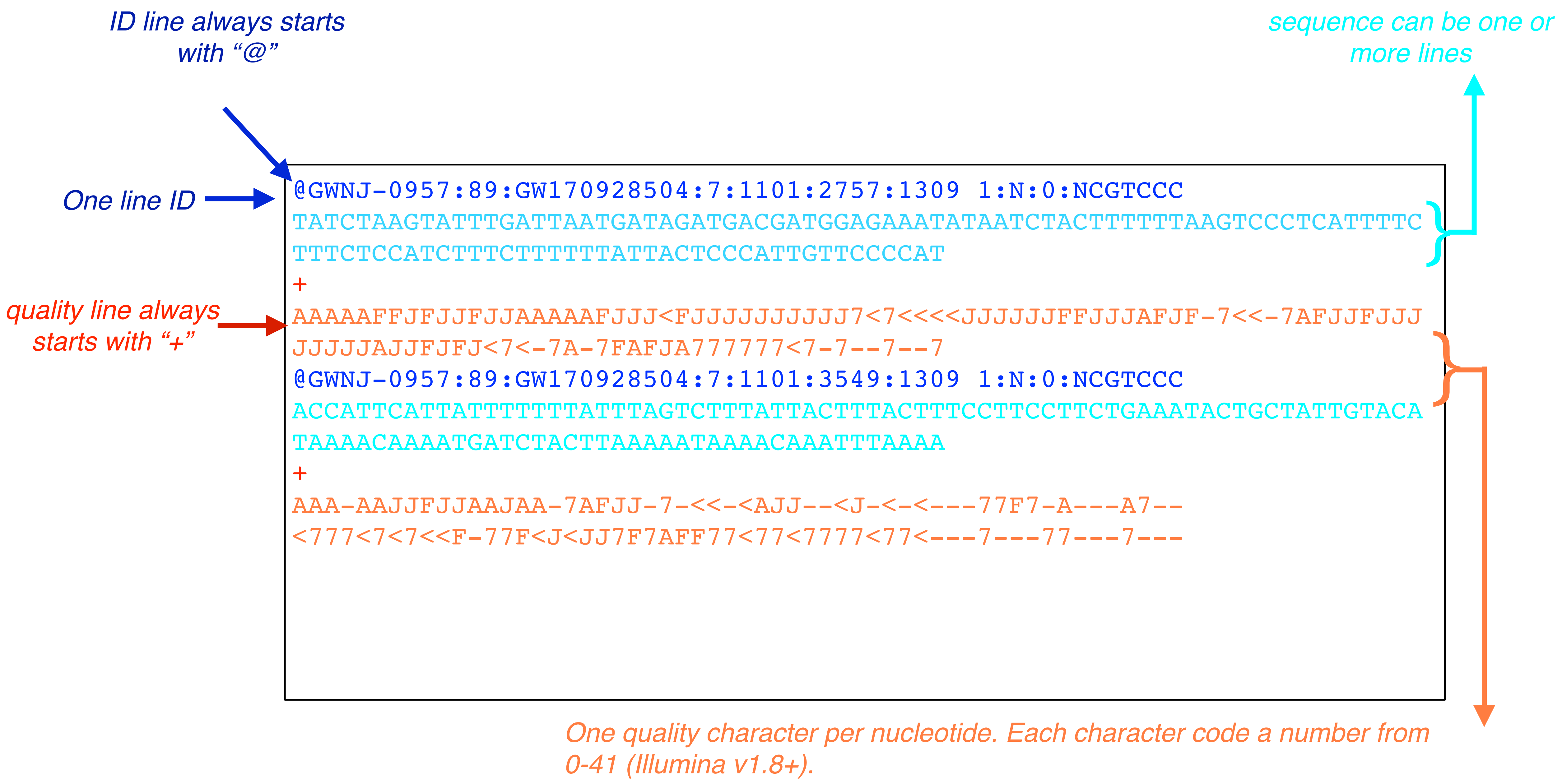


Outline of Topics

2.1. FastQ files

II. FASTQ

FASTQ format is a text based file format that store usually DNA sequences. It contains information about the sequencing QUALITY of each nucleotide.



ID line always starts with “@”

sequence can be one or more lines

One line ID

quality line always starts with “+”

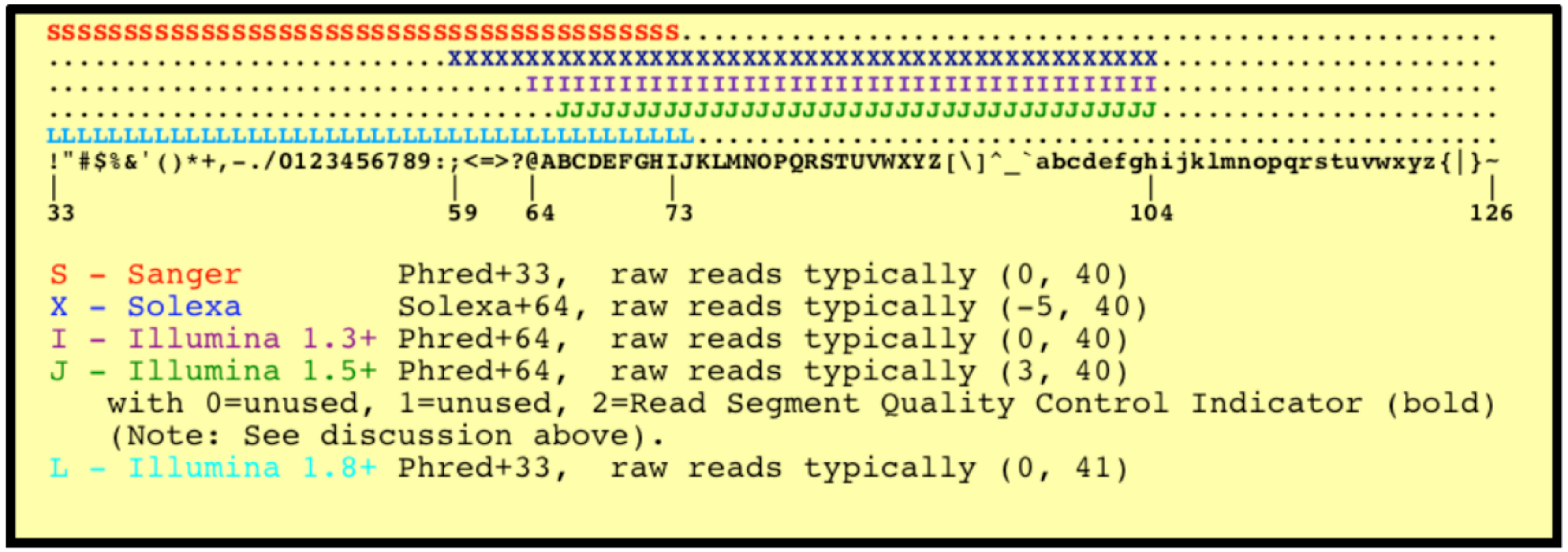
One quality character per nucleotide. Each character code a number from 0-41 (Illumina v1.8+).

```
@GWNJ-0957:89:GW170928504:7:1101:2757:1309 1:N:0:NCGTCCC
TATCTAAGTATTTGATTAATGATAGATGACGATGGAGAAATATAATCTACTTTTTAAGTCCCTCATTTTC
TTTCTCCATCTTTCTTTTTTATTACTCCCATTGTTCCCCAT
+
AAAAAFFJFJJFJJAAAAAFJJJ<FJJJJJJJJJJ7<7<<<<JJJJJJFFJJJAFJF-7<<-7AFJJFJJJ
JJJJJAJJFJFJ<7<-7A-7FAFJA777777<7-7--7--7
@GWNJ-0957:89:GW170928504:7:1101:3549:1309 1:N:0:NCGTCCC
ACCATTCATTATTTTTTTATTTAGTCTTTATTACTTTACTTTCCTTCCTTCTGAAATACTGCTATTGTACA
TAAAACAAAATGATCTACTTAAAAATAAAACAAATTTAAAA
+
AAA-AAJJFJJAAJAA-7AFJJ-7-<<-<AJJ--<J-<-<---77F7-A---A7--
<777<7<7<<F-77F<J<JJ7F7AFF77<77<7777<77<---7---77---7---
```

2.1. FastQ files

II. FASTQ

FASTQ format is a text based file format that store usually DNA sequences. It contains information about the sequencing QUALITY of each nucleotide.



```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX......................
...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
.................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ......................
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL....................................................
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
|                         |    |        |                              |                     |
33                        59   64       73                            104                   126

S - Sanger        Phred+33,  raw reads typically (0, 40)
X - Solexa        Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40)
   with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
   (Note: See discussion above).
L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

Phred score of a base is: **Qphred=-10 log10 (e)**

Q=15	e=0.03	(min. used Sanger)
Q=20	e=0.01	(min. used 454 and Illumina)
Q=30	e=0.001	(standard used 454)

2.1. FastQ files

II. FASTQ

FASTQ format is a text based file format that store usually DNA sequences. It contains information about the sequencing QUALITY of each nucleotide.

Phred quality scores Q are defined as a property which is logarithmically related to the base-calling error probabilities P.[2]

$$Q = -10 \log_{10} P$$

or

$$P = 10^{\frac{-Q}{10}}$$

For example, if Phred assigns a quality score of 30 to a base, the chances that this base is called incorrectly are 1 in 1000.

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

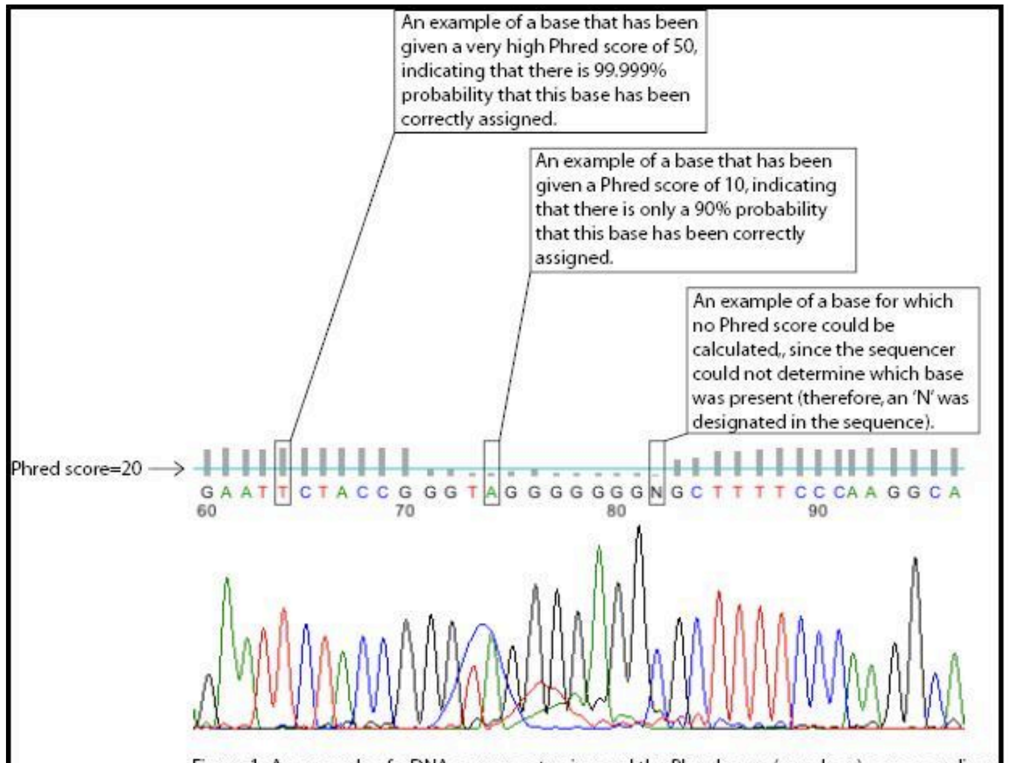


Figure 1. An example of a DNA sequence tracing and the Phred score (grey bars) corresponding to each colored peak. The colored peaks on the trace correspond to each DNA letter. For example 'T' bases are represented in red, and this sequence has four 'T' bases on a row, as viewed by the four red peaks in the sequence. The aqua horizontal line placed across the grey bars represents a Phred score of 20 which is considered an acceptable level of accuracy. As indicated in Table 1, a Phred score of 20 corresponds to a 99% accuracy in the base call. Therefore, bars above this line indicate base calls that have a higher than 99% probability of being correct. Those below have less than a 99% probability of being correct. Sequence tracing program is courtesy of FinchTV (www.geospiza.com).

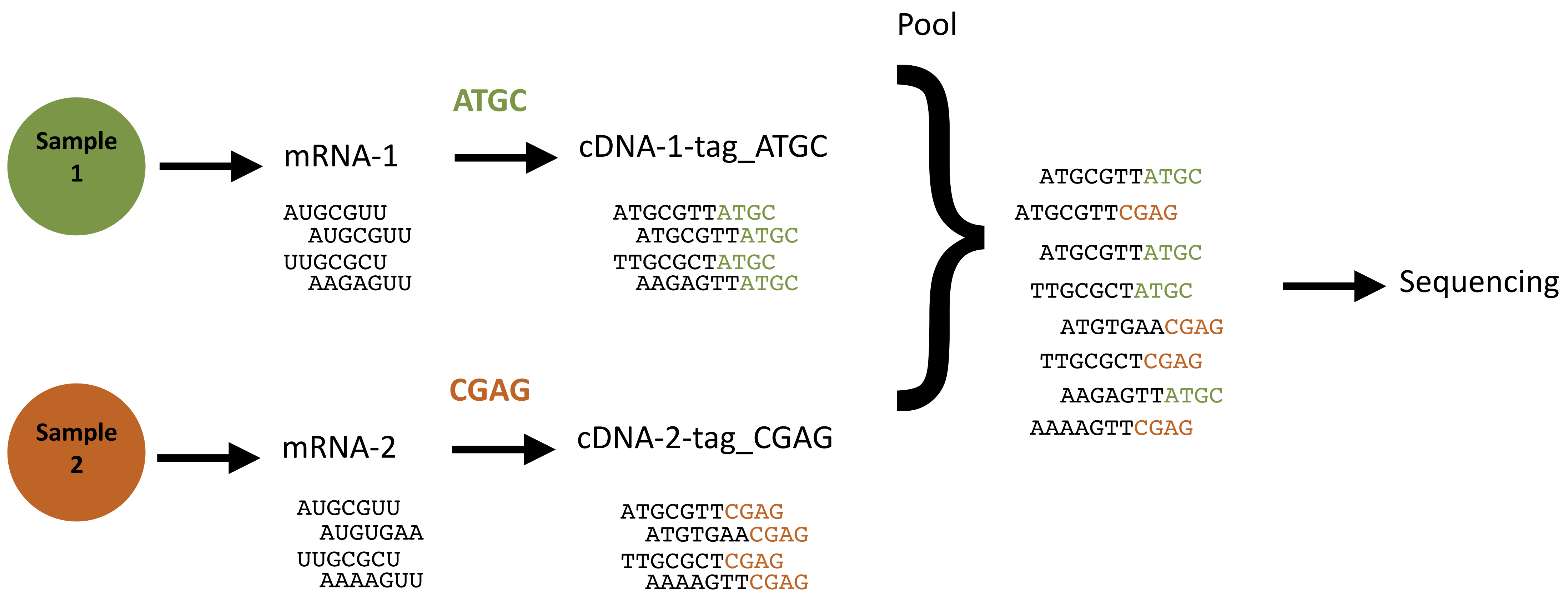
https://en.wikipedia.org/wiki/Phred_quality_score

Outline of Topics

2.2. Demultiplexing

Multiplexing

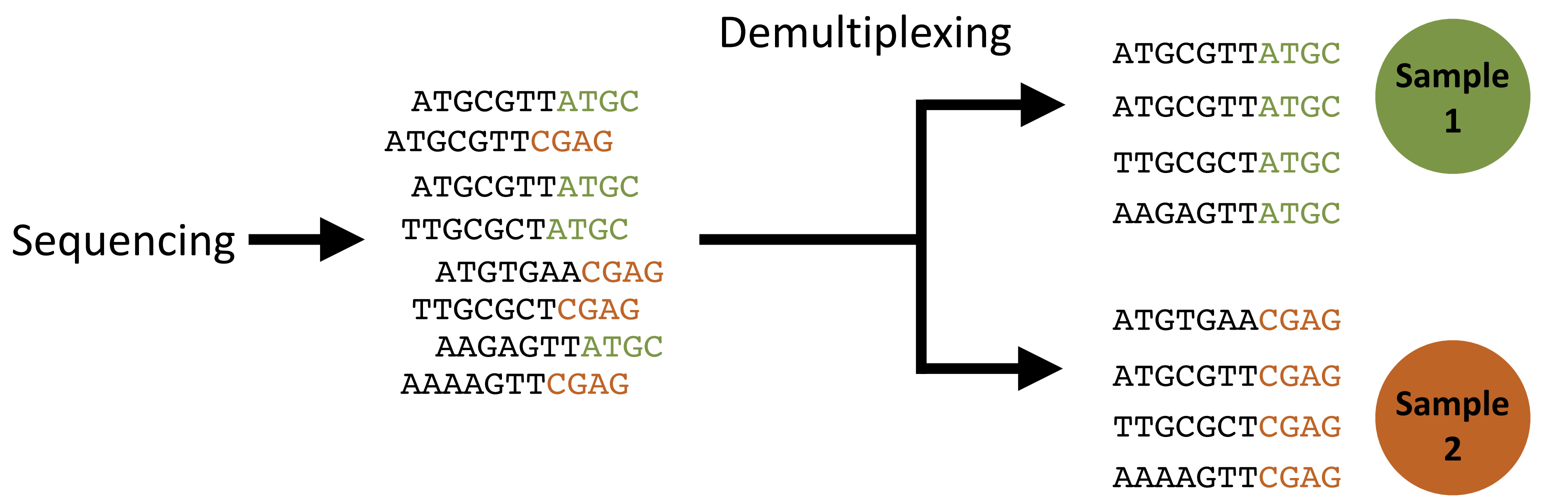
Use of DNA tags (4-7 bp) to identify samples in the same sequencing lane, cell or sector.



2.2. Demultiplexing

De-Multiplexing

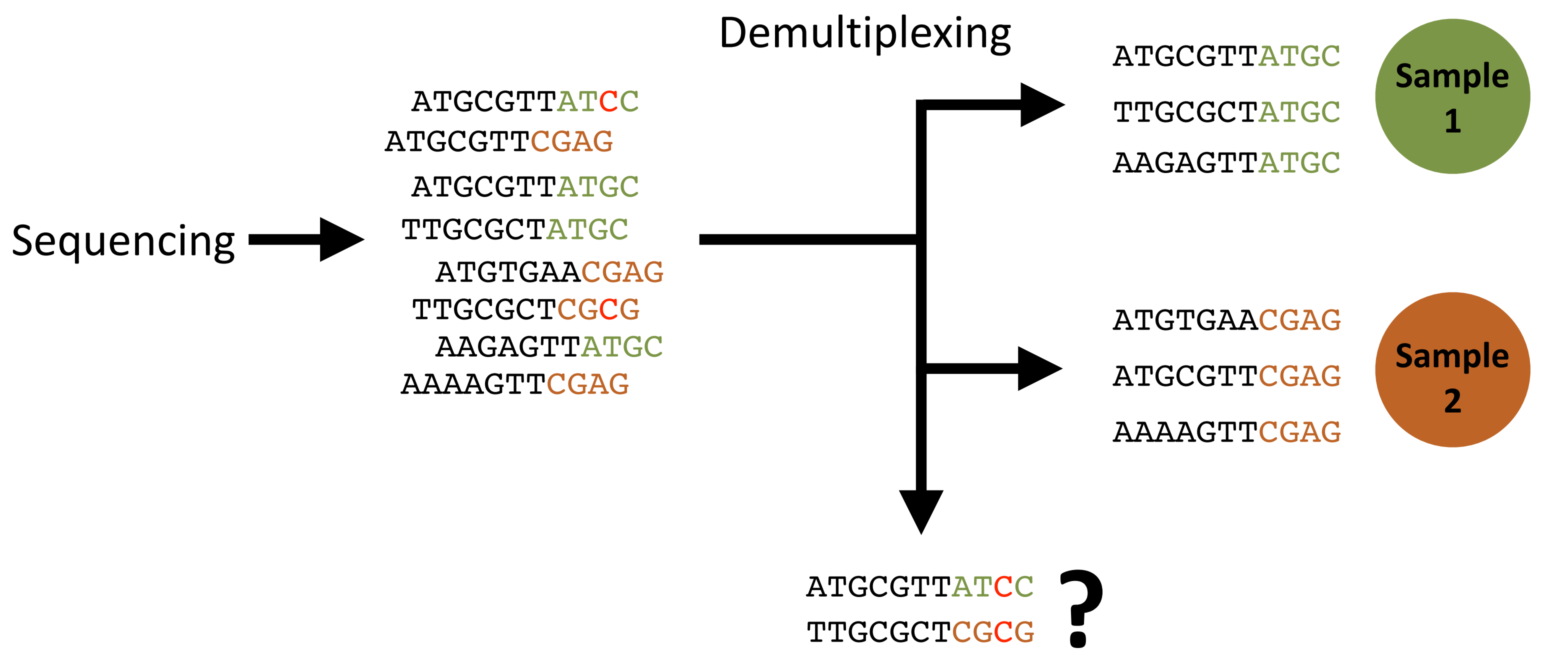
Identification of the sequenced DNA samples using the DNA tag



2.2. Demultiplexing

De-Multiplexing

Identification of the sequenced DNA samples using the DNA tag



2.2. Demultiplexing

De-Multiplexing

Keys for barcode/tag designing (GBS/RADseq):

- The barcode does not contain or recreate the **enzyme cut site**.
- *Any barcode in a set is at least **two substitutions away from any other barcode**.*
- They **vary in length** as a set (to avoid the all cut site bases appearing at the same positions in the sequencing read).
- They contain the **complementary sticky** end to the enzyme cut site.

http://www.maizegenetics.net/genotyping-by-sequencing-gbs

2.2. Demultiplexing

De-Multiplexing

Identification of the sequenced DNA samples using the DNA tag

Software	RE	Link
Fastx-toolkit (Barcode splitter)	No	http://hannonlab.cshl.edu/fastx_toolkit/
Ea-utils (Fastq-multx)	No	https://expressionanalysis.github.io/ea-utils/
GBSX	Yes	https://github.com/GenomicsCoreLeuven/GBSX
TASSEL	Yes	http://www.maizegenetics.net/tassel

Outline of Topics

2.3. Read Quality Evaluation

- Does the sequencing produced the expected number of reads?

  READ COUNTS

- Do the reads have the expected average length?

  AVERAGE READ LENGTH

- Do the reads have the expected nucleotide qscore?

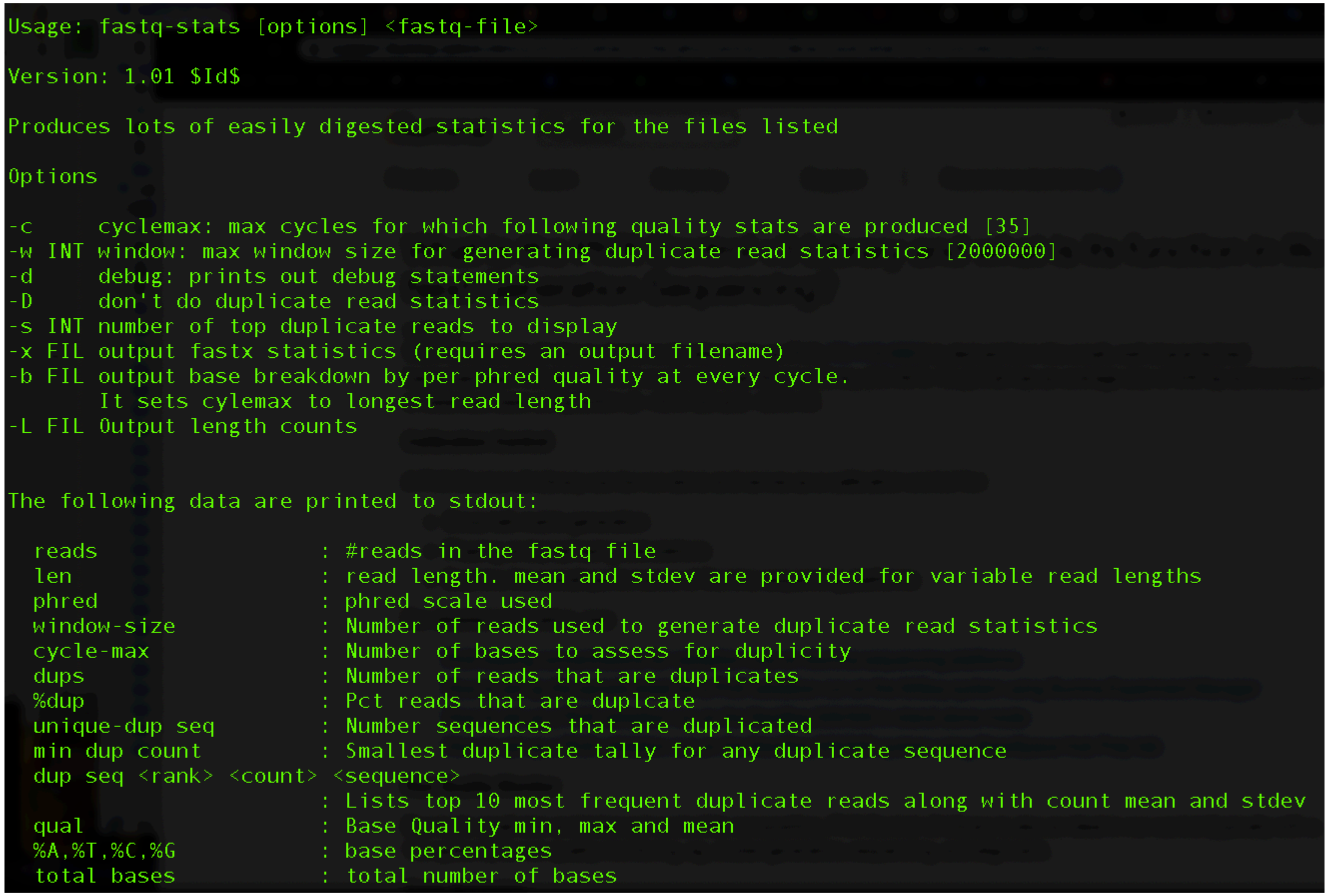
  QSCORE NUCLEOTIDE BOXES

$$\downarrow$$

$\boxed{\textbf{Technology dependent}}$

2.3. Read Quality Evaluation

Fastq-Stats (http://expressionanalysis.github.io/ea-utils/)

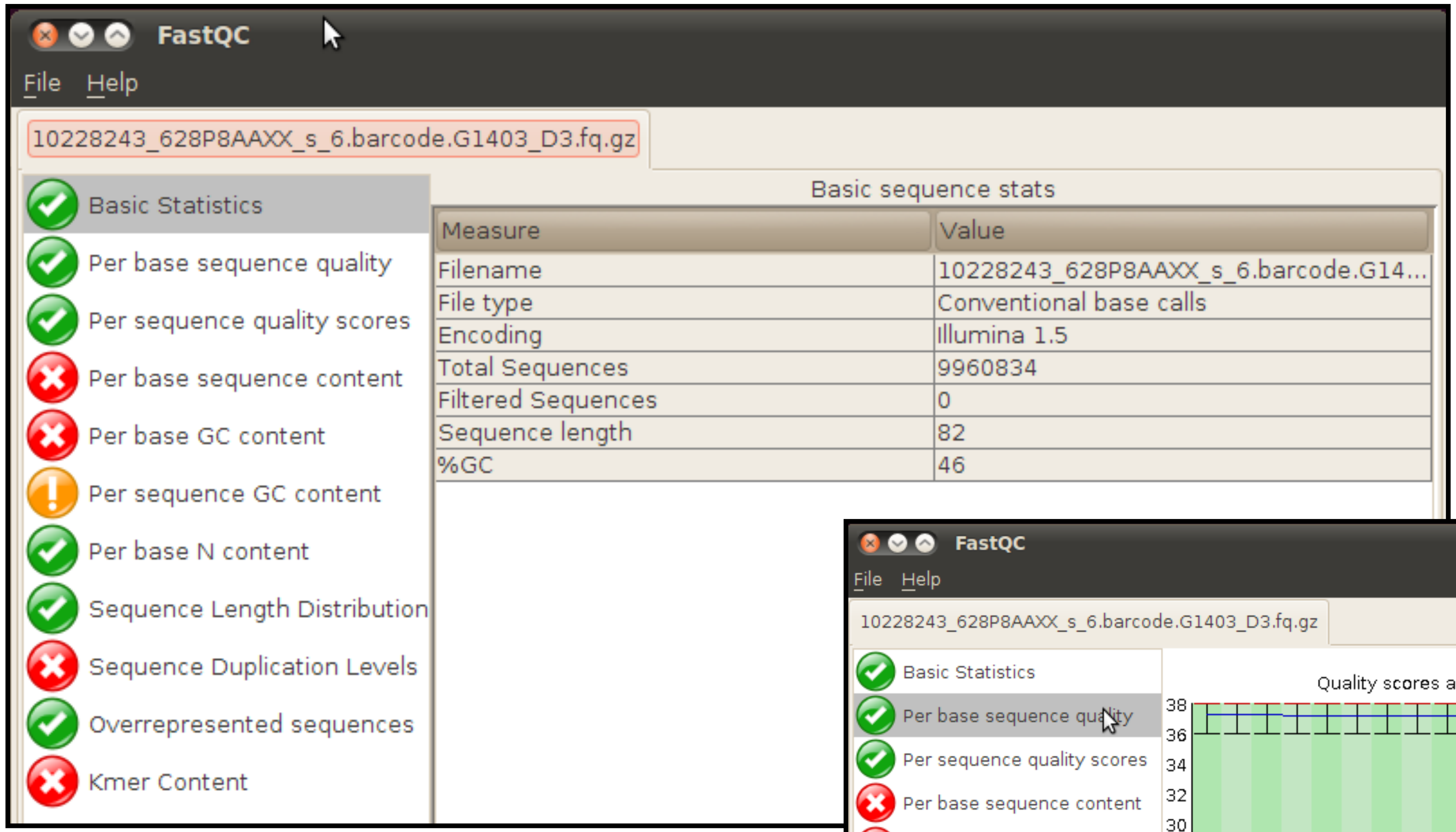


```
Usage: fastq-stats [options] <fastq-file>

Version: 1.01 $Id$

Produces lots of easily digested statistics for the files listed

Options

-c     cyclemax: max cycles for which following quality stats are produced [35]
-w INT window: max window size for generating duplicate read statistics [2000000]
-d     debug: prints out debug statements
-D     don't do duplicate read statistics
-s INT number of top duplicate reads to display
-x FIL output fastx statistics (requires an output filename)
-b FIL output base breakdown by per phred quality at every cycle.
       It sets cylemax to longest read length
-L FIL Output length counts


The following data are printed to stdout:

  reads                    : #reads in the fastq file
  len                      : read length. mean and stdev are provided for variable read lengths
  phred                    : phred scale used
  window-size              : Number of reads used to generate duplicate read statistics
  cycle-max                : Number of bases to assess for duplicity
  dups                     : Number of reads that are duplicates
  %dup                     : Pct reads that are duplcate
  unique-dup seq           : Number sequences that are duplicated
  min dup count            : Smallest duplicate tally for any duplicate sequence
  dup seq <rank> <count> <sequence>
                           : Lists top 10 most frequent duplicate reads along with count mean and stdev
  qual                     : Base Quality min, max and mean
  %A,%T,%C,%G              : base percentages
  total bases              : total number of bases
```

```
$ fastq-stats <my_fastq_file> > my_fastq_stats
```

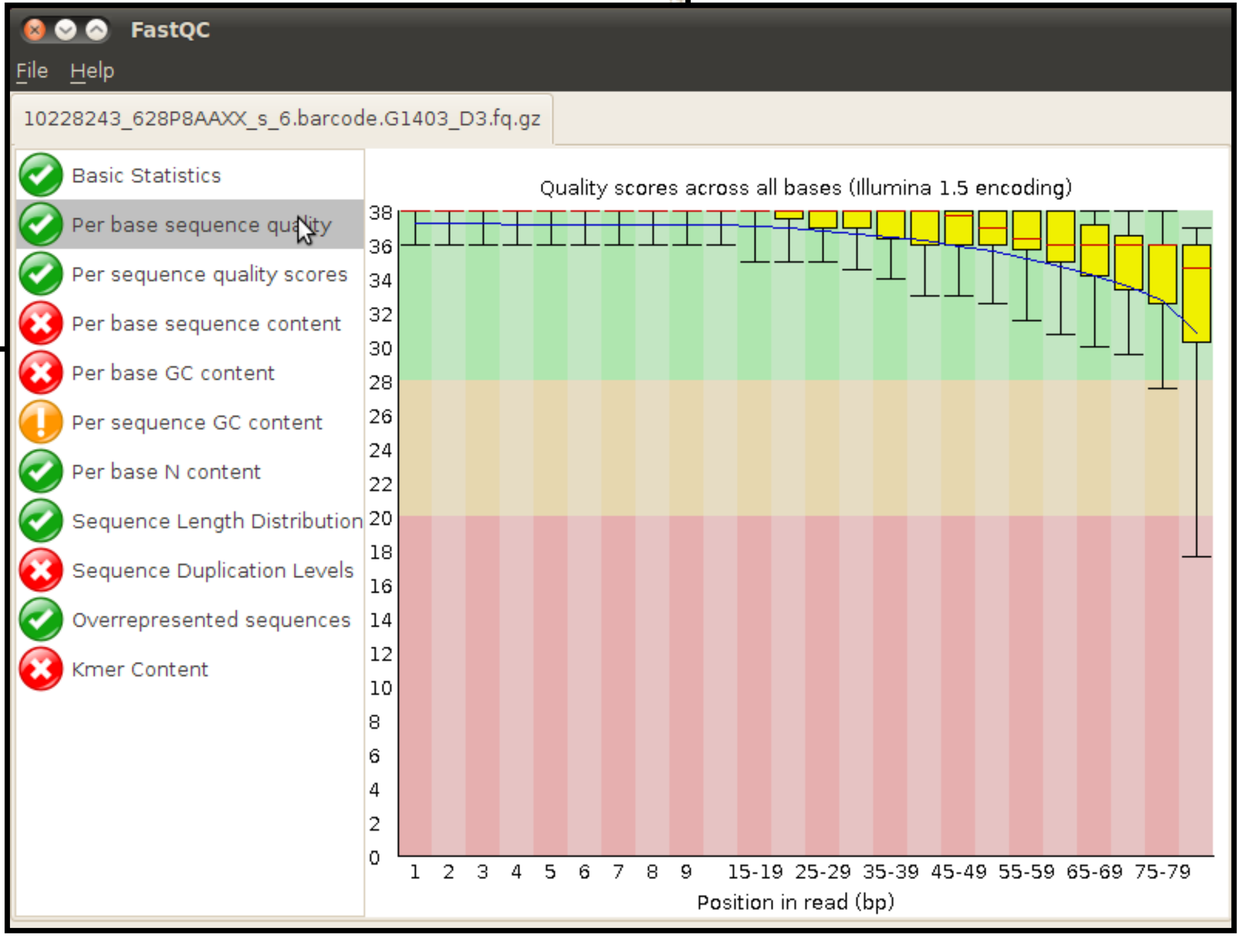
2.3. Read Quality Evaluation

FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)


FastQC
File Help
10228243_628P8AAXX_s_6.barcode.G1403_D3.fq.gz
✓ Basic Statistics
✓ Per base sequence quality
✓ Per sequence quality scores
✗ Per base sequence content
✗ Per base GC content
! Per sequence GC content
✓ Per base N content
✓ Sequence Length Distribution
✗ Sequence Duplication Levels
✓ Overrepresented sequences
✗ Kmer Content

Basic sequence stats

Measure	Value
Filename	10228243_628P8AAXX_s_6.barcode.G14...
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	9960834
Filtered Sequences	0
Sequence length	82
%GC	46


FastQC
File Help
10228243_628P8AAXX_s_6.barcode.G1403_D3.fq.gz
✓ Basic Statistics
✓ Per base sequence quality
✓ Per sequence quality scores
✗ Per base sequence content
✗ Per base GC content
! Per sequence GC content
✓ Per base N content
✓ Sequence Length Distribution
✗ Sequence Duplication Levels
✓ Overrepresented sequences
✗ Kmer Content
Quality scores across all bases (Illumina 1.5 encoding)
38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
1 2 3 4 5 6 7 8 9 15-19 25-29 35-39 45-49 55-59 65-69 75-79
Position in read (bp)

2.3. Read Quality Evaluation

NanoPlot (https://github.com/wdecoster/NanoPlot)

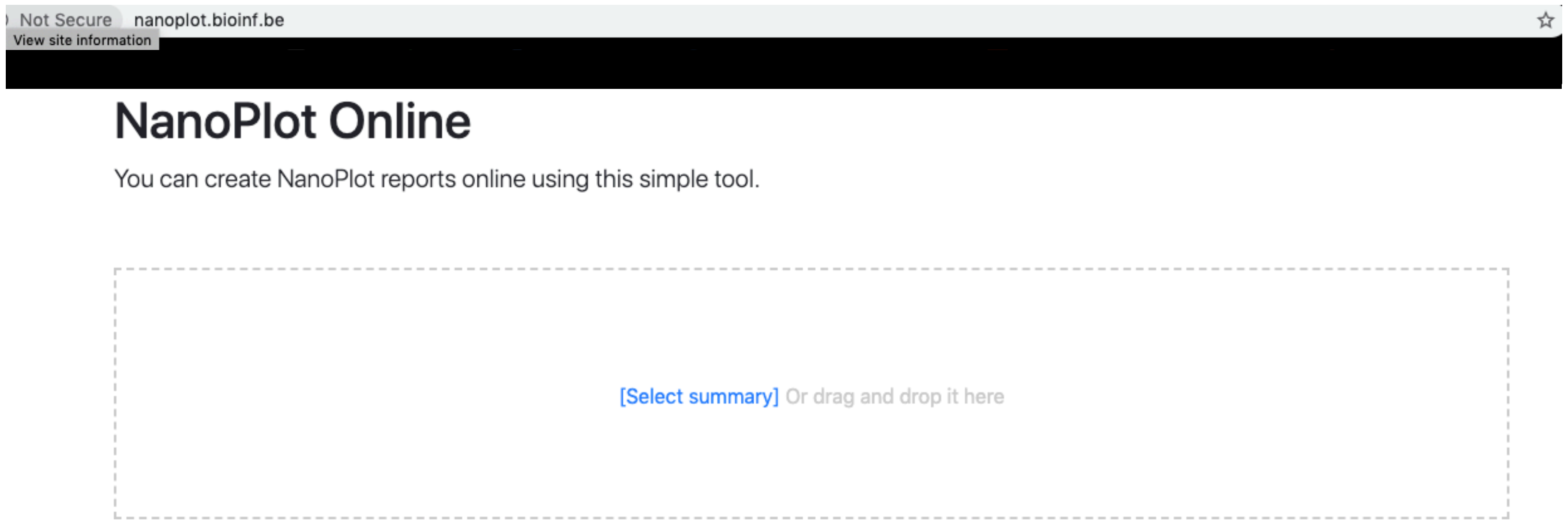
Oxford Nanopore sequencing data

USAGE

Command Line Interface

```
NanoPlot [-h] [-v] [-t THREADS] [--verbose] [--store] [--raw]
                [-o OUTDIR] [-p PREFIX] [--maxlength N] [--minlength N]
                [--drop_outliers] [--downsample N] [--loglength]
                [--percentqual] [--alength] [--minqual N]
                [--readtype {1D,2D,1D2}] [--barcoded] [--runtime_until N]
                [-c COLOR]
                [-f {eps,jpeg,jpg,pdf,pgf,png,ps,raw,rgba,svg,svgz,tif,tiff}]
                [--plots [{kde,hex,dot,pauvre} [{kde,hex,dot,pauvre} ...]]]
                [--listcolors] [--no-N50] [--N50] [--title TITLE]
                (--fastq file [file ...] | --fasta file [file ...] | --fastq_rich file [file ...] | --fastq_minimal
```

Online Tool: http://nanoplot.bioinf.be/


Not Secure nanoplot.bioinf.be
View site information

NanoPlot Online

You can create NanoPlot reports online using this simple tool.

[Select summary] Or drag and drop it here

Outline of Topics

2.4. Quality Filtering

- **Trimming and removing of low quality reads (Short Reads)**
- **Adapter removal (Short & Sometimes Long Reads)**
- **Filtering of reads below certain length (Short & Long Reads)**
- **Remove of the PCR duplications (Short Reads - Assemblies)**
- **Remove of contaminations (Short and Long Reads)**
- **Sequence correction (Short & Long Reads - Assemblies)**

2.4. Quality Filtering

- Most common **filtering** is associated with a **minimum length** and a **minimum qscore** (extremes, by average, minimum for all the nucleotides)

Tecnology	min. length (bp)	min. qscore
454	100	20
Illumina	50	30
SOLiD	20	30
Ion Torrent	50	20
PacBio	1000	NA
Oxford Nanopore	1000	NA

2.4. Quality Filtering

- Most common filtering is associated with a **minimum length** and a **minimum qscore** (extremes, by average, minimum for all the nucleotides)
- Some of these tools also **remove the adapters**
- Tools and time for processing varies depending of the technology.

Software	Link
Fastx-toolkit	http://hannonlab.cshl.edu/fastx_toolkit/
Ea-utils	https://expressionanalysis.github.io/ea-utils/
PrinSeq	http://prinseq.sourceforge.net/
Trimmomatic	http://www.usadellab.org/cms/?page=trimmomatic

E.g. running Ea-utils command fastq-mcf:

```
$ fastq-mcf -q 30 -l 50 -o s01_Q30L50_R1.fq Illumina_Adapters.fa s01_R1.fq
```

2.4. Quality Filtering

- **Contaminations** can be removed **mapping the reads against a reference** with the contaminants such as E. coli and human genomes. The most common tools

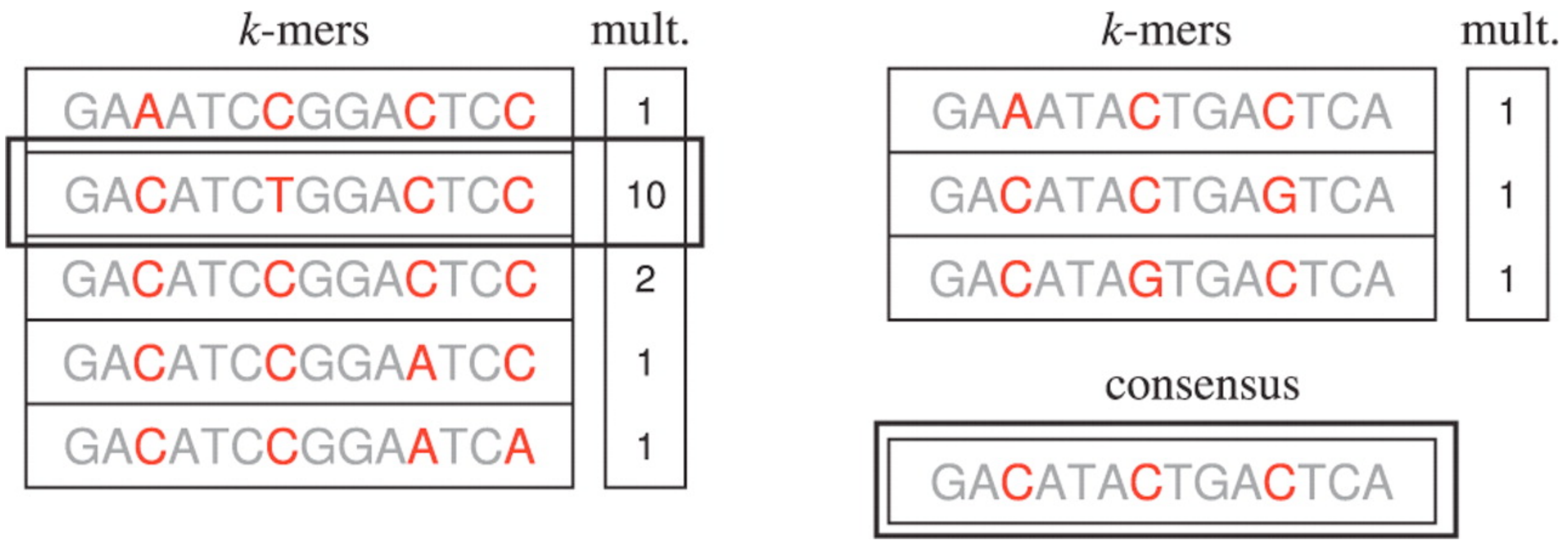
    - ✴ Bowtie or BWA (for short reads)

    - ✴ BlasR or Minimap2 (for long reads).

- It is specially important to remove the **contaminations on assembly processes**. They can removed before or after the assembly.

2.4. Quality Filtering

- **Read correction** is the process for which **single nucleotide errors** are corrected comparing them with Kmers, assembled sequences or other reads.
- Read correction can improve the sequence assembly.



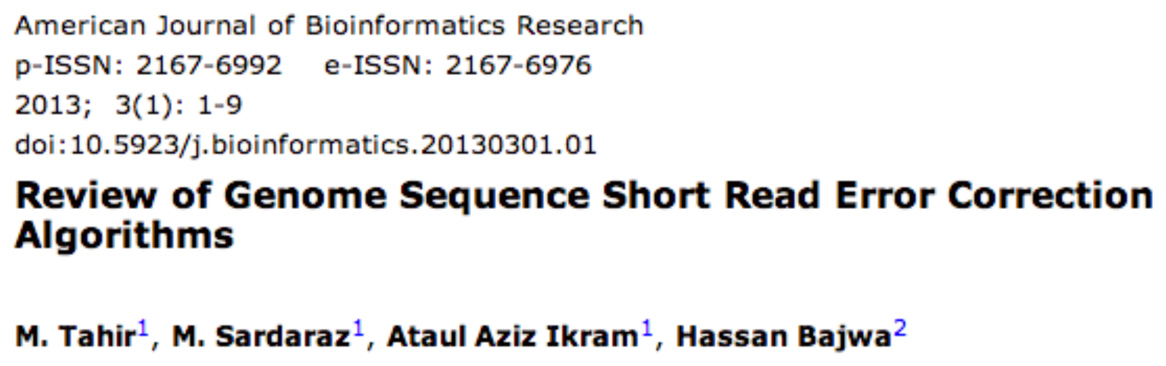
Medvedev P. et al. Error correction of high-throughput sequencing datasets with non-uniform coverage Bioinformatics. 2011 27 (13):i137-i141

2.4. Quality Filtering

- **Read correction** is the process for which **single nucleotide errors** are corrected comparing them with Kmers, assembled sequences or other reads.
- Read correction can improve the sequence assembly.

Popular tools for short reads

- Quake (http://www.cbcb.umd.edu/software/quake/index.html)
- Reptile (http://aluru-sun.ece.iastate.edu/doku.php?id=software)
- ECHO (http://uc-echo.sourceforge.net/)
- Corrector (http://soap.genomics.org.cn/soapdenovo.html)
- Musket (http://musket.sourceforge.net/)



American Journal of Bioinformatics Research
p-ISSN: 2167-6992 e-ISSN: 2167-6976
2013; 3(1): 1-9
doi:10.5923/j.bioinformatics.20130301.01

Review of Genome Sequence Short Read Error Correction Algorithms

M. Tahir[1], **M. Sardaraz**[1], **Ataul Aziz Ikram**[1], **Hassan Bajwa**[2]

2.4. Quality Filtering

- **Read correction** is the process for which **single nucleotide errors** are corrected comparing them with Kmers, assembled sequences or other reads.
- Read correction can improve the sequence assembly.

Popular tools for long reads

- Self-alignment of PacBio reads.

- PBcR (Celera Assembler: http://wgs-assembler.sourceforge.net/)
- Canu (https://github.com/marbl/canu)

- Alignment of PacBio reads with Illumina contigs.

- PacBioToCA (Celera Assembler: http://wgs-assembler.sourceforge.net/)

- Alignment of PacBio reads with Illumina reads.

- Lordec (http://www.atgc-montpellier.fr/lordec/)
- Proovread (https://github.com/BioInf-Wuerzburg/proovread)
- LSC (http://www.healthcare.uiowa.edu/labs/au/LSC/)

Exercise 4.3 - Write a Bash script to retrieve the stats information of Fastq files

MATERIAL SOURCE:

/data/99_genomicclass/00_shared_data/Genomics/01_Illumina

EXPECTED OUTPUT:

A single file per FastQ file with the stats produced by fastq-stats