FastQC Report Generator: Technical Documentation

FastQC Report Generator is a user-friendly command-line utility for generating report files and visualisations from FastQC text files. It is developed and tested in Python 3.8 using the scientific python libraries: matplotlib, seaborn, pandas, NumPy, SciPy, in addition to os, sys, abc and argparse from the Python Standard Library.

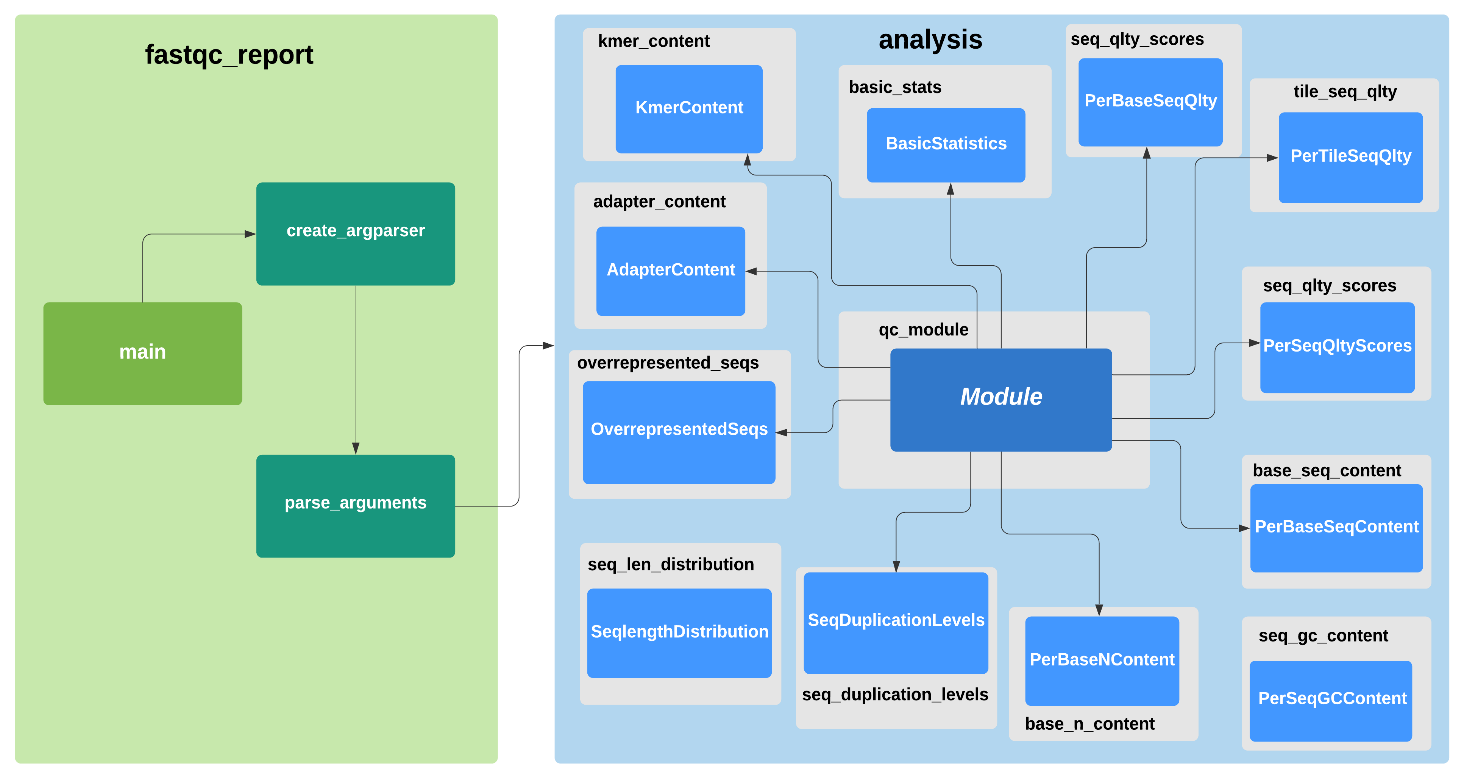
External modules used:

1. **NumPy** (https://numpy.org/) - to generate ndarrays (Harris et al., 2020).
2. **Pandas** (<https://pandas.pydata.org/>) - for tabulation of parsed data and data preparation for plotting (McKinney et al., 2010).
3. **Matplotlib** (<https://matplotlib.org/>) - to plot figures (Hunter, 2007).
4. **Seaborn** (<https://seaborn.pydata.org/>) - to plot graphs (Waskom, 2021).
5. **SciPy** library (<https://www.scipy.org/scipylib/index.html>) - to fit a normal distribution to data (Virtanen et al., 2020).

# 1. General Structure

FastQC Report Generator is designed to parse data from FastQC report files, that correspond to the various modular analyses that FastQC performs when assessing the quality of raw sequencing data from high throughput pipelines (Andrews, 2017). A complete list of all analysis modules is shown in Table 1. In FastQC Report Generator, a user inputs an input file path for the FastQC file to be parsed, an output directory path in which the report and visualisation files will be generated and a compulsory basic statistics command-line argument to print the basic statistics of the FastQC file to the command-line (the Basic Statistics module data). Users can input optional command-line arguments for specific FastQC modules or users can input an “-all” flag to generates reports and visualisations for all FastQC modules.

The program is arranged into a command-line executable script (fastqc\_report.py) and a package of analysis modules corresponding to each of the 12 FastQC modular analyses (figure 1.).



**Figure 1. Interaction between command-line script (green panel) and the analysis package (blue panel) within FastQC Report Generator.** The create\_argparser function in the fastqc\_report command-line script creates an ArgumentParser object that is parsed by the parse\_arguments function, which then calls the corresponding analysis module (highlighted in grey) and respective class (light blue squares) representing the FastQC analysis module. The abstract Module class in qc\_module module provides shared I/O functionality including, module directory creation, FastQC parsing functionality and filter and report text file methods for all subclasses in the package.

# 2. Command-Line Interface

As a command-line parsing tool, FastQC Report Generator allows users to select specific QC modules from an input FastQC text file for parsing at the command-line and generate FastQC-styled graphs, in addition to report and filter text files in a module-named directory, nested inside a user-specified output directory. The program facilitates rapid user-specified QC module data extraction from FastQC report text files generated from performing FastQC on various sequencing files.

The argparse module (https://docs.python.org/3/library/argparse.html) was used to construct the command-line interface, enabling parsing of command-line arguments provided by the user when starting FastQC Report Generator. All help and usage error messages displayed when users input invalid arguments are automatically provided by argparse.

## 2.1. Command Line Helper Functions

The command line script encloses two helper functions for handling command-line arguments: **create\_argparser** and **process\_args**.

### 2.1. Create\_argparser

The function create\_argparser creates an ArgumentParser Object which stores the required information for parsing the command-line input into Python data types. The input file path, output directory and basic statistics arguments are compulsory positional arguments.

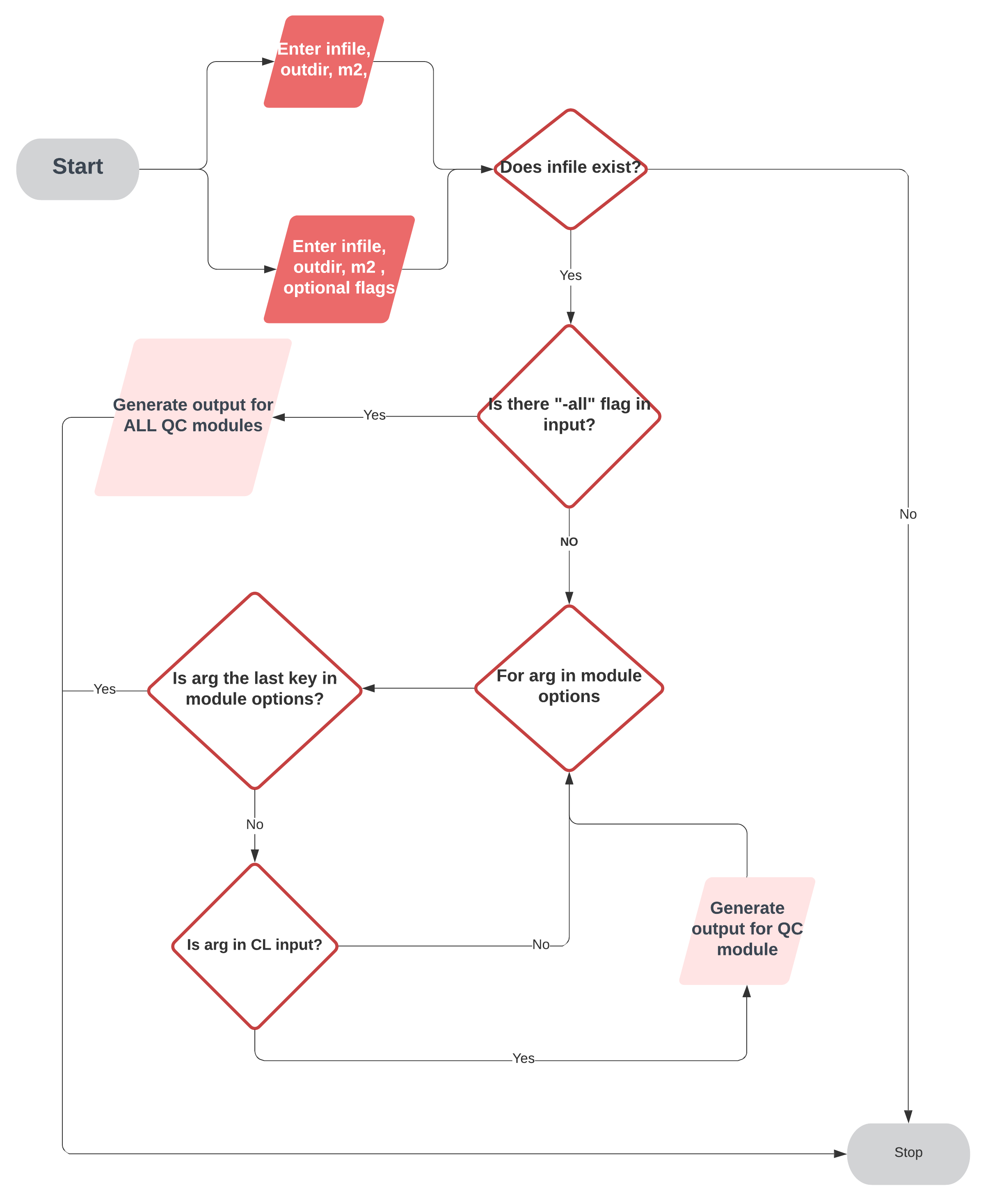
### 2.2. Process\_args

The function process\_args uses the namespace object returned from calling parse\_args on the ArgumentParser and invokes the appropriate action to generate the required output for the user-specified arguments (Table 1).

A local variable **module\_options** in the function uses each optional argument as a key that is mapped to a list consisting of the parsed attribute and its corresponding class object.

#### 2.2.1. Processing Arguments Algorithm

The algorithm for argument processing executed by the process\_args is illustrated in Figure 2. The function first validates the presence of “file” and “outdir” attributes in the args Namespace object (args.file and args.outdir) and the BasicStatistics class is instantiated with the values of these attributes. If the file path of args.file (input FastQC file) is not found then a FileNotFoundError exception is raised, the program notifies the user on the command-line and then the program exits with error code 1. On the contrary, if the file path of args.file exists then the program checks whether the –all flag was used in the command-line (stored in args as args.all\_modules). If “args.all\_modules” exists, output for all modules is generated by looping through module\_options keys and instantiating all the analysis package subclasses that represent FastQC modular analyses using the input file and output directory. If the “all” flag is absent in the parsed command-line input, the command-line arguments for optional modules (positioned at the first index of module\_options dictionary) are looped through and its corresponding class is instantiated using the input file and output directory, allowing generation of module output for selected QC modules using the module’s module\_output method (see Figure 2.).



**Figure 1. Flowchart illustrating algorithm for command-line processing in FastQC Report Generator.** Infile, input FastQC file; outdir, output directory, m2; basic statistics; CL, command-line

**Table 1. Output generated for FastQC analysis modules when running FastQC Report Generator.** QCR, QC report; G, Graph; F, Filter text

|  |  |
| --- | --- |
| Analysis Module | Output |
| Basic Statistics | Screen display of data |
| Per base sequence quality | QCR, G, F |
| Per tile sequence quality | QCR, G, F |
| Per sequence quality scores | QCR, G, F |
| Per base sequence Content | QCR, G, F |
| Per sequence GC content | QCR, G, F |
| Per base N content | QCR, G, F |
| Per sequence length distribution | QCR, G, F |
| Sequence Duplication Levels | QCR, G, F |
| Overrepresented Seqs | QCR, F |
| Adaptor content | QCR, G, F |
| K-mer content | QCR, G, F |

# 3. Analysis Package

FastQC Report Generator contains a package of a single module called qc\_module and another 12 modules each containing a class corresponding to a FastQC analysis module (see figure 1).

### 3.1 Module Class

The Module class is an abstract class providing generic and shared functionalities for all FastQC analysis modules. The class was designed as an abstract class, to enforce polymorphism of the module output method and to share common attributes and methods with the FastQC subclasses. All 12 subclasses override the abstract method, **module\_output**.

### 3.1.1 Attributes

The Module class has five Instance variables that are inherited by all 12 FastQC subclasses when they are instantiated:

* **lines** – a list holder variable that stores parsed lines
* **name** – string with the name of the QC module
* **dir\_name** – string with module directory name
* **infile** – input file path of FastQC file to be parsed
* **outdir** – file path of output directory in which all file output from the program will be generated.

### 3.1.2 Methods

The Module class also has the following six methods:

* **parser\_text** – utilises the “>>” delimiting feature for QC modules in FastQC text files and appends the parsed lines to the lines instance variable.
* **make\_dir** – generates a directory named after the QC module specified within the user-specified output directory using makedirs function from the os module.
* **create\_report** – creates a report text file named “QC\_report.txt” containing parsed lines for the user-specified QC module. The file path for the report file is joined to the module directory path generated via the make\_dir method using the os.path submodule.
* **create\_filter\_text** – parses out filter information (either *pass*/*warn*/*fail*) from the input FastQC file and writes it to a text file named “filter.txt” in the output directory.
* **Clean\_lines** – returns a tuple of cleaned lines list and columns list
* **module\_output** – abstract method that is overridden and extended by all subclasses of Module and used as a second-order function calling various methods to generate the module output for a given QC module. In the Module class, the method calls parse\_text and prints a statement on the command line indicating output is being generated for the FastQC module.

## 3.2. BasicStatistics

BasicStatistics contains methods for generating output for Basic Statistics module:

* **display\_stats** – prints parsed basic statistics information from input file on the command-line, by joining the split lines in the lines list.
* **module\_output** – generates output for basic statistics by calling display\_stats method.

## 3.3. PerBaseSeqQlty

PerBaseSeqQlty contains methods for generating output for the Per base sequence quality FastQC module:

* **get\_encoding** – extracts type of quality score encoding information from the input FastQC file.
* **prep\_data** – prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates a FastQC-styled box and whisker plot showing quality values across all base positions in the read is created using matplotlib, saved in PNG format
* **module\_output** – generates output for Per base sequence quality module

### 3.4. PerTileSeqQlty

PerTileSeqQlty contains methods for generating output for the Per tile sequence quality FastQC module:

* **Prep\_data** – prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph –** generates a heatmap for deviation of quality per tile using seaborn, saved in PNG format
* **Module\_output –** generates output for Per tile sequence quality module

### 3.5. PerSeqQltyScores

PerSeqQltyScores contains methods for generating output for the Per sequence quality scores FastQC module:

* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates a line plot for quality scores across all sequences using seaborn, saved in PNG format
* **Module\_output** – generates module output for Per sequence quality scores

### 3.6. PerBaseSeqContent

PerBaseSeqContent contains methods for generating output for the Per base sequence content FastQC module:

* **prep\_data -** prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph –** creates line plot showing proportion of bases calls at given base position using seaborn, saved in PNG format
* **Module\_output –** generates module output for Per base sequence content module

### 3.7. PerSeqGCContent

PerSeqGCContent contains methods for generating output for the Per sequence GC content FastQC module:

* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates a comparison line graph of GC content across the sequence and a modelled normal distribution of GC content using seaborn, saved in PNG format. SciPy was used to fit a normal distribution to the GC content counts.
* **Module\_output** – generates output for Per sequence GC content module

### 3.8. PerBaseNContent

PerBaseNContent contains methods for generating output for the Per base N content FastQC module.

* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates a line graph displaying the number of base N calls at each position in the read using seaborn, saved in PNG format.
* **Module\_output** - generates report text file, filter text file and graph for Per base N content module

### 3.9. SeqLengthDistribution

PerBaseNContent contains methods for generating output for the Per base N content FastQC module:

* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **create\_graph** – creates a line plot showing sequence length distribution if there are multiple sequence lengths, otherwise it shows a single bar plot. Both graph types are generated using seaborn, saved in PNG format.
* **module\_output** - generates output for Sequence length distribution

### 3.10. SeqDuplicationLevels

SeqDuplicationLevels contains methods for generating output for the Sequence Duplication Levels FastQC module:

* **clean\_lines** – overrides Module superclass method to strip the hashtag (“#”) from the extra header line in the module data from the FastQC file.
* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates line graph showing proportion of sequences with varying degrees of duplication using seaborn, saved in PNG format.
* **Module\_output** – generates output for Sequence Duplication Levels module

### 3.11. OverrepresentedSeqs

OverrepresentedSeqs contains methods for generating output for the Overrepresented sequences FastQC module:

* **module\_output** – generates output for Overrepresented sequences FastQC module: a module directory, containing a report file and filter text file with Overrrepresented sequences data.

### 3.12. AdapterContent

AdapterContent contains methods for generating output for the Adapter content FastQC module:

* **prep\_data -** prepares and casts data to appropriate types for plotting, generating a dataframe.
* **create\_graph –** creates a line graph showing the cumulative percentage count of the library where an adaptor sequence was detected for a given position in the read using seaborn, saved in PNG format.
* **module\_output** – generates output for Adapter content module

### 3.13. KmerContent

KmerContent contains methods for generating output for the K-mer content FastQC module:

* **prep\_data** - prepares and casts data to appropriate types for plotting, generating a dataframe.
* **Create\_graph** – creates a line graph if the top six most biased kmers have a range of positions or a grouped bar plot if the top 6 kmers have data for only a single position. Both plots are created using seaborn, saved in PNG format.
* **Module\_output** – generates output for K-mer content module

# References

Andrews, S., 2017. FastQC: a quality control tool for high throughput sequence data. 2010.

Harris, C.R. et al., 2020. Array programming with NumPy. *Nature*, 585, pp.357–362.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(03), pp.90-95.

McKinney, W. & others, 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*. pp. 51–56.

Virtanen, P. et al., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, pp.261–272.

Waskom, M.L., 2021. Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), p.3021.