

Short Reads Quality Control and Preprocessing - Tutorial

1.- Open a Terminal Window

The programs used in this tutorial are called from the command line. In order to do that the first step is to open a Terminal window. To do this go to

Applications → Accessories → Terminal



A new window will open with a prompt ready for input. Now change to the directory with the sequence data. Type on the Terminal:

cd mda11_QC_data

2.- Open FastQC program

To start the FastQC program, you have to type on the Terminal window:

fastqc

The FastQC application will start and it will open a new window.

3.- Load a file into FastQC

From the FastQC window go to:

File → Open

And load from the folder **mda11_QC_data** the file called












bacteria.fastq

4.- Look at the different FastQC result sections

Here is a copy of the results:



Summary

-  [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](#)

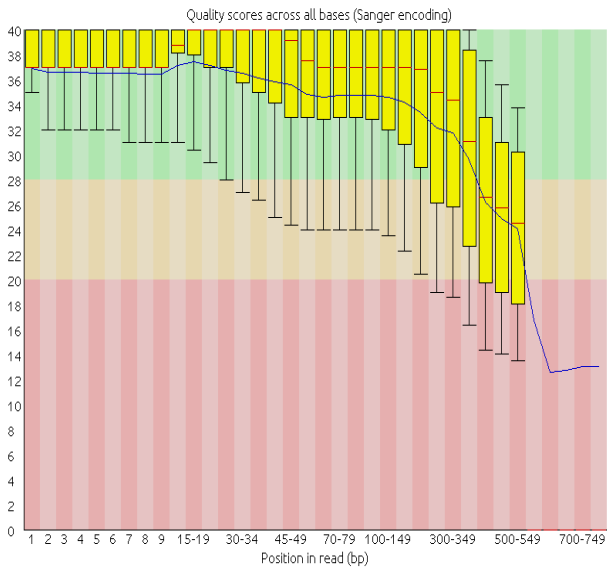


Basic Statistics

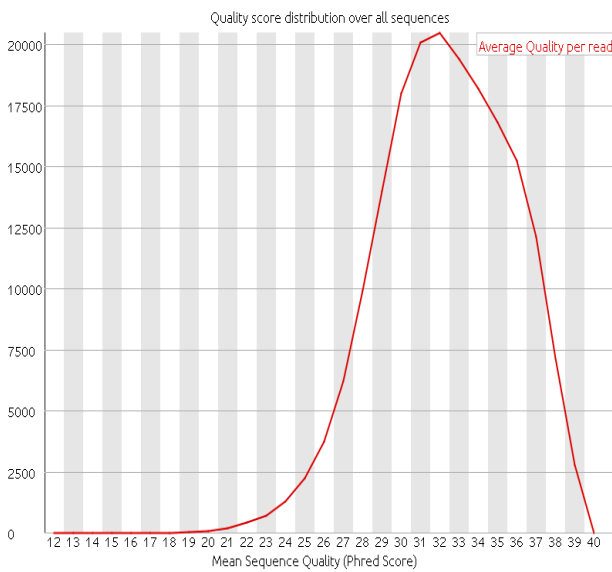
Measure	Value
Filename	bacteria.fastq
File type	Conventional base calls
Total Sequences	189472
Sequence length	40-763
%GC	54



Per base sequence quality

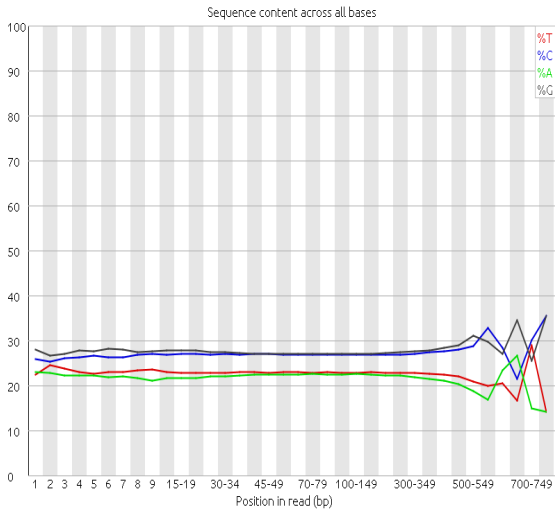


Per sequence quality scores

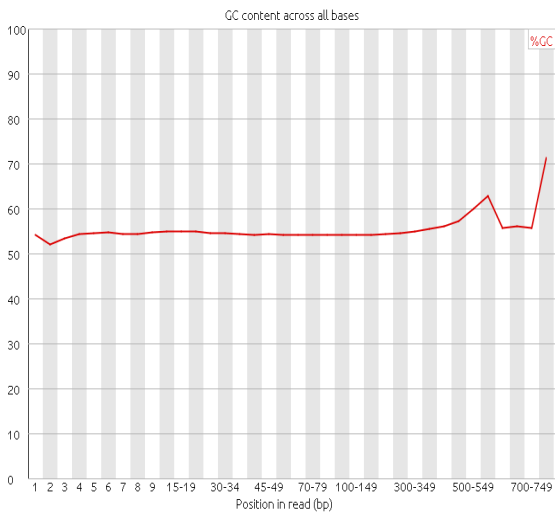




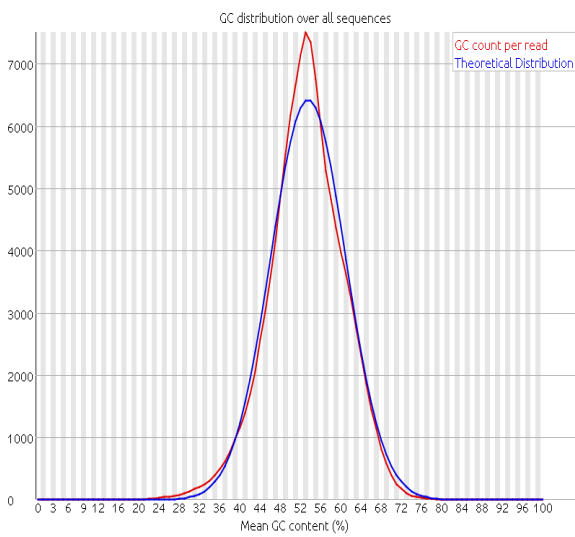
base sequence content



Per base GC content

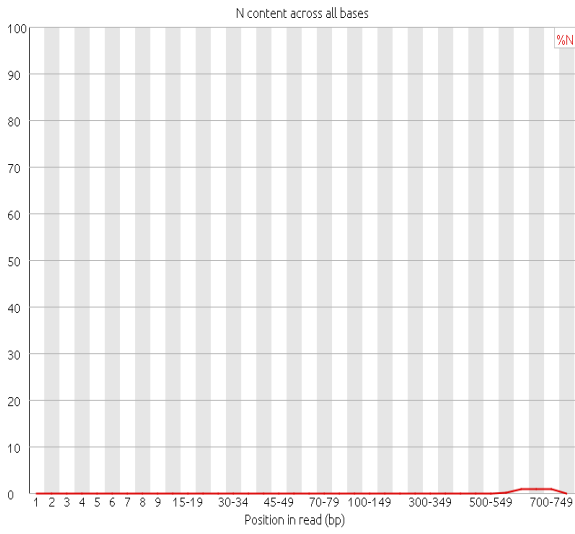


Per sequence GC content

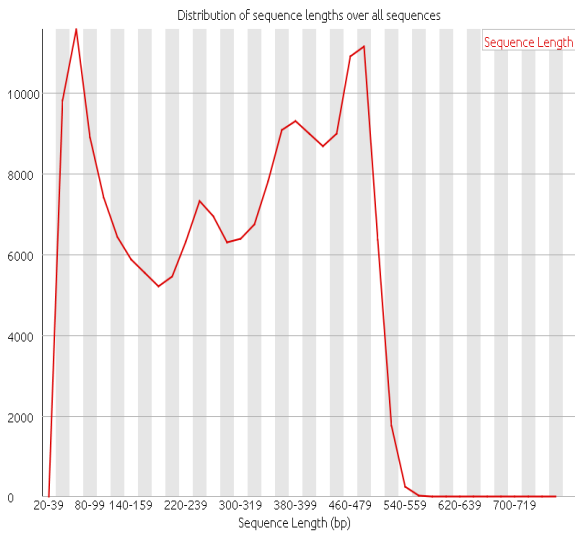




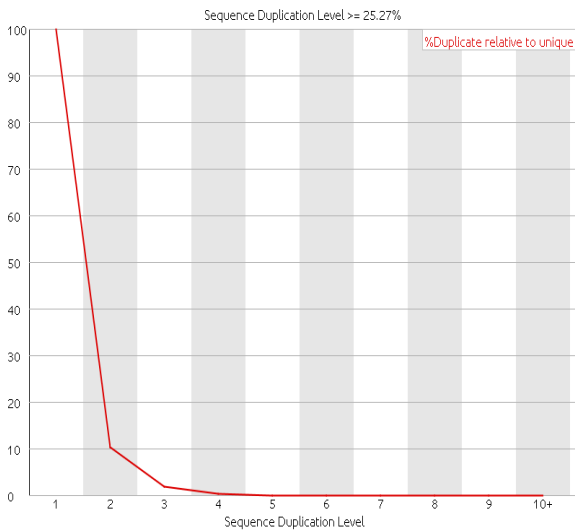
Per base N content



Sequence Length Distribution



Sequence Duplication Levels



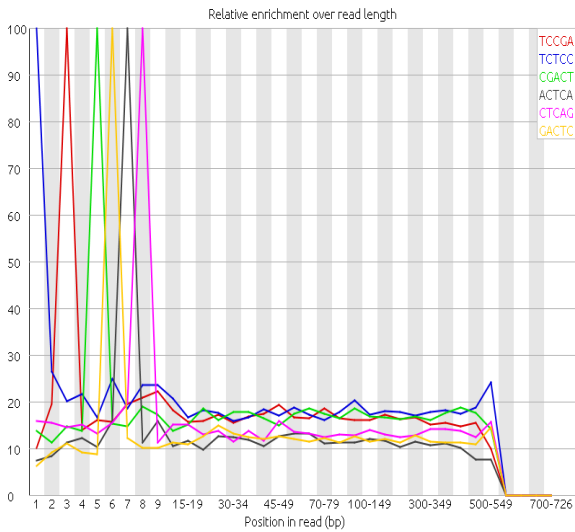


Overrepresented sequences

No overrepresented sequences



Kmer Content



Sequence	Count	Obs/Exp Overall	Obs/Exp Max	Max Obs/Exp	Position
TCCGA	55865	0.99584657	5.905306	3	
TCTCC	53810	0.94410264	5.163556	1	
CGACT	51120	0.91126245	5.27155	5	
ACTCA	35355	0.779002	6.610214	7	
CTCAG	42805	0.7630397	5.499646	8	
GACTC	35435	0.63166255	5.1448307	6	

5- Questions

How does per base sequence quality looks like?

What is the quality encoding?

Is anything wrong about the sequence length distribution?

What is the level of sequence duplication?

Is there any over-represented sequences?

6- Process reads using Fastx-toolkit

Sequences can be filtered by quality, length and they can also be trimmed using the fastx-toolkit in order to improve quality. All of these can be done with the Fastx-toolkit. To obtain arguments for each tool type the name of the tool followed by **-h**. See Fastx-toolkit command line help at the end of this tutorial

7- Process reads using Fastx-toolkit: Filter sequences smaller than 50 nt

To filter sequences smaller than 50 nt the tool fastx_clipper can be used. By default reads with more than five Ns are also filtered. Copy and paste the following line in the Terminal window.

```
fastx_clipper -i bacteria.fastq -o bacteria_m50.fastq -l 50 -Q 33 -v
```

8- Process reads using Fastx-toolkit: Trim 10 first nt and set the maximum size to 500

We are going to use as input the output of the filtering in the previous step. To trim the first 10 nt and set a maximum size of 500 we can use the tool `fastx_trimmer`. Because now we can end up with sequences smaller than 50 nt we need to filter them by setting the flag `-m`. Copy and paste the following line in the terminal window.

```
fastx_trimmer -i bacteria_m50.fastq -o bacteria_m50_trim10-500.fastq -f 11 -l 500 \
-m 50 -Q 33 -v
```

9- Process reads using Fastx-toolkit: Filter reads with a quality less than 20 over the 80% of nt

We are going to remove those reads that have less than 80% of bases with quality lower than a value of 20. We can do this using the `fastq_quality_filter`. Copy and paste the following command line:

```
fastq_quality_filter -i bacteria_m50_trim10-500.fastq \
-o bacteria_m50_trim10-500_Qual20.fastq -q 20 -p 80 -Q 33 -v
```

10- Run FastQC and analyse the last output file.

Run FastQC using the file `bacteria_m50_trim10-500_Qual20.fastq` and compare the result with the result of the same datasets without any filtering (exercise number 4)

11- Analyse exoma.fastq and mirnas.fastq files and try to improve their sequence quality.

First use FastQC to visualize the data and then use different tools from the Fastx-toolkit to filter low quality reads and to trim sequences. Look and see if you can remove regions and reads of low quality.

12- Analyse the solid.fastq file with FastQC

Compare the per base qualities with the other datasets. What is different?

FastQC Help

Basic Statistics

Summary

The Basic Statistics module generates some simple composition statistics for the file analysed.

- **Filename:** The original filename of the file which was analysed
- **File type:** Says whether the file appeared to contain actual base calls or colorspace data which had to be converted to base calls
- **Total Sequences:** A count of the total number of sequences processed. There are two values reported, actual and estimated. At the moment these will always be the same. In the future it may be possible to analyse just a subset of sequences and estimate the total number, to speed up the analysis, but since we have found that problematic sequences are not evenly distributed through a file we have disabled this for now.
- **Sequence Length:** Provides the length of the shortest and longest sequence in the set. If all sequences are the same length only one value is reported.
- **%GC:** The overall %GC of all bases in all sequences

Warning

Basic Statistics never raises a warning.

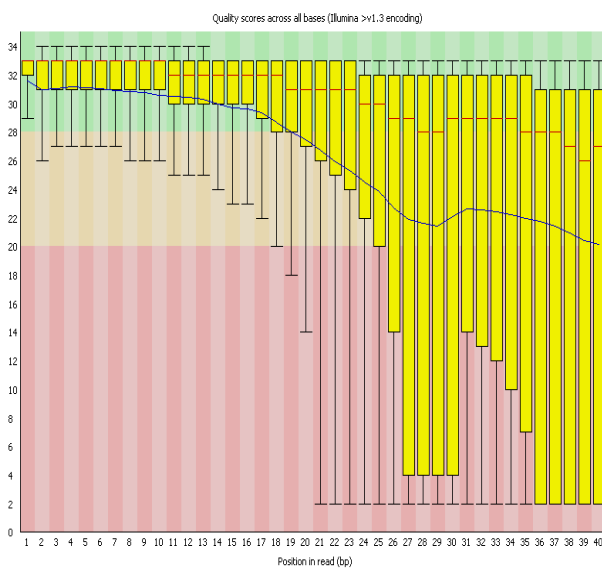
Failure

Basic Statistics never raises an error.

Per Base Sequence Quality

Summary

This view shows an overview of the range of quality values across all bases at each position in the FastQ file.



For each position a BoxWhisker type plot is drawn. The elements of the plot are as follows:

- The central red line is the median value
- The yellow box represents the inter-quartile range (25-75%)
- The upper and lower whiskers represent the 10% and 90% points
- The blue line represents the mean quality

The y-axis on the graph shows the quality scores. The higher the score the better the base call. The background of the graph divides the y axis into very good quality calls (green), calls of reasonable quality (orange), and calls of poor quality (red). The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the orange area towards the end of a read.

It should be mentioned that there are number of different ways to encode a quality score in a FastQ file. FastQC attempts to automatically determine which encoding method was used, but in some very limited datasets it is possible that it will guess this incorrectly (ironically only when your data is universally very good!). The title of the graph will describe the encoding FastQC thinks your file used.

Warning

A warning will be issued if the lower quartile for any base is less than 10, or if the median for any base is less than 25.

Failure

This module will raise a failure if the lower quartile for any base is less than 5 or if the median for any base is less than 20.

Per Sequence Quality Scores

Summary

The per sequence quality score report allows you to see if a subset of your sequences have universally low quality values. It is often the case that a subset of sequences will have universally poor quality, often because they are poorly imaged (on the edge of the field of view etc), however these should represent only a small percentage of the total sequences.

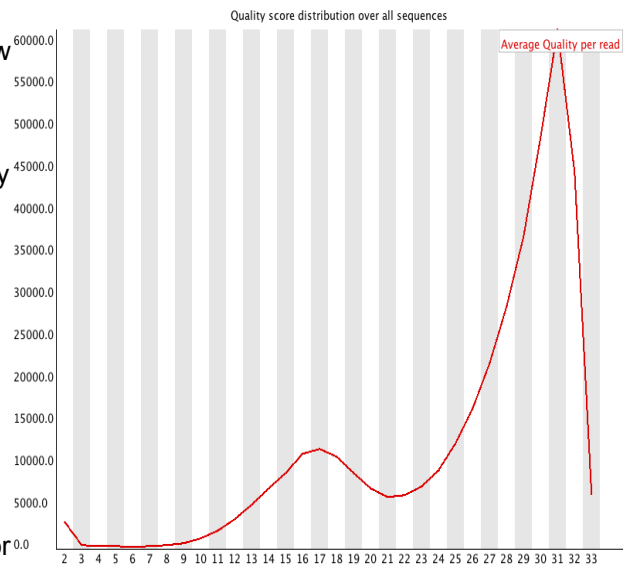
If a significant proportion of the sequences in a run have overall low quality then this could indicate some kind of systematic problem - possibly with just part of the run (for example one end of a flowcell).

Warning

A warning is raised if the most frequently observed mean quality is below 27 - this equates to a 0.2% error rate.

Failure

An error is raised if the most frequently observed mean quality is below 20 - this equates to a 1% error rate.



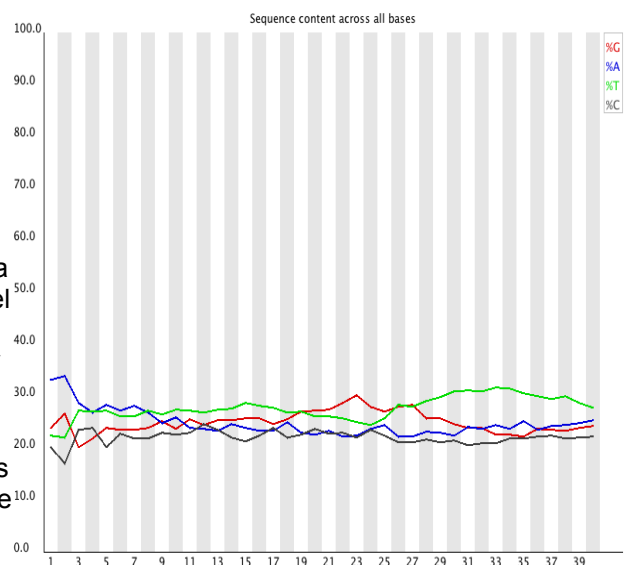
Per Base Sequence Content

Summary

Per Base Sequence Content plots out the proportion of each base position in a file for which each of the four normal DNA bases has been called.

In a random library you would expect that there would be little to no difference between the different bases of a sequence run, so the lines in this plot should run parallel with each other. The relative amount of each base should reflect the overall amount of these bases in your genome, but in any case they should not be hugely imbalanced from each other.

If you see strong biases which change in different bases then this usually indicates an overrepresented sequence which is contaminating your library. A bias which is consistent across all bases either indicates that the



original library was sequence biased, or that there was a systematic problem during the sequencing of the library.

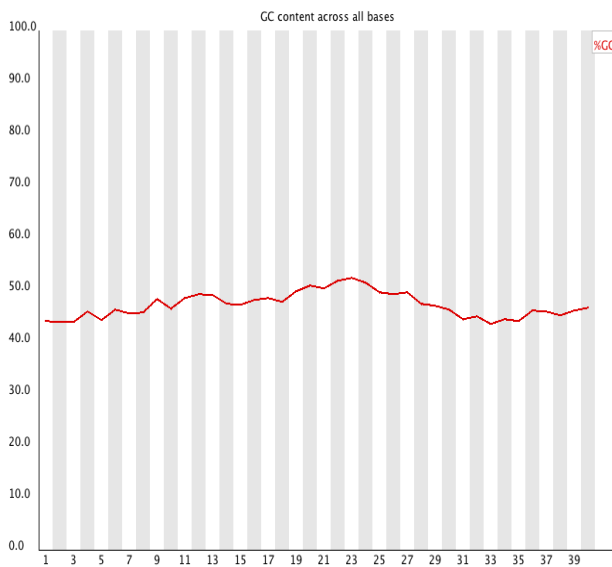
Warning

This module issues a warning if the difference between A and T, or G and C is greater than 10% in any position.

Failure

This module will fail if the difference between A and T, or G and C is greater than 20% in any position.

Per Base GC Content



Summary

Per Base GC Content plots out the GC content of each base position in a file.

In a random library you would expect that there would be little to no difference between the different bases of a sequence run, so the line in this plot should run horizontally across the graph. The overall GC content should reflect the GC content of the underlying genome.

If you see a GC bias which changes in different bases then this could indicate an overrepresented sequence which is contaminating your library. A bias which is consistent across all bases either indicates that the original library was sequence biased, or that there was a systematic problem during the sequencing of the library.

Warning

This module issues a warning if the GC content of any base strays more than 5% from the mean GC content.

Failure

This module will fail if the GC content of any base strays more than 10% from the mean GC content.

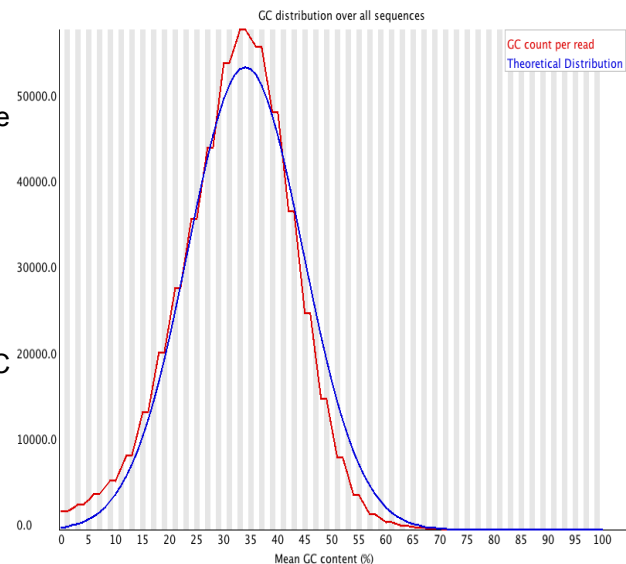
Per Sequence GC Content

Summary

This module measures the GC content across the whole length of each sequence in a file and compares it to a modelled normal distribution of GC content.

In a normal random library you would expect to see a roughly normal distribution of GC content where the central peak corresponds to the overall GC content of the underlying genome. Since we don't know the the GC content of the genome the modal GC content is calculated from the observed data and used to build a reference distribution.

An unusually shaped distribution could indicate a contaminated library or some other kinds of biased



subset. A normal distribution which is shifted indicates some systematic bias which is independent of base position. If there is a systematic bias which creates a shifted normal distribution then this won't be flagged as an error by the module since it doesn't know what your genome's GC content should be.

Warning

A warning is raised if the sum of the deviations from the normal distribution represents more than 15% of the reads.

Failure

This module will indicate a failure if the sum of the deviations from the normal distribution represents more than 30% of the reads.

Per Base N Content

Summary

If a sequencer is unable to make a base call with sufficient confidence then it will normally substitute an N rather than a conventional base call

This module plots out the percentage of base calls at each position for which an N was called.

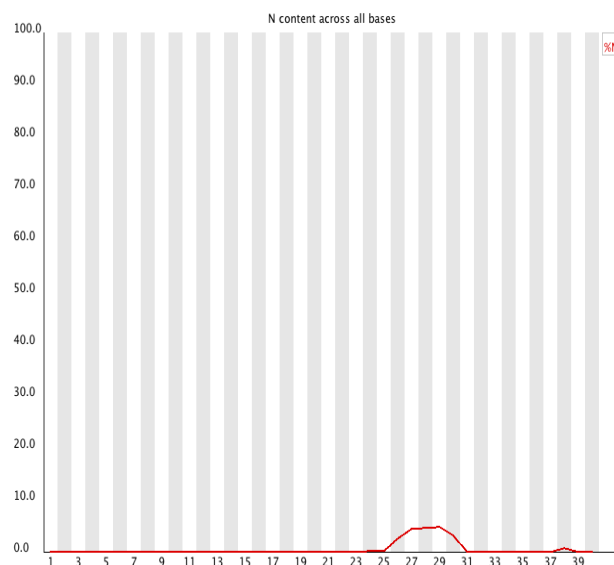
It's not unusual to see a very low proportion of Ns appearing in a sequence, especially nearer the end of a sequence. However, if this proportion rises above a few percent it suggests that the analysis pipeline was unable to interpret the data well enough to make valid base calls.

Warning

This module raises a warning if any position shows an N content of >5%.

Failure

This module will raise an error if any position shows an N content of >20%.



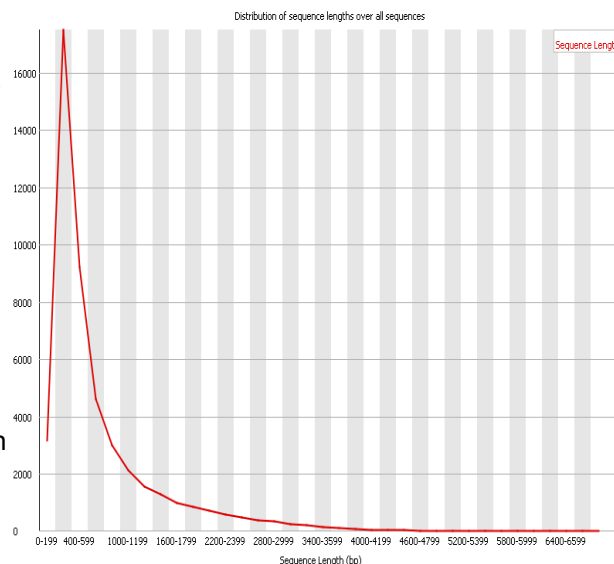
Sequence Length Distribution

Summary

Some high throughput sequencers generate sequence fragments of uniform length, but others can contain reads of wildly varying lengths. Even within uniform length libraries some pipelines will trim sequences to remove poor quality base calls from the end.

This module generates a graph showing the distribution of fragment sizes in the file which was analysed.

In many cases this will produce a simple graph showing a peak only at one size, but for variable length FastQ files this will show the relative amounts of each different size of sequence fragment.



Warning

This module will raise a warning if all sequences are not the same length.

Failure

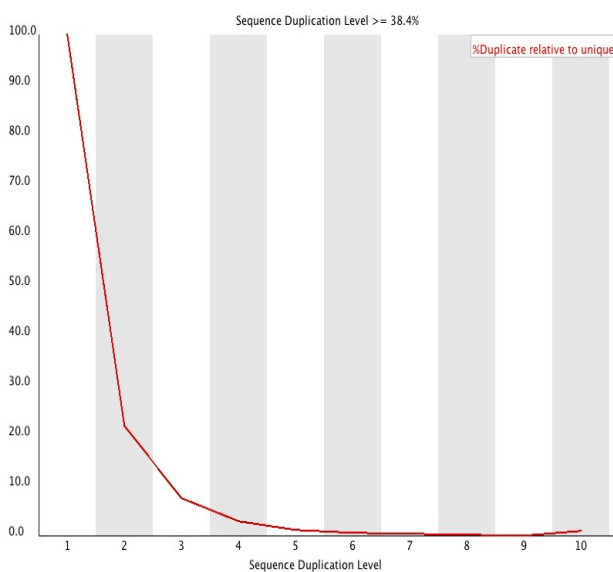
This module will raise an error if any of the sequences have zero length.

Duplicate Sequences

Summary

In a diverse library most sequences will occur only once in the final set. A low level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).

This module counts the degree of duplication for every sequence in the set and creates a plot showing the relative number of sequences with different degrees of duplication.



To cut down on the memory requirements for this module only sequences which occur in the first 200,000 sequences in each file are analysed, but this should be enough to get a good impression for the duplication levels in the whole file. Each sequence is tracked to the end of the file to give a representative count of the overall duplication level. To cut down on the amount of information in the final plot any sequences with more than 10 duplicates are placed into the 10 duplicates category - so it's not unusual to see a small rise in this final category. If you see a big rise in this final category then it means you have a large number of sequences with very high levels of duplication.

Because the duplication detection requires an exact sequence match over the whole length of the sequence any reads over 75bp in length are truncated to 50bp for the purposes of this analysis. Even so, longer reads are more likely to contain sequencing errors which will

artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

Warning

This module will issue a warning if non-unique sequences make up more than 20% of the total.

Failure

This module will issue a error if non-unique sequences make up more than 50% of the total.

Overrepresented Sequences

Summary

A normal high-throughput library will contain a diverse set of sequences, with no individual sequence making up a tiny fraction of the whole. Finding that a single sequence is very overrepresented in the set either means that it is highly biologically significant, or indicates that the library is contaminated, or not as diverse as you expected.

This module lists all of the sequence which make up more than 0.1% of the total. To conserve memory only sequences which appear in the first 200,000 sequences are tracked to the end of the file. It is therefore possible that a sequence which is overrepresented but doesn't appear at the start of the file for some reason

could be missed by this module.

For each overrepresented sequence the program will look for matches in a database of common contaminants and will report the best hit it finds. Hits must be at least 20bp in length and have no more than 1 mismatch. Finding a hit doesn't necessarily mean that this is the source of the contamination, but may point you in the right direction. It's also worth pointing out that many adapter sequences are very similar to each other so you may get a hit reported which isn't technically correct, but which has very similar sequence to the actual match.

Because the duplication detection requires an exact sequence match over the whole length of the sequence any reads over 75bp in length are truncated to 50bp for the purposes of this analysis. Even so, longer reads are more likely to contain sequencing errors which will artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

Warning

This module will issue a warning if any sequence is found to represent more than 0.1% of the total.

Failure

This module will issue an error if any sequence is found to represent more than 1% of the total.

Overrepresented Kmers

Summary

The analysis of overrepresented sequences will spot an increase in any exactly duplicated sequences, but there are a different subset of problems where it will not work.

- If you have very long sequences with poor sequence quality then random sequencing errors will dramatically reduce the counts for exactly duplicated sequences.
- If you have a partial sequence which is appearing at a variety of places within your sequence then this won't be seen either by the per base content plot or the duplicate sequence analysis.

This module counts the enrichment of every 5-mer within the sequence library. It calculates an expected level at which this k-mer should have been seen based on the base content of the library as a whole and then uses the actual count to calculate an observed/expected ratio for that k-mer. In addition to reporting a list of hits it will draw a graph for the top 6 hits to show the pattern of enrichment of that Kmer across the length of your reads. This will show if you have a general enrichment, or if there is a pattern of bias at different points over your read length.

Any k-mer showing more than a 3 fold overall enrichment or a 5 fold enrichment at any given base position will be reported by this module.

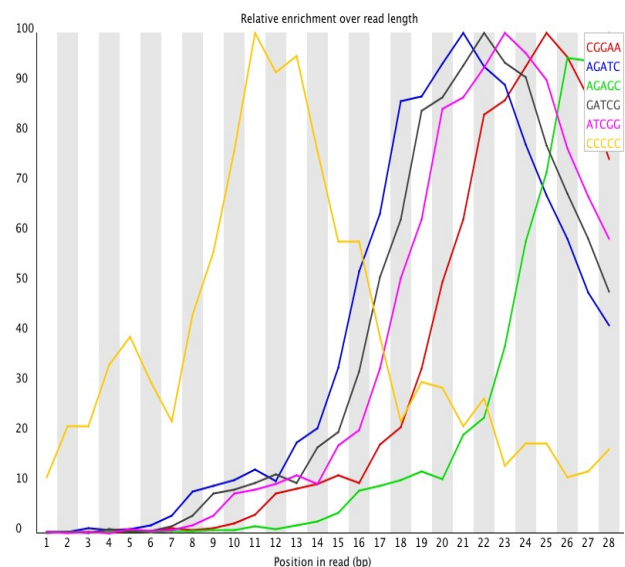
To allow this module to run in a reasonable time only 20% of the whole library is analysed and the results are extrapolated to the rest of the library.

Warning

This module will issue a warning if any k-mer is enriched more than 3 fold overall, or more than 5 fold at any individual position.

Failure

This module will issue a error if any k-mer is enriched more than 10 fold at any individual base position.



Fastx-toolkit command line instructions

http://hannonlab.cshl.edu/fastx_toolkit/commandline.html

Command Line Arguments

- [FASTQ-to-FASTA](#)
- [FASTQ/A Quality Statistics](#)
- [FASTQ Quality chart](#)
- [FASTQ/A Nucleotide Distribution chart](#)
- [FASTQ/A Clipper](#)
- [FASTQ/A Renamer](#)
- [FASTQ/A Trimmer](#)
- [FASTQ/A Collapser](#)
- [FASTQ/A Artifacts Filter](#)
- [FASTQ Quality Filter](#)
- [FASTQ/A Reverse Complement](#)
- [FASTA Formatter](#)
- [FASTA nucleotides changer](#)
- [FASTA Clipping Histogram](#)
- [FASTX Barcode Splitter](#)
- [Example: FASTQ Information](#)
- [Example: FASTQ/A manipulation](#)

Command Line Arguments

- Most tools show usage information with **-h**.
- Tools can read from STDIN and write to STDOUT, or from a specific input file (**-i**) and specific output file (**-o**).
- Tools can operate silently (producing no output if everything was OK), or print a short summary (**-v**).
If output goes to STDOUT, the summary will be printed to STDERR.
If output goes to a file, the summary will be printed to STDOUT.
- Some tools can compress the output with GZIP (**-z**).

FASTQ-to-FASTA

```
$ fastq_to_fasta -h
usage: fastq_to_fasta [-h] [-r] [-n] [-v] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6
[-h]           = This helpful help screen.
[-r]           = Rename sequence identifiers to numbers.
[-n]           = keep sequences with unknown (N) nucleotides.
                Default is to discard such sequences.
[-v]           = Verbose - report number of sequences.
                If [-o] is specified, report will be printed to STDOUT.
                If [-o] is not specified (and output goes to STDOUT),
                report will be printed to STDERR.
[-z]           = Compress output with GZIP.
[-i INFILE]    = FASTA/Q input file. default is STDIN.
[-o OUTFILE]   = FASTA output file. default is STDOUT.
```

FASTX Statistics

```
$ fastx_quality_stats -h
usage: fastx_quality_stats [-h] [-i INFILE] [-o OUTFILE]
```

version 0.0.6 (C) 2008 by Assaf Gordon (gordon@cshl.edu)

[-h] = This helpful help screen.
[-i INFILE] = FASTA/Q input file. default is STDIN.
 If FASTA file is given, only nucleotides
 distribution is calculated (there's no quality info).
[-o OUTFILE] = TEXT output file. default is STDOUT.

The output TEXT file will have the following fields (one row per column):

column = column number (1 to 36 for a 36-cycles read solexa file)
count = number of bases found in this column.
min = Lowest quality score value found in this column.
max = Highest quality score value found in this column.
sum = Sum of quality score values for this column.
mean = Mean quality score value for this column.
Q1 = 1st quartile quality score.
med = Median quality score.
Q3 = 3rd quartile quality score.
IQR = Inter-Quartile range (Q3-Q1).
lW = 'Left-Whisker' value (for boxplotting).
rW = 'Right-Whisker' value (for boxplotting).
A_Count = Count of 'A' nucleotides found in this column.
C_Count = Count of 'C' nucleotides found in this column.
G_Count = Count of 'G' nucleotides found in this column.
T_Count = Count of 'T' nucleotides found in this column.
N_Count = Count of 'N' nucleotides found in this column.
max-count = max. number of bases (in all cycles)

FASTQ Quality Chart

```
$ fastq_quality_boxplot_graph.sh -h  
Solexa-Quality BoxPlot plotter  
Generates a solexa quality score box-plot graph
```

Usage: /usr/local/bin/fastq_quality_boxplot_graph.sh [-i INPUT.TXT] [-t TITLE] [-p] [-o OUTPUT]

[-p] - Generate PostScript (.PS) file. Default is PNG image.
[-i INPUT.TXT] - Input file. Should be the output of
"solexa_quality_statistics" program.
[-o OUTPUT] - Output file name. default is STDOUT.
[-t TITLE] - Title (usually the solexa file name) - will be plotted on the
graph.

FASTA/Q Nucleotide Distribution

```
$ fastx_nucleotide_distribution_graph.sh -h  
FASTA/Q Nucleotide Distribution Plotter
```

Usage: /usr/local/bin/fastx_nucleotide_distribution_graph.sh [-i INPUT.TXT] [-t TITLE] [-p] [-o OUTPUT]

[-p] - Generate PostScript (.PS) file. Default is PNG image.
[-i INPUT.TXT] - Input file. Should be the output of "fastx_quality_statistics"
program.
[-o OUTPUT] - Output file name. default is STDOUT.
[-t TITLE] - Title - will be plotted on the graph.

FASTA/Q Clipper

```
$ fastx_clipper -h  
usage: fastx_clipper [-h] [-a ADAPTER] [-D] [-l N] [-n] [-d N] [-c] [-C] [-o] [-v] [-z] [-i INFILE] [-o OUTFILE]
```

version 0.0.6

[-h] = This helpful help screen.
[-a ADAPTER] = ADAPTER string. default is CCTTAAGG (dummy adapter).

[-l N] = discard sequences shorter than N nucleotides. default is 5.
 [-d N] = Keep the adapter and N bases after it.
 (using '-d 0' is the same as not using '-d' at all. which is the default).
 [-c] = Discard non-clipped sequences (i.e. - keep only sequences which contained the adapter).
 [-C] = Discard clipped sequences (i.e. - keep only sequences which did not contained the adapter).
 [-k] = Report Adapter-Only sequences.
 [-n] = keep sequences with unknown (N) nucleotides. default is to discard such sequences.
 [-v] = Verbose - report number of sequences.
 If [-o] is specified, report will be printed to STDOUT.
 If [-o] is not specified (and output goes to STDOUT), report will be printed to STDERR.
 [-z] = Compress output with GZIP.
 [-D] = DEBUG output.
 [-i INFILE] = FASTA/Q input file. default is STDIN.
 [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Renamer

\$ **fastx_renamer -h**

usage: fastx_renamer [-n TYPE] [-h] [-z] [-v] [-i INFILE] [-o OUTFILE]
 Part of FASTX Toolkit 0.0.10 by A. Gordon (gordon@cshl.edu)

[-n TYPE] = rename type:
 SEQ - use the nucleotides sequence as the name.
 COUNT - use simply counter as the name.
 [-h] = This helpful help screen.
 [-z] = Compress output with GZIP.
 [-i INFILE] = FASTA/Q input file. default is STDIN.
 [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Trimmer

\$ **fastx_trimmer -h**

usage: fastx_trimmer [-h] [-f N] [-l N] [-z] [-v] [-i INFILE] [-o OUTFILE]

version 0.0.6

[-h] = This helpful help screen.
 [-f N] = First base to keep. Default is 1 (=first base).
 [-l N] = Last base to keep. Default is entire read.
 [-z] = Compress output with GZIP.
 [-i INFILE] = FASTA/Q input file. default is STDIN.
 [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Collapser

\$ **fastx_collapser -h**

usage: fastx_collapser [-h] [-v] [-i INFILE] [-o OUTFILE]

version 0.0.6

[-h] = This helpful help screen.
 [-v] = verbose: print short summary of input/output counts
 [-i INFILE] = FASTA/Q input file. default is STDIN.
 [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTQ/A Artifacts Filter

\$ **fastx_artifacts_filter -h**

usage: fastq_artifacts_filter [-h] [-v] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6

[-h] = This helpful help screen.
 [-i INFILE] = FASTA/Q input file. default is STDIN.

```
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.
[-z]          = Compress output with GZIP.
[-v]          = Verbose - report number of processed reads.
               If [-o] is specified, report will be printed to STDOUT.
               If [-o] is not specified (and output goes to STDOUT),
               report will be printed to STDERR.
```

FASTQ Quality Filter

```
$ fastq_quality_filter -h
```

```
usage: fastq_quality_filter [-h] [-v] [-q N] [-p N] [-z] [-i INFILE] [-o OUTFILE]
```

```
version 0.0.6
```

```
[-h]          = This helpful help screen.
[-q N]        = Minimum quality score to keep.
[-p N]        = Minimum percent of bases that must have [-q] quality.
[-z]          = Compress output with GZIP.
[-i INFILE]   = FASTA/Q input file. default is STDIN.
[-o OUTFILE]  = FASTA/Q output file. default is STDOUT.
[-v]          = Verbose - report number of sequences.
               If [-o] is specified, report will be printed to STDOUT.
               If [-o] is not specified (and output goes to STDOUT),
               report will be printed to STDERR.
```

FASTQ/A Reverse Complement

```
$ fastx_reverse_complement -h
```

```
usage: fastx_reverse_complement [-h] [-r] [-z] [-v] [-i INFILE] [-o OUTFILE]
```

```
version 0.0.6
```

```
[-h]          = This helpful help screen.
[-z]          = Compress output with GZIP.
[-i INFILE]   = FASTA/Q input file. default is STDIN.
[-o OUTFILE]  = FASTA/Q output file. default is STDOUT.
```

FASTA Formatter

```
$ fasta_formatter -h
```

```
usage: fasta_formatter [-h] [-i INFILE] [-o OUTFILE] [-w N] [-t] [-e]
```

```
Part of FASTX Toolkit 0.0.7 by gordon@cshl.edu
```

```
[-h]          = This helpful help screen.
[-i INFILE]   = FASTA/Q input file. default is STDIN.
[-o OUTFILE]  = FASTA/Q output file. default is STDOUT.
[-w N]        = max. sequence line width for output FASTA file.
               When ZERO (the default), sequence lines will NOT be wrapped -
               all nucleotides of each sequences will appear on a single
               line (good for scripting).
[-t]          = Output tabulated format (instead of FASTA format).
               Sequence-Identifiers will be on first column,
               Nucleotides will appear on second column (as single line).
[-e]          = Output empty sequences (default is to discard them).
               Empty sequences are ones who have only a sequence identifier,
               but not actual nucleotides.
```

```
Input Example:
```

```
>MY-ID
AAAAAGGGGG
CCCCCTTTT
AGCTN
```

```
Output example with unlimited line width [-w 0]:
```

```
>MY-ID
AAAAAGGGGGCCCCCTTTTtagctn
```

```
Output example with max. line width=7 [-w 7]:
```

```
>MY-ID
```



```
AAAAAGG
GGGTTTT
TCCCCCA
GCTN
```

Output example with tabular output [-t]:
MY-ID AAAAAGGGGGCCCCCTTTTAGCTN

example of empty sequence:
(will be discarded unless [-e] is used)
>REGULAR-SEQUENCE-1
AAAGGGTTTCCC
>EMPTY-SEQUENCE
>REGULAR-SEQUENCE-2
AAGTAGTAGTAGTAGT
GTATTTTATAT

FASTA Nucleotides Changer

```
$ fasta_nucleotide_changer -h
usage: fasta_nucleotide_changer [-h] [-z] [-v] [-i INFILE] [-o OUTFILE] [-r] [-d]

version 0.0.7
[-h]            = This helpful help screen.
[-z]            = Compress output with GZIP.
[-v]            = Verbose mode. Prints a short summary.
                 with [-o], summary is printed to STDOUT.
                 Otherwise, summary is printed to STDERR.
[-i INFILE]    = FASTA/Q input file. default is STDIN.
[-o OUTFILE]   = FASTA/Q output file. default is STDOUT.
[-r]           = DNA-to-RNA mode - change T's into U's.
[-d]           = RNA-to-DNA mode - change U's into T's.
```

FASTA Clipping Histogram

```
$ fasta_clipping_histogram.pl
Create a Linker Clipping Information Histogram

usage: fasta_clipping_histogram.pl INPUT_FILE.FA OUTPUT_FILE.PNG

      INPUT_FILE.FA    = input file (in FASTA format, can be GZIPped)
      OUTPUT_FILE.PNG = histogram image
```

FASTX Barcode Splitter

```
$ fastx_barcode_splitter.pl
Barcode Splitter, by Assaf Gordon (gordon@cshl.edu), 11sep2008

This program reads FASTA/FASTQ file and splits it into several smaller files,
Based on barcode matching.
FASTA/FASTQ data is read from STDIN (format is auto-detected.)
Output files will be written to disk.
Summary will be printed to STDOUT.

usage: /usr/local/bin/fastx_barcode_splitter.pl --bcfile FILE --prefix PREFIX [--
suffix SUFFIX] [--bol|--eol]
          [--mismatches N] [--exact] [--partial N] [--help] [--quiet] [--debug]

Arguments:

--bcfile FILE    - Barcodes file name. (see explanation below.)
--prefix PREFIX - File prefix. will be added to the output files. Can be used
                 to specify output directories.
--suffix SUFFIX - File suffix (optional). Can be used to specify file
                 extensions.
--bol           - Try to match barcodes at the BEGINNING of sequences.
                 (What biologists would call the 5' end, and programmers
```

would call index 0.)

```
--eol          - Try to match barcodes at the END of sequences.
                (What biologists would call the 3' end, and programmers
                would call the end of the string.)
                NOTE: one of --bol, --eol must be specified, but not both.
--mismatches N - Max. number of mismatches allowed. default is 1.
--exact        - Same as '--mismatches 0'. If both --exact and --mismatches
                are specified, '--exact' takes precedence.
--partial N    - Allow partial overlap of barcodes. (see explanation below.)
                (Default is not partial matching)
--quiet        - Don't print counts and summary at the end of the run.
                (Default is to print.)
--debug        - Print lots of useless debug information to STDERR.
--help         - This helpful help screen.
```

Example (Assuming 's_2_100.txt' is a FASTQ file, 'mybarcodes.txt' is the barcodes file):

```
$ cat s_2_100.txt | /usr/local/bin/fastx_barcode_splitter.pl --bcfile
mybarcodes.txt --bol --mismatches 2 \
  --prefix /tmp/bla_ --suffix ".txt"
```

Barcode file format

Barcode files are simple text files. Each line should contain an identifier (descriptive name for the barcode), and the barcode itself (A/C/G/T), separated by a TAB character. Example:

```
#This line is a comment (starts with a 'number' sign)
BC1 GATCT
BC2 ATCGT
BC3 GTGAT
BC4 TGTCT
```

For each barcode, a new FASTQ file will be created (with the barcode's identifier as part of the file name). Sequences matching the barcode will be stored in the appropriate file.

Running the above example (assuming "mybarcodes.txt" contains the above barcodes), will create the following files:

```
/tmp/bla_BC1.txt
/tmp/bla_BC2.txt
/tmp/bla_BC3.txt
/tmp/bla_BC4.txt
/tmp/bla_unmatched.txt
```

The 'unmatched' file will contain all sequences that didn't match any barcode.

Barcode matching

** Without partial matching:

Count mismatches between the FASTA/Q sequences and the barcodes. The barcode which matched with the lowest mismatches count (providing the count is small or equal to '--mismatches N') 'gets' the sequences.

Example (using the above barcodes):

```
Input Sequence:
  GATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
```

```
Matching with '--bol --mismatches 1':
  GATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
  GATCT (1 mismatch, BC1)
  ATCGT (4 mismatches, BC2)
  GTGAT (3 mismatches, BC3)
  TGTCT (3 mismatches, BC4)
```

This sequence will be classified as 'BC1' (it has the lowest mismatch count). If '--exact' or '--mismatches 0' were specified, this sequence would be classified as 'unmatched' (because, although BC1 had the lowest mismatch count, it is above the maximum allowed mismatches).

Matching with '--eol' (end of line) does the same, but from the other side of the sequence.

** With partial matching (very similar to indels):

Same as above, with the following addition: barcodes are also checked for partial overlap (number of allowed non-overlapping bases is '--partial N').

Example:

Input sequence is ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG

(Same as above, but note the missing 'G' at the beginning.)

Matching (without partial overlapping) against BC1 yields 4 mismatches:

ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG

GATCT (4 mismatches)

Partial overlapping would also try the following match:

-ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG

GATCT (1 mismatch)

Note: scoring counts a missing base as a mismatch, so the final mismatch count is 2 (1 'real' mismatch, 1 'missing base' mismatch). If running with '--mismatches 2' (meaning allowing upto 2 mismatches) - this sequence will be classified as BC1.