FastQCParser

FastQCParser User Guide:

It is recommended to open this via browser: User Guide

Introduction

A Python program to parse FastQCtext files, and generate reports and plots.

1. Clone directory: `git clone https://github.com/ms2206/FastQCParser.git`
2. Make a new python environment based from requirments.yaml `conda env create -f requirements.yaml --<NAME>`
3. Load environment env. `conda activate <NAME>`
4. Change directory into FastQCParser
5. Example Usage: `python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a`

-- plots are downloaded to the users `~/Downloads/` folder.

Set up

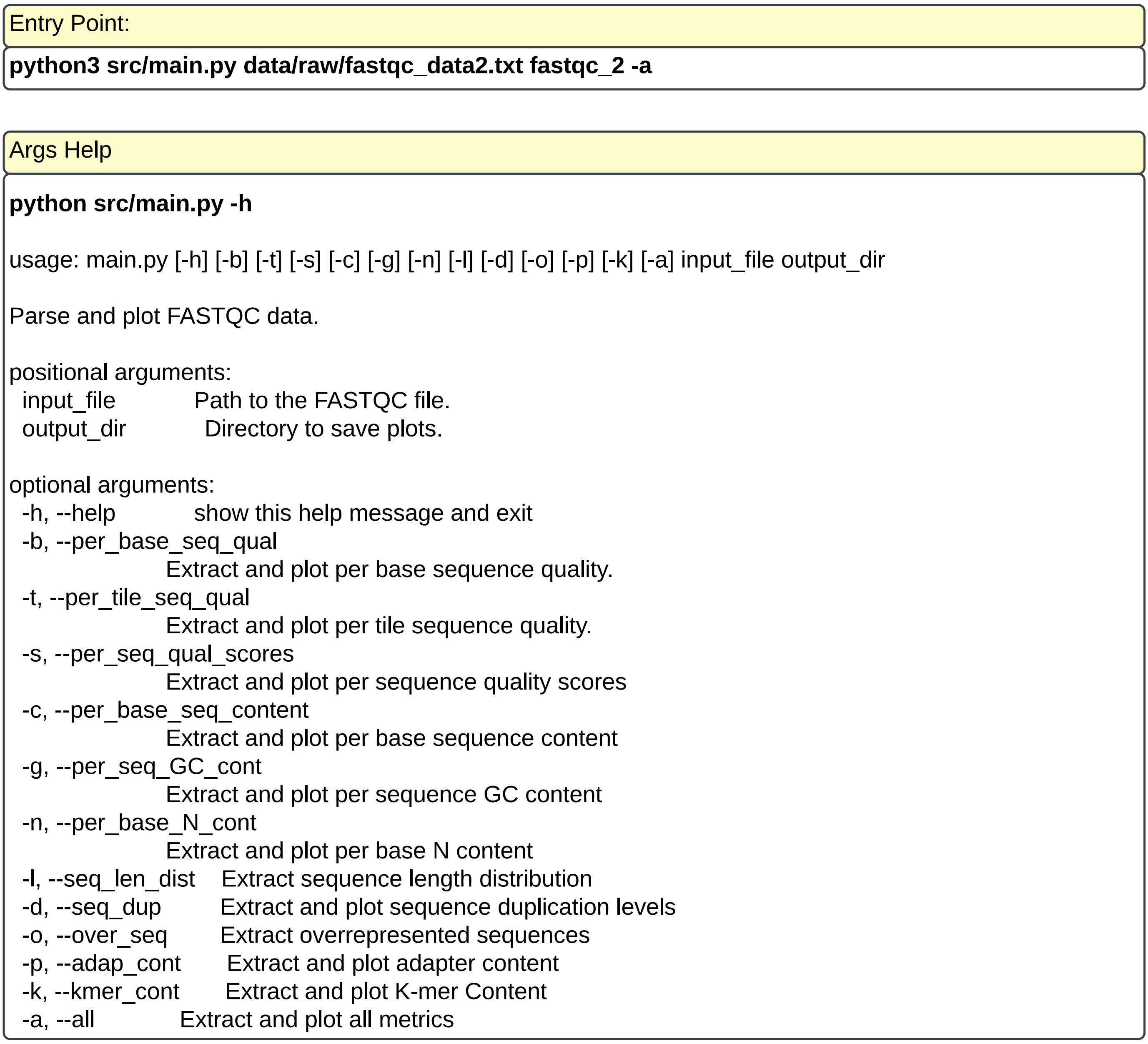
Example Usage: With python3, run executable found at `src/main.py` .

Pass `data/raw/fastqc_data2.txt` (or any fastqc file) - as input file, and `fastqc_2` as output directory (or use a customer directory name).

Use optional argument `-a` .

```
python3 src/main.py <FASTQ FILE> <DIR_NAME> [<OPTIONAL_ARGS>]
```

```
python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a
```



Entry Point:

python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a

Args Help

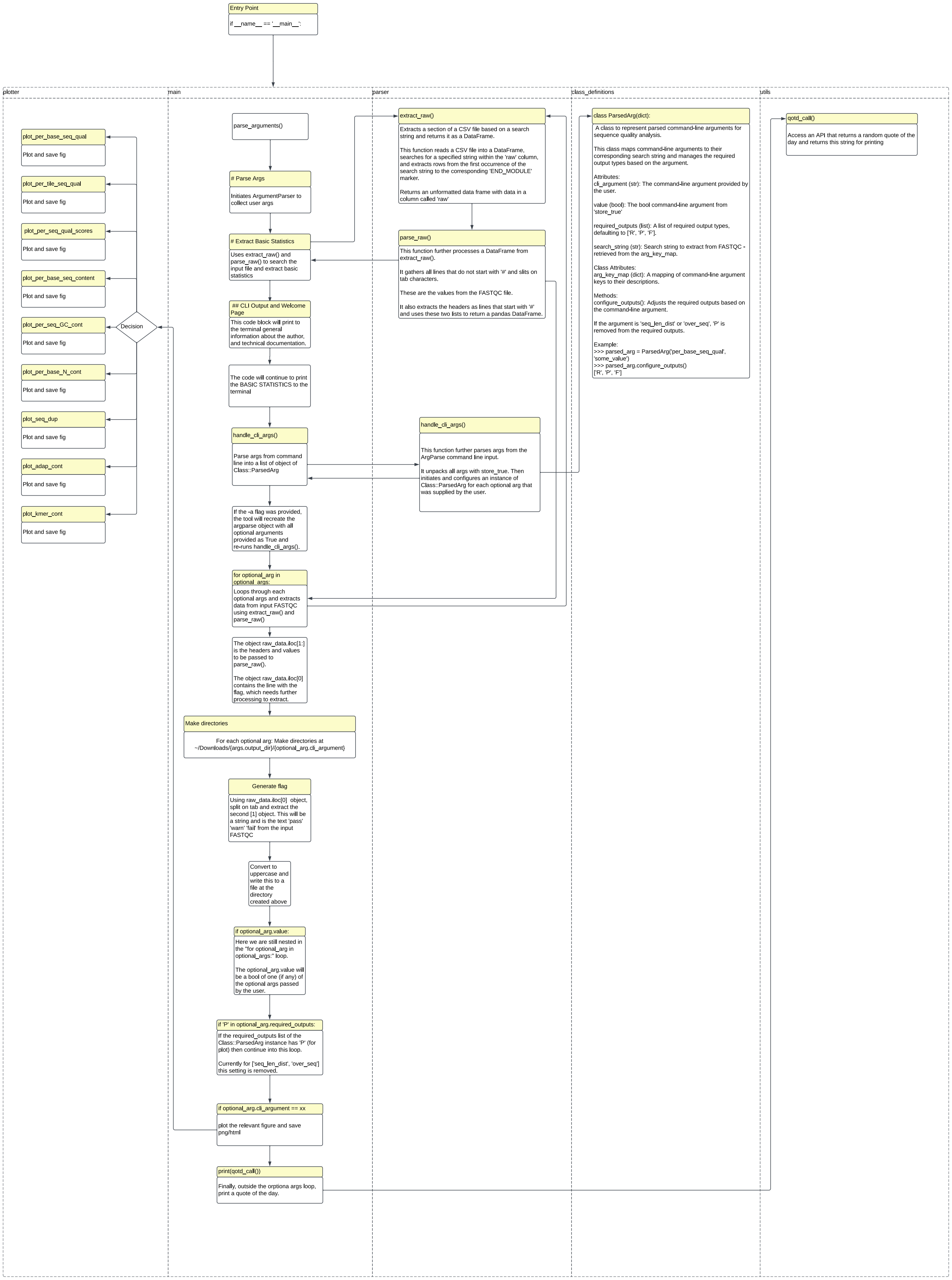
python src/main.py -h

```
usage: main.py [-h] [-b] [-t] [-s] [-c] [-g] [-n] [-l] [-d] [-o] [-p] [-k] [-a] input_file output_dir

Parse and plot FASTQC data.

positional arguments:
  input_file            Path to the FASTQC file.
  output_dir            Directory to save plots.

optional arguments:
  -h, --help            show this help message and exit
  -b, --per_base_seq_qual
                        Extract and plot per base sequence quality.
  -t, --per_tile_seq_qual
                        Extract and plot per tile sequence quality.
  -s, --per_seq_qual_scores
                        Extract and plot per sequence quality scores
  -c, --per_base_seq_content
                        Extract and plot per base sequence content
  -g, --per_seq_GC_cont
                        Extract and plot per sequence GC content
  -n, --per_base_N_cont
                        Extract and plot per base N content
  -l, --seq_len_dist    Extract sequence length distribution
  -d, --seq_dup         Extract and plot sequence duplication levels
  -o, --over_seq        Extract overrepresented sequences
  -p, --adap_cont       Extract and plot adapter content
  -k, --kmer_cont       Extract and plot K-mer Content
  -a, --all             Extract and plot all metrics
```

Optional Args

Help and misc information provided by ArgeParse for optional arguments.

```
python3 src/main.py -h
```



Entry Point

if __name__ == '__main__':

plotter

plot_per_base_seq_qual

Plot and save fig

plot_per_tile_seq_qual

Plot and save fig

plot_per_seq_qual_scores

Plot and save fig

plot_per_base_seq_content

Plot and save fig

plot_per_seq_GC_cont

Plot and save fig

plot_per_base_N_cont

Plot and save fig

plot_seq_dup

Plot and save fig

plot_adap_cont

Plot and save fig

plot_kmer_cont

Plot and save fig

Decision

main

parse_arguments()

Parse Args

Initiates ArgumentParser to collect user args

Extract Basic Statistics

Uses extract_raw() and parse_raw() to search the input file and extract basic statistics

CLI Output and Welcome Page

This code block will print to the terminal general information about the author, and technical documentation.

The code will continue to print the BASIC STATISTICS to the terminal

handle_cli_args()

Parse args from command line into a list of object of Class::ParsedArg

If the -a flag was provided, the tool will recreate the argparse object with all optional arguments provided as True and re-runs handle_cli_args().

for optional_arg in optional_args:

Loops through each optional args and extracts data from input FASTQC using extract_raw() and parse_raw()

The object raw_data.iloc[1:] is the headers and values to be passed to parse_raw().

The object raw_data.iloc[0] contains the line with the flag, which needs further processing to extract.

Make directories

For each optional arg: Make directories at ~/Downloads/{args.output_dir}/{optional_arg.cli_argument}

Generate flag

Using raw_data.iloc[0] object, split on tab and extract the second [1] object. This will be a string and is the text 'pass' 'warn' 'fail' from the input FASTQC

Convert to uppercase and write this to a file at the directory created above

if optional_arg.value:

Here we are still nested in the "for optional_arg in optional_args:" loop.

The optional_arg.value will be a bool of one (if any) of the optional args passed by the user.

if 'P' in optional_arg.required_outputs:

If the required_outputs list of the Class::ParsedArg instance has 'P' (for plot) then continue into this loop.

Currently for ['seq_len_dist', 'over_seq'] this setting is removed.

if optional_arg.cli_argument == xx

plot the relevant figure and save png/html

print(qotd_call())

Finally, outside the orptiona args loop, print a quote of the day.

parser

extract_raw()

Extracts a section of a CSV file based on a search string and returns it as a DataFrame.

This function reads a CSV file into a DataFrame, searches for a specified string within the 'raw' column, and extracts rows from the first occurrence of the search string to the corresponding 'END_MODULE' marker.

Returns an unformatted data frame with data in a column called 'raw'

parse_raw()

This function further processes a DataFrame from extract_raw().

It gathers all lines that do not start with '#' and slits on tab characters.

These are the values from the FASTQC file.

It also extracts the headers as lines that start with '#' and uses these two lists to return a pandas DataFrame.

handle_cli_args()

This function further parses args from the ArgParse command line input.

It unpacks all args with store_true. Then initiates and configures an instance of Class::ParsedArg for each optional arg that was supplied by the user.

class_definitions

class ParsedArg(dict):

A class to represent parsed command-line arguments for sequence quality analysis.

This class maps command-line arguments to their corresponding search string and manages the required output types based on the argument.

Attributes:
cli_argument (str): The command-line argument provided by the user.

value (bool): The bool command-line argument from 'store_true'

required_outputs (list): A list of required output types, defaulting to ['R', 'P', 'F'].

search_string (str): Search string to extract from FASTQC - retrieved from the arg_key_map.

Class Attributes:
arg_key_map (dict): A mapping of command-line argument keys to their descriptions.

Methods:
configure_outputs(): Adjusts the required outputs based on the command-line argument.

If the argument is 'seq_len_dist' or 'over_seq', 'P' is removed from the required outputs.

Example:

```
>>> parsed_arg = ParsedArg('per_base_seq_qual', 'some_value')
>>> parsed_arg.configure_outputs()
['R', 'P', 'F']
```

utils

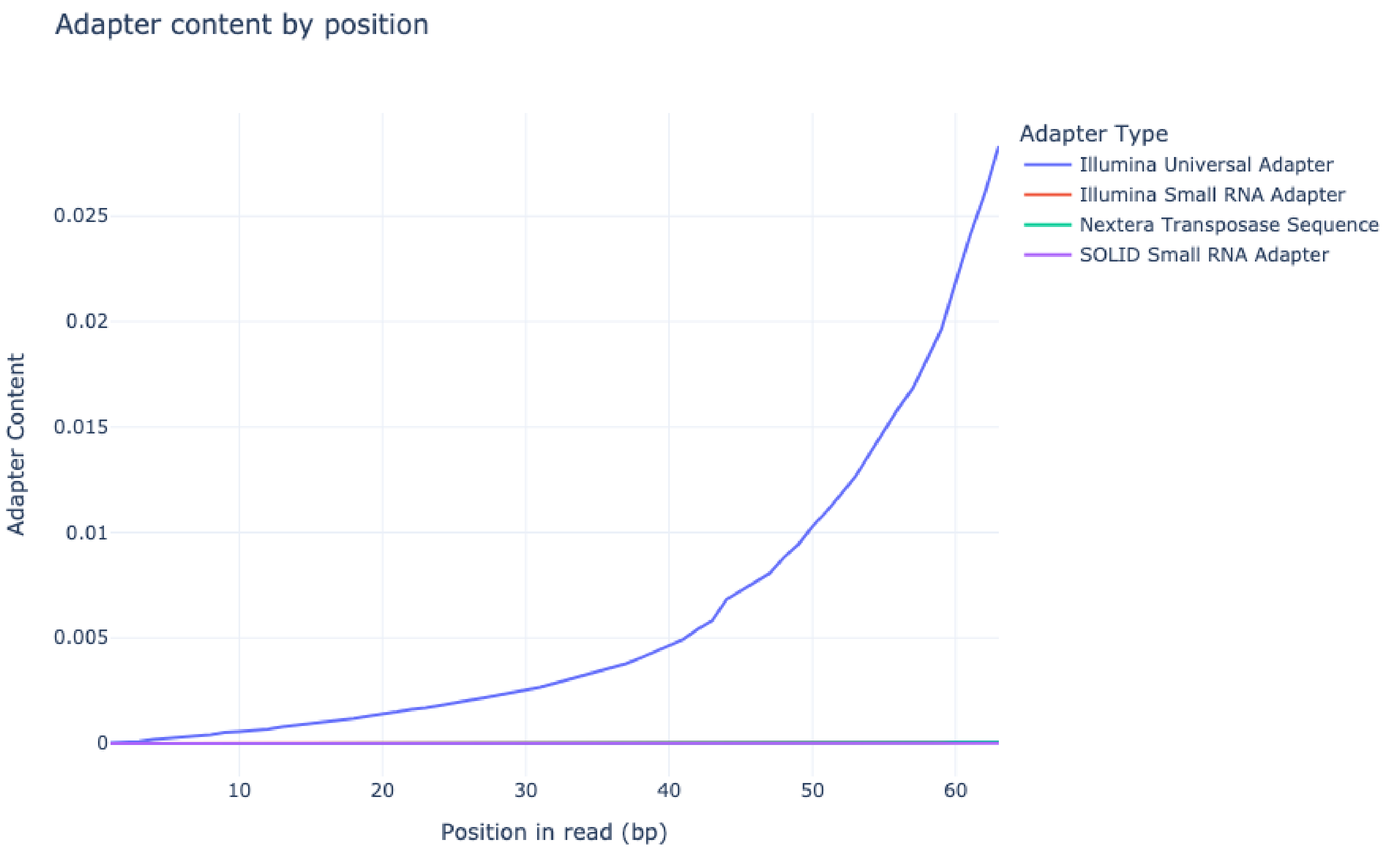
qotd_call()

Access an API that returns a random quote of the day and returns this string for printing

Example Plots

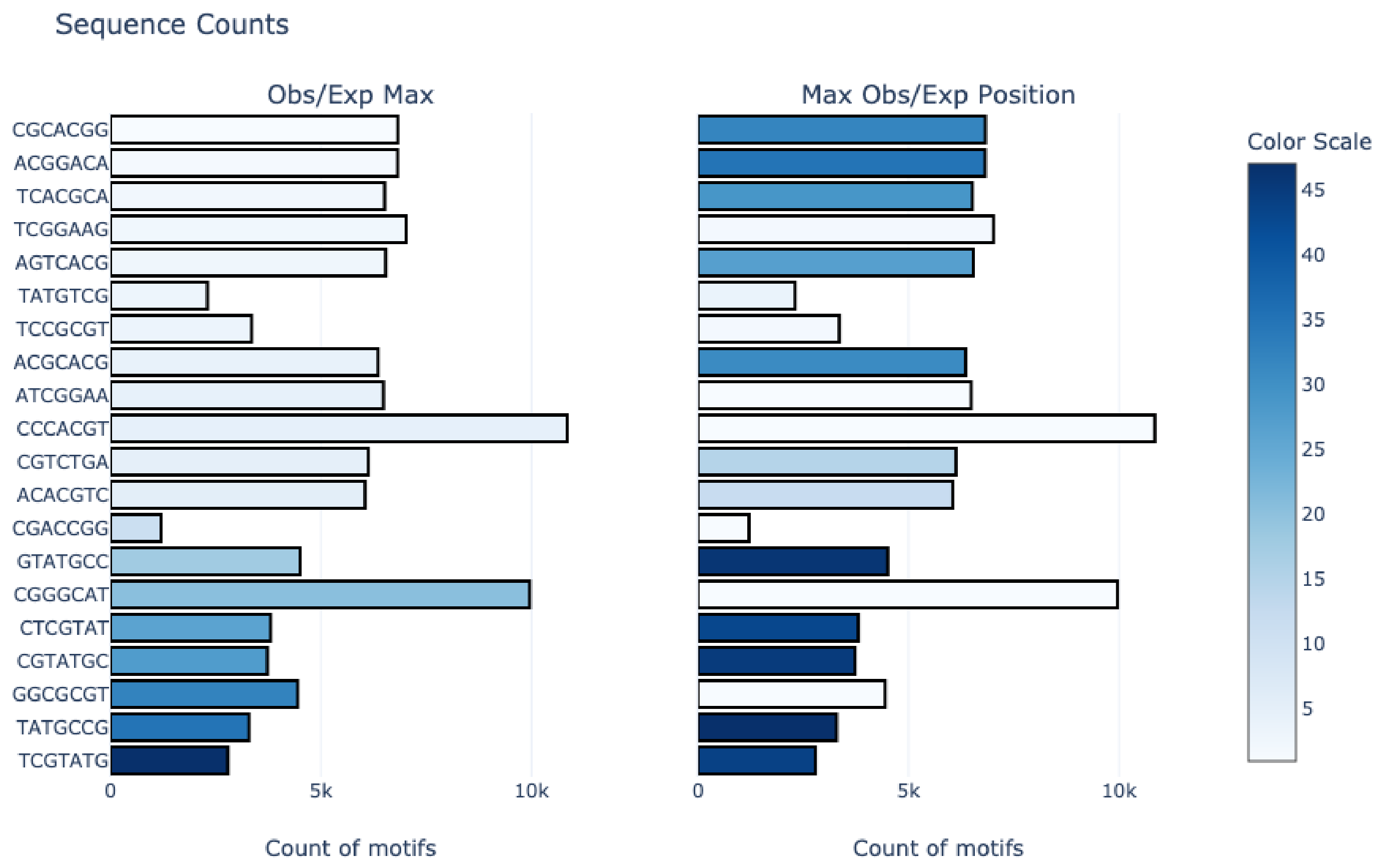
Adapter Content

Plot's adapter content by position.



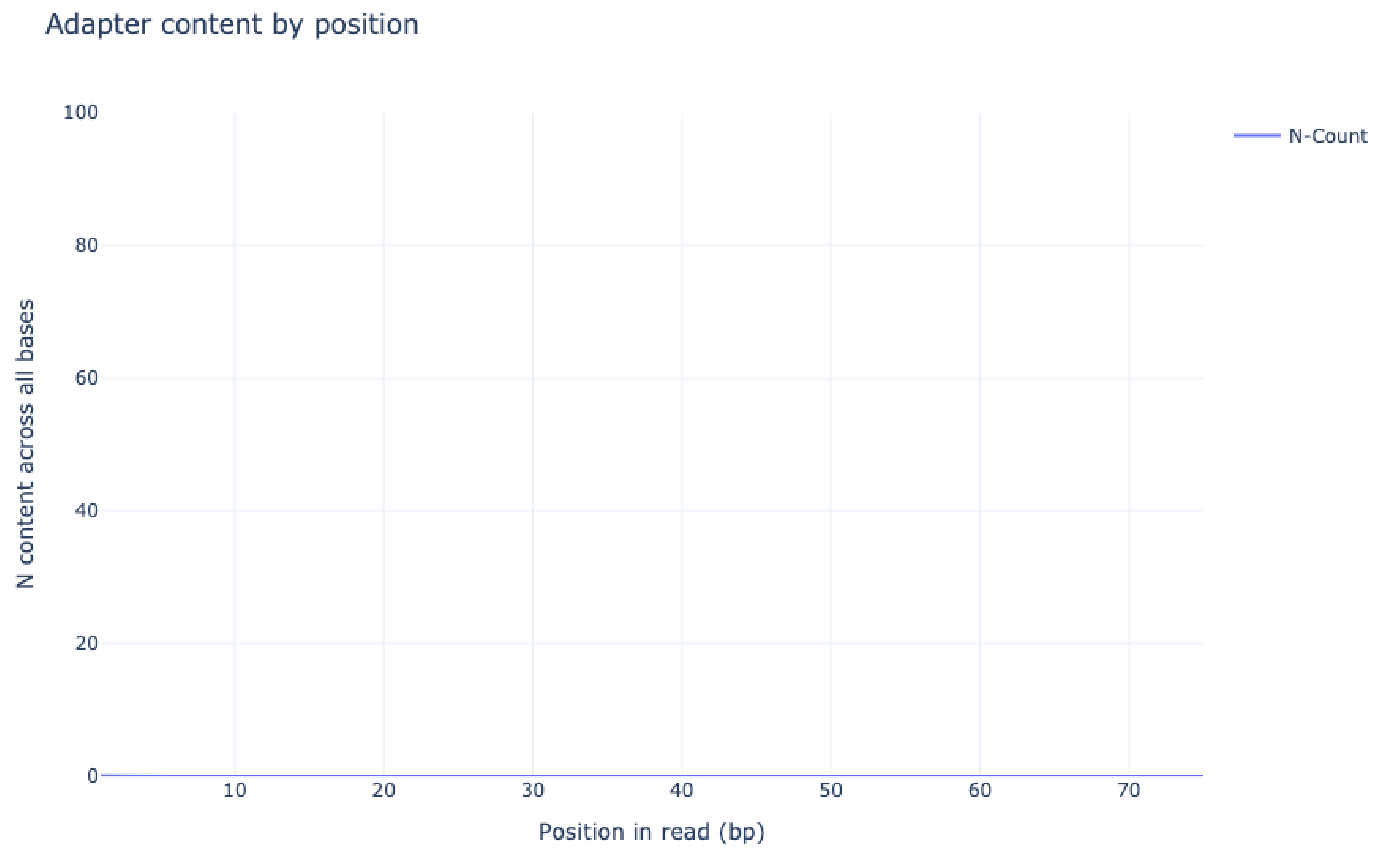
Kmer Content

Plot's kmer content by position.



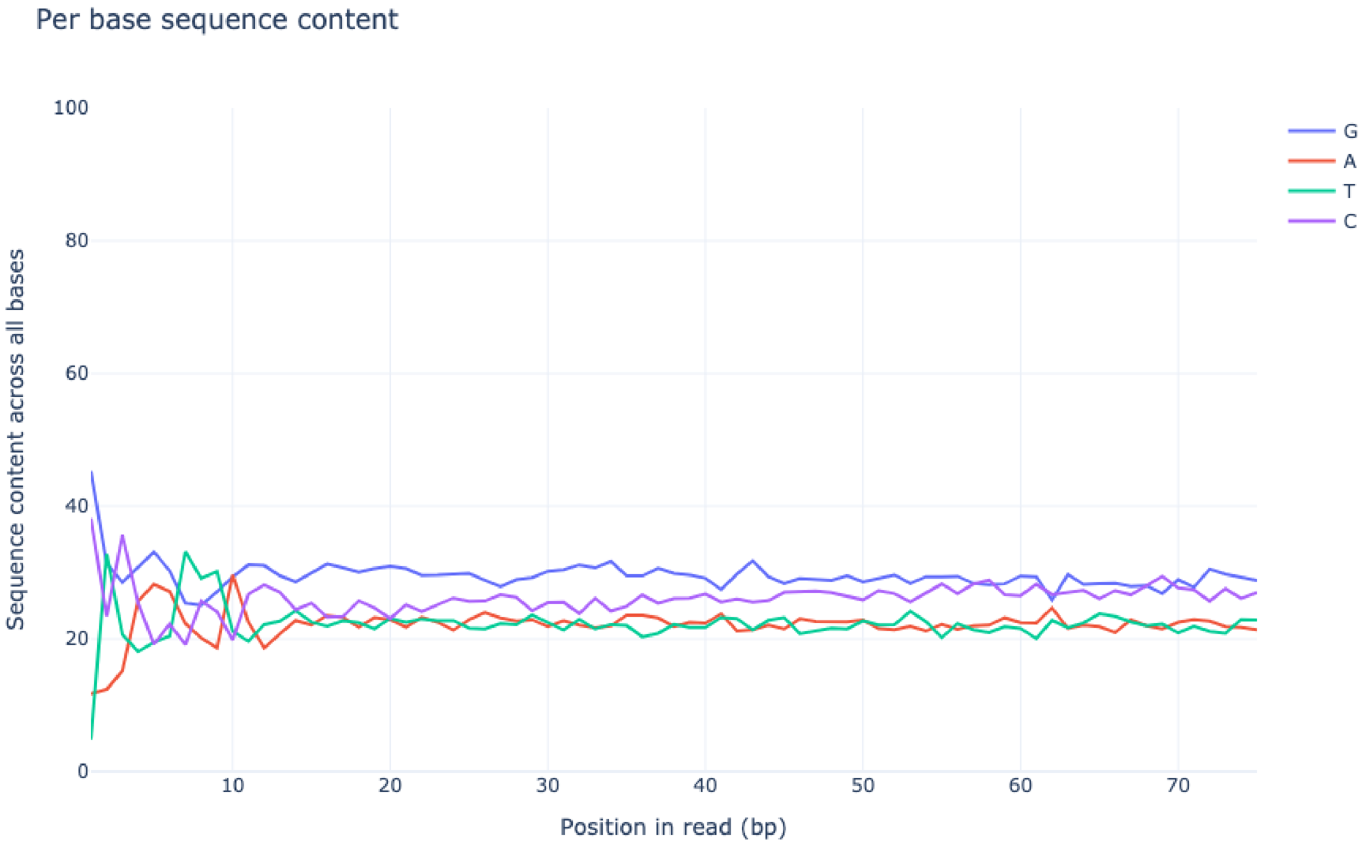
Overrepresented sequences

Plot's Per base N content.



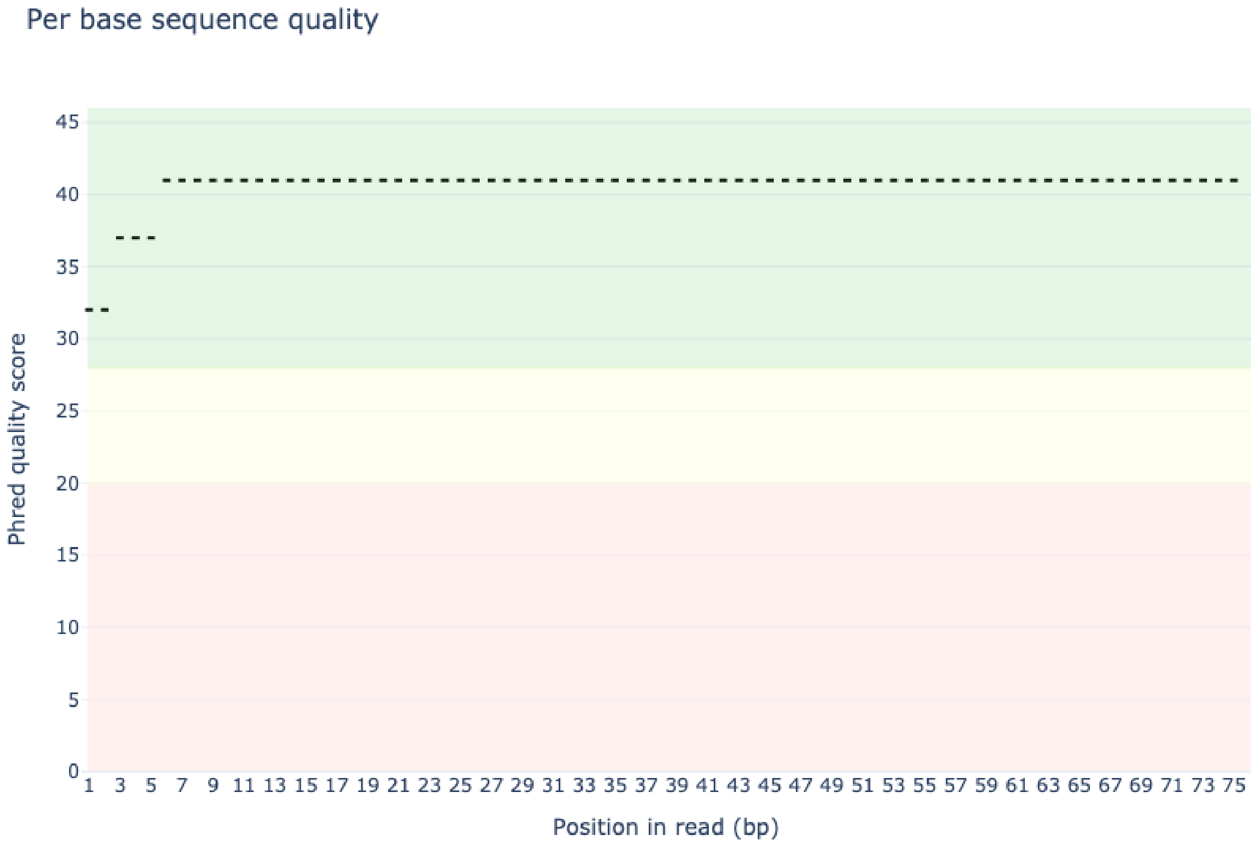
Per base sequence content

Plot's Per base sequence content.



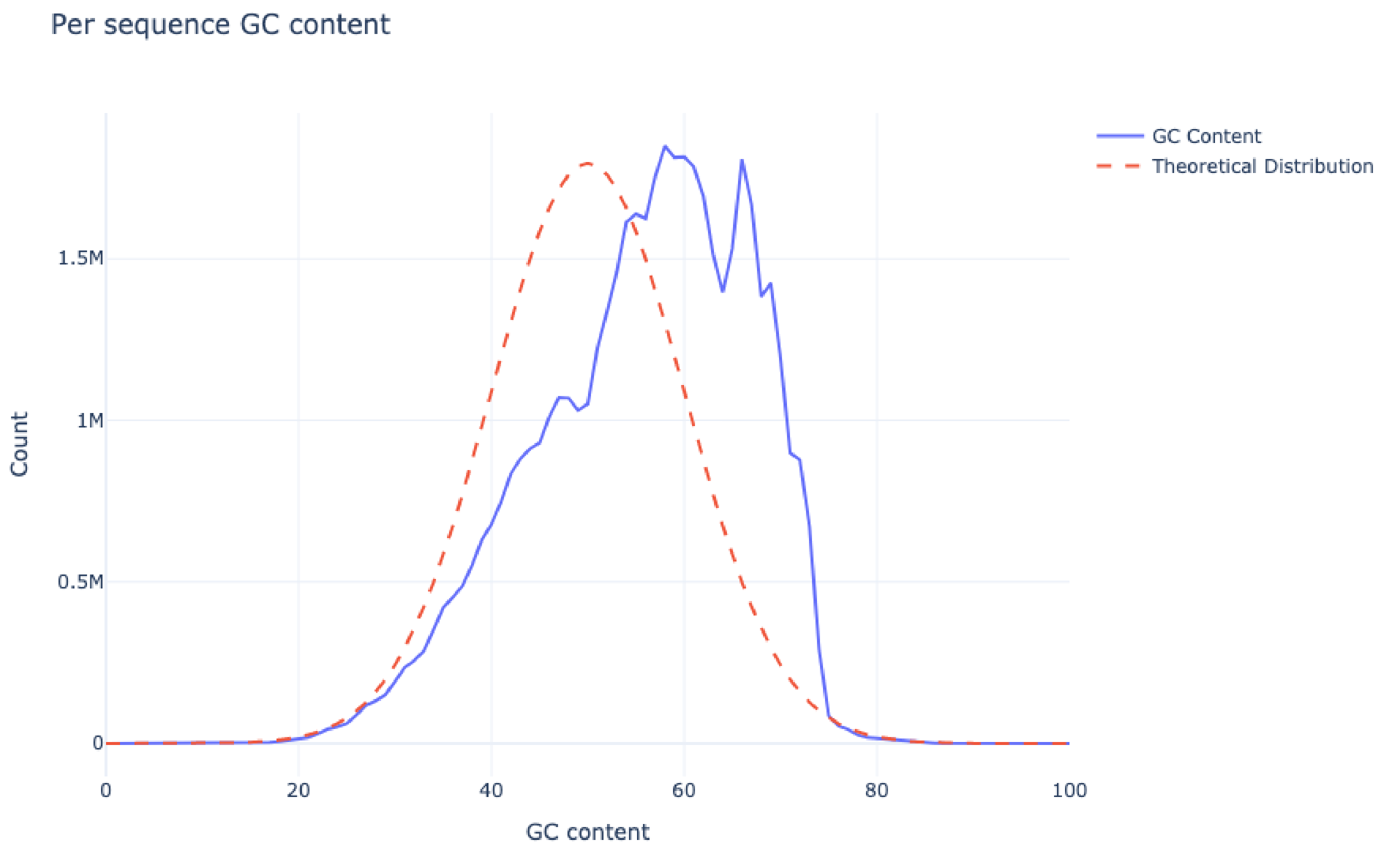
Per sequence quality scores

Plot's Per sequence quality scores.



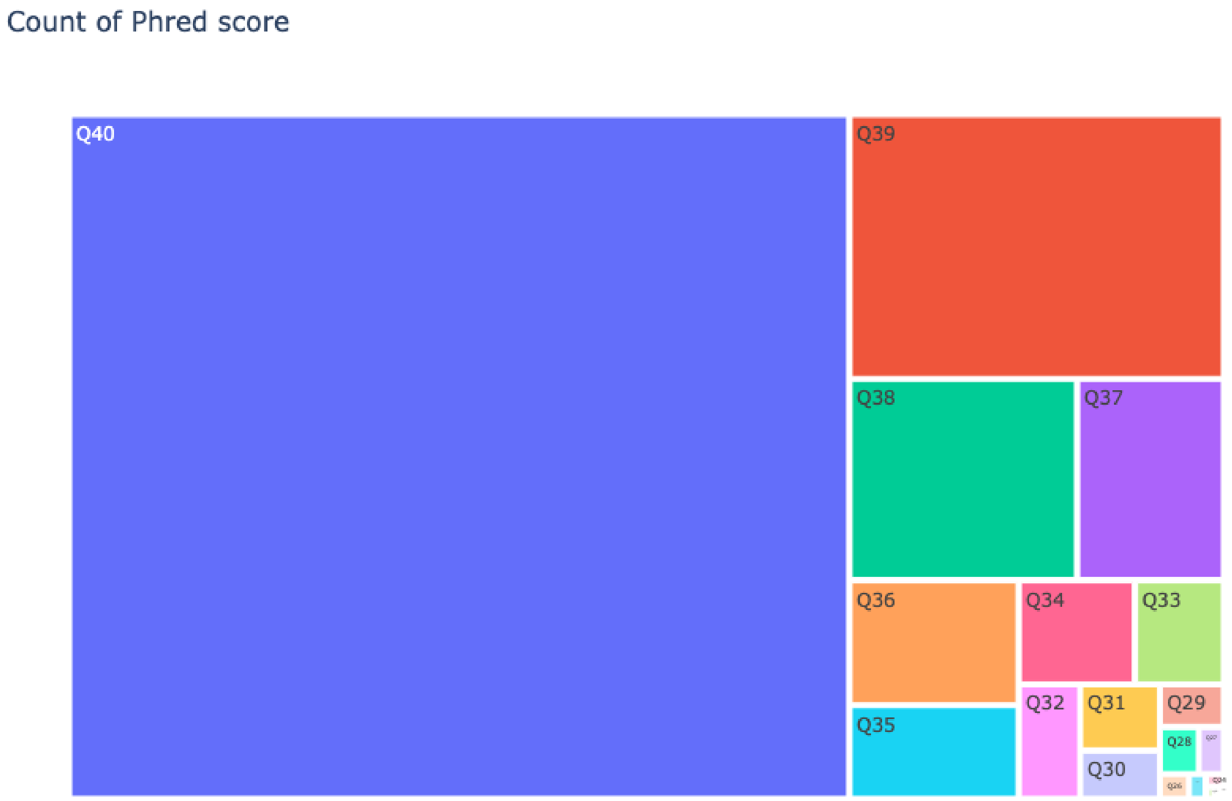
Per sequence GC content

Plot's Per sequence GC content.



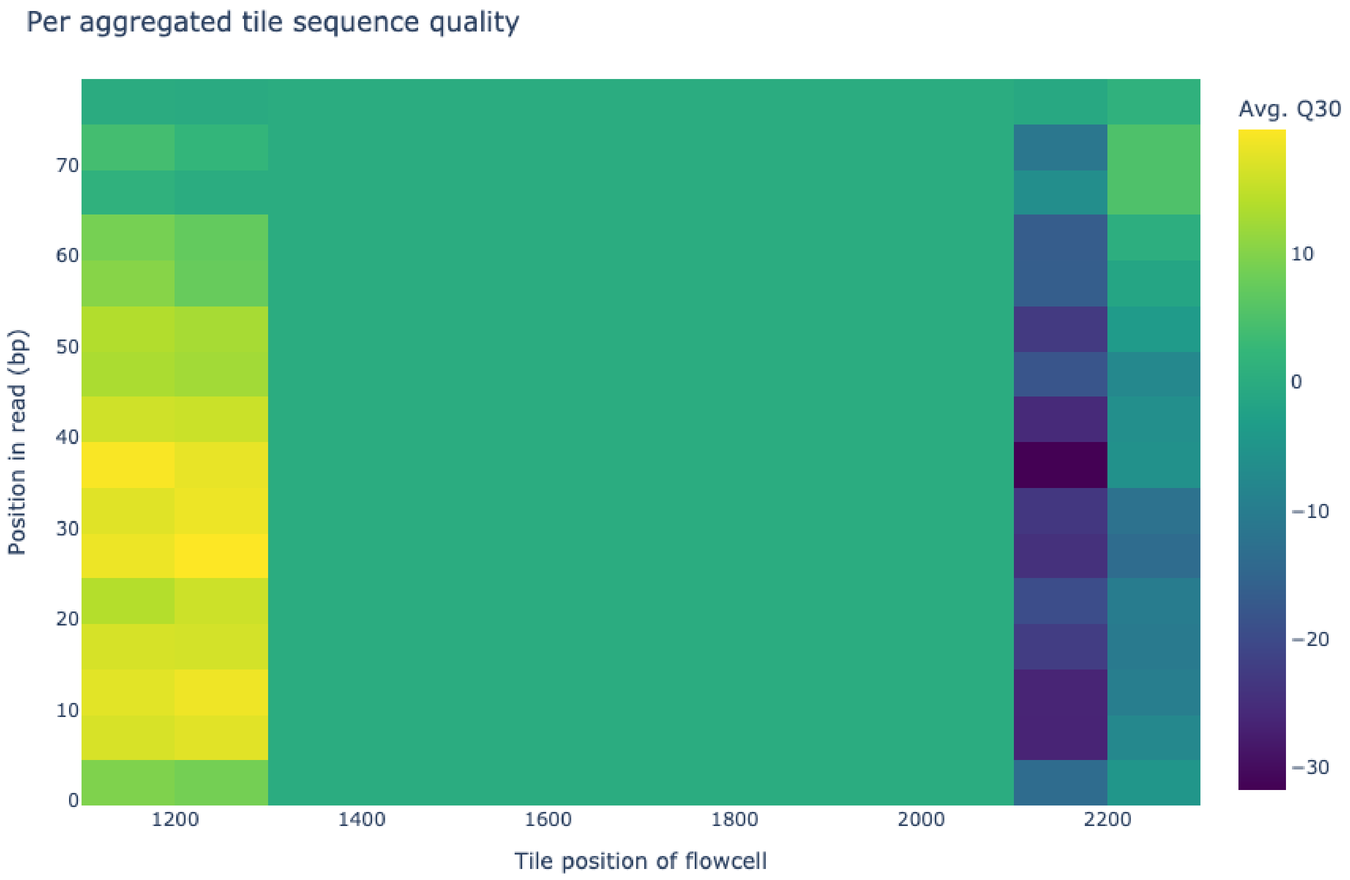
Per sequence quality scores

Plot's Per sequence quality scores.



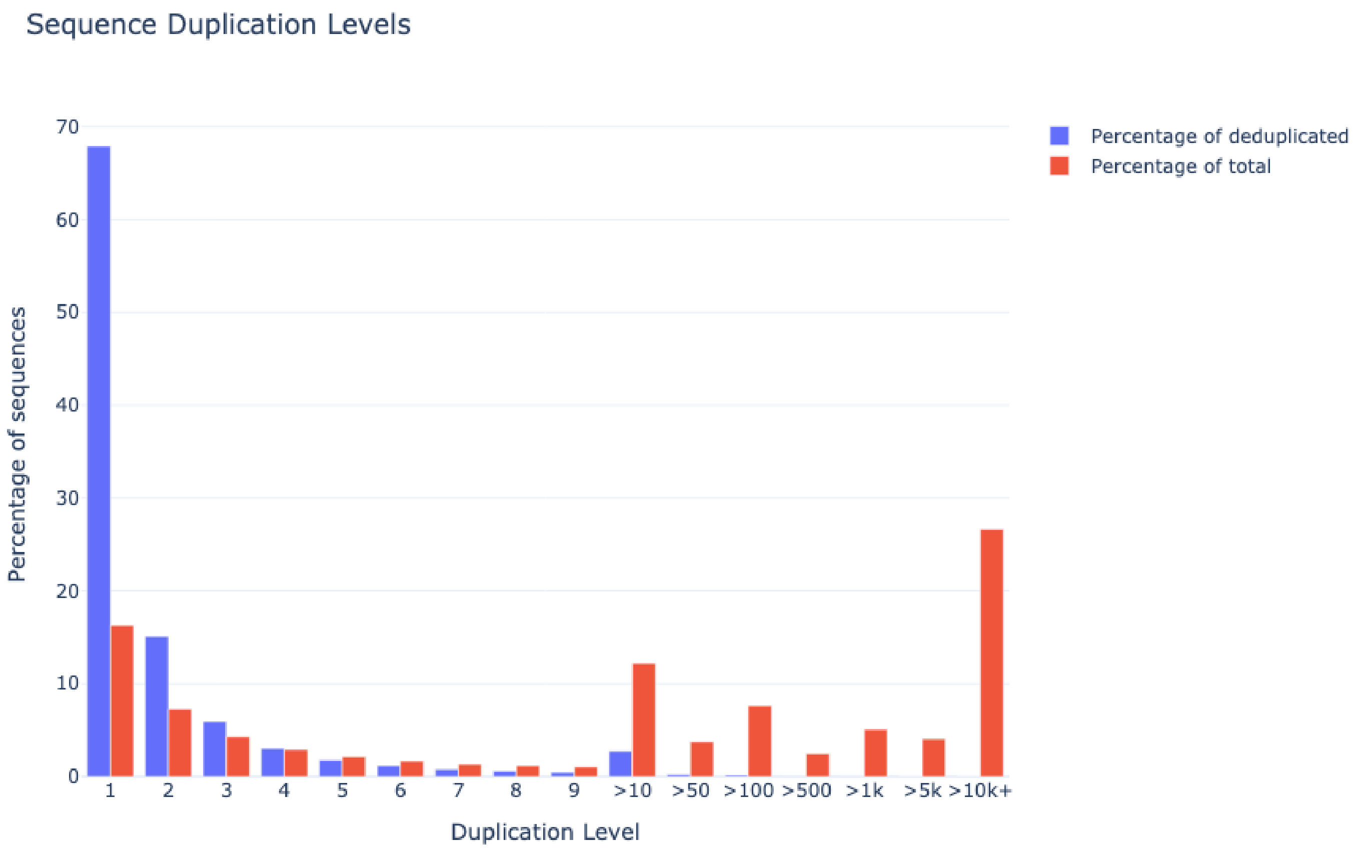
Per tile sequence quality

Plot's Per tile sequence quality.



Sequence Duplication Levels

Plot Sequence Duplication Levels.



GitHub

https://github.com/ms2206/FastQCParser.git

FastQCParser

FastQCParser User Guide:

It is recommended to open this via browser: User Guide

Introduction

A Python program to parse FastQCtext files, and generate reports and plots.

1. Clone directory: `git clone https://github.com/ms2206/FastQCParser.git`
2. Make a new python environment based from requirments.yaml `conda env create -f requirements.yaml --<NAME>`
3. Load environment env. `conda activate <NAME>`
4. Change directory into FastQCParser
5. Example Usage: `python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a`

-- plots are downloaded to the users `~/Downloads/` folder.

Set up

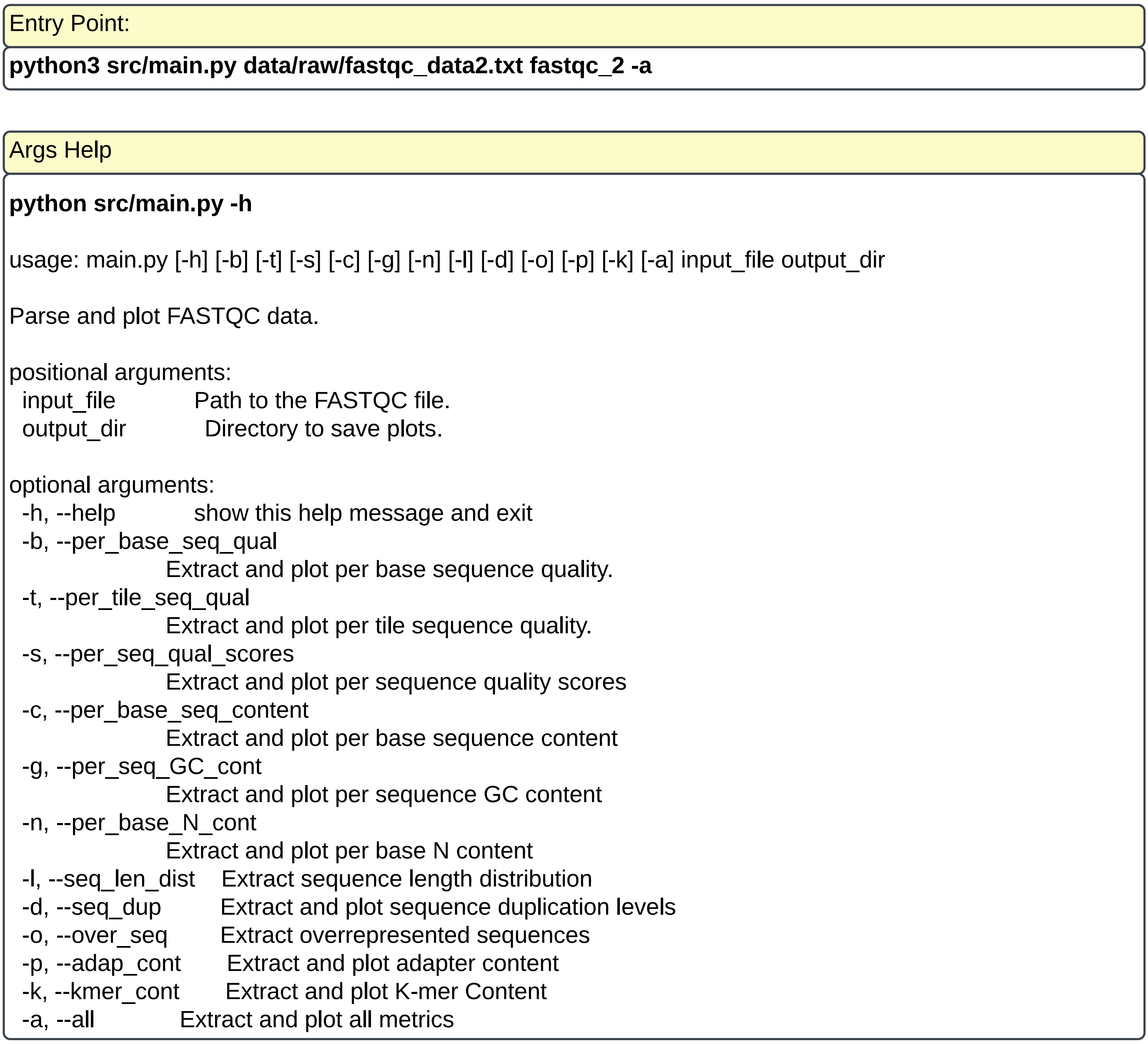
Example Usage: With python3, run executable found at `src/main.py` .

Pass `data/raw/fastqc_data2.txt` (or any fastqc file) - as input file, and `fastqc_2` as output directory (or use a customer directory name).

Use optional argument `-a` .

```
python3 src/main.py <FASTQ FILE> <DIR_NAME> [<OPTIONAL_ARGS>]
```

```
python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a
```



Entry Point:

python3 src/main.py data/raw/fastqc_data2.txt fastqc_2 -a

Args Help

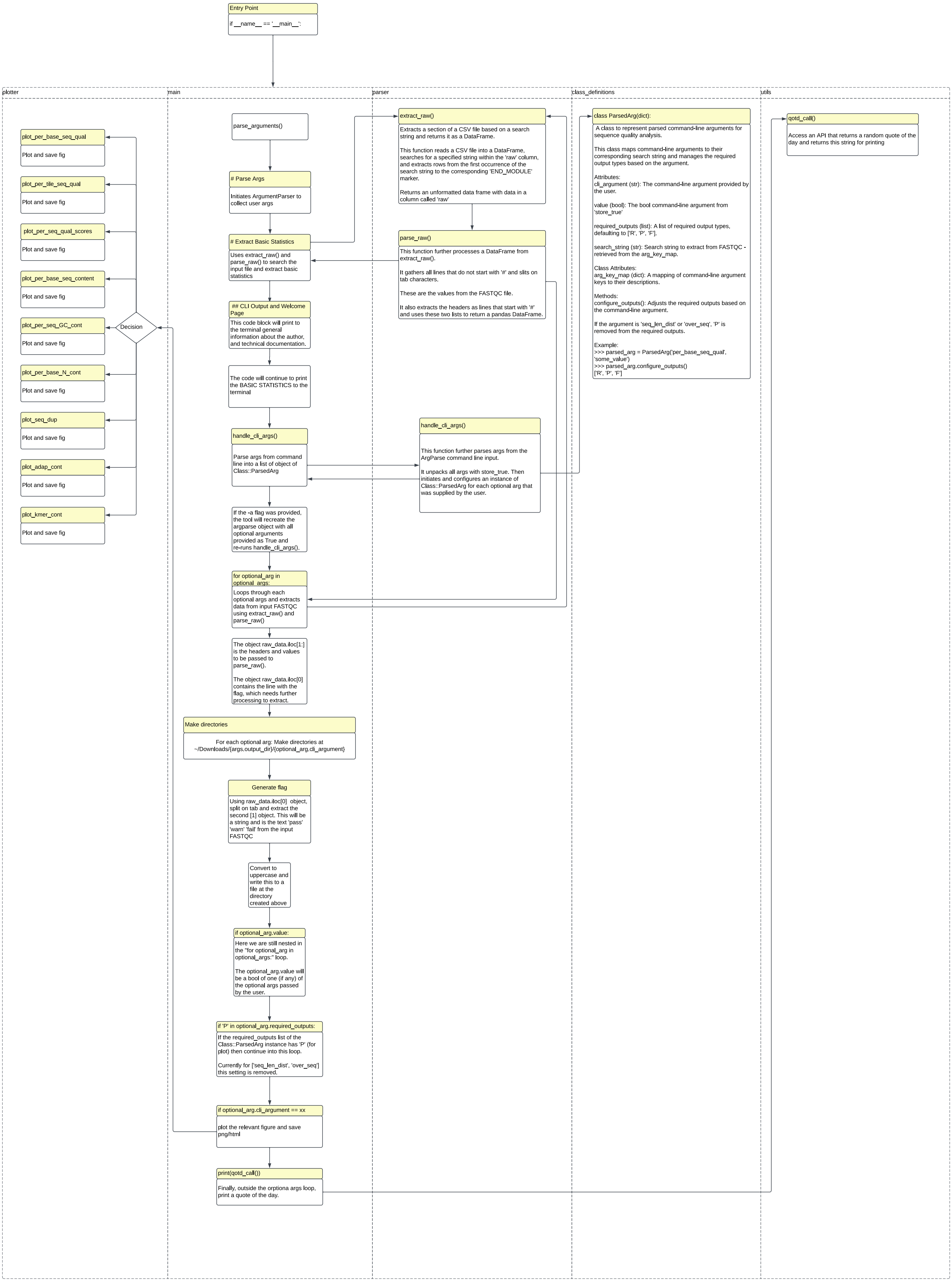
python src/main.py -h

```
usage: main.py [-h] [-b] [-t] [-s] [-c] [-g] [-n] [-l] [-d] [-o] [-p] [-k] [-a] input_file output_dir

Parse and plot FASTQC data.

positional arguments:
  input_file            Path to the FASTQC file.
  output_dir            Directory to save plots.

optional arguments:
  -h, --help            show this help message and exit
  -b, --per_base_seq_qual
                        Extract and plot per base sequence quality.
  -t, --per_tile_seq_qual
                        Extract and plot per tile sequence quality.
  -s, --per_seq_qual_scores
                        Extract and plot per sequence quality scores
  -c, --per_base_seq_content
                        Extract and plot per base sequence content
  -g, --per_seq_GC_cont
                        Extract and plot per sequence GC content
  -n, --per_base_N_cont
                        Extract and plot per base N content
  -l, --seq_len_dist    Extract sequence length distribution
  -d, --seq_dup         Extract and plot sequence duplication levels
  -o, --over_seq        Extract overrepresented sequences
  -p, --adap_cont       Extract and plot adapter content
  -k, --kmer_cont       Extract and plot K-mer Content
  -a, --all             Extract and plot all metrics
```

Optional Args

Help and misc information provided by ArgeParse for optional arguments.

```
python3 src/main.py -h
```



Entry Point

if __name__ == '__main__':

plotter

main

parser

class_definitions

utils

parse_arguments()

Parse Args

Initiates ArgumentParser to collect user args

Extract Basic Statistics

Uses extract_raw() and parse_raw() to search the input file and extract basic statistics

CLI Output and Welcome Page

This code block will print to the terminal general information about the author, and technical documentation.

The code will continue to print the BASIC STATISTICS to the terminal

handle_cli_args()

Parse args from command line into a list of object of Class::ParsedArg

If the -a flag was provided, the tool will recreate the argparse object with all optional arguments provided as True and re-runs handle_cli_args().

for optional_arg in optional_args:

Loops through each optional args and extracts data from input FASTQC using extract_raw() and parse_raw()

The object raw_data.iloc[1:] is the headers and values to be passed to parse_raw().

The object raw_data.iloc[0] contains the line with the flag, which needs further processing to extract.

Make directories

For each optional arg: Make directories at ~/Downloads/{args.output_dir}/{optional_arg.cli_argument}

Generate flag

Using raw_data.iloc[0] object, split on tab and extract the second [1] object. This will be a string and is the text 'pass' 'warn' 'fail' from the input FASTQC

Convert to uppercase and write this to a file at the directory created above

if optional_arg.value:

Here we are still nested in the "for optional_arg in optional_args:" loop.

The optional_arg.value will be a bool of one (if any) of the optional args passed by the user.

if 'P' in optional_arg.required_outputs:

If the required_outputs list of the Class::ParsedArg instance has 'P' (for plot) then continue into this loop.

Currently for ['seq_len_dist', 'over_seq'] this setting is removed.

if optional_arg.cli_argument == xx

plot the relevant figure and save png/html

print(qotd_call())

Finally, outside the orptiona args loop, print a quote of the day.

Decision

plot_per_base_seq_qual

Plot and save fig

plot_per_tile_seq_qual

Plot and save fig

plot_per_seq_qual_scores

Plot and save fig

plot_per_base_seq_content

Plot and save fig

plot_per_seq_GC_cont

Plot and save fig

plot_per_base_N_cont

Plot and save fig

plot_seq_dup

Plot and save fig

plot_adap_cont

Plot and save fig

plot_kmer_cont

Plot and save fig

extract_raw()

Extracts a section of a CSV file based on a search string and returns it as a DataFrame.

This function reads a CSV file into a DataFrame, searches for a specified string within the 'raw' column, and extracts rows from the first occurrence of the search string to the corresponding 'END_MODULE' marker.

Returns an unformatted data frame with data in a column called 'raw'

parse_raw()

This function further processes a DataFrame from extract_raw().

It gathers all lines that do not start with '#' and slits on tab characters.

These are the values from the FASTQC file.

It also extracts the headers as lines that start with '#' and uses these two lists to return a pandas DataFrame.

handle_cli_args()

This function further parses args from the ArgParse command line input.

It unpacks all args with store_true. Then initiates and configures an instance of Class::ParsedArg for each optional arg that was supplied by the user.

class ParsedArg(dict):

A class to represent parsed command-line arguments for sequence quality analysis.

This class maps command-line arguments to their corresponding search string and manages the required output types based on the argument.

Attributes:
cli_argument (str): The command-line argument provided by the user.

value (bool): The bool command-line argument from 'store_true'

required_outputs (list): A list of required output types, defaulting to ['R', 'P', 'F'].

search_string (str): Search string to extract from FASTQC - retrieved from the arg_key_map.

Class Attributes:
arg_key_map (dict): A mapping of command-line argument keys to their descriptions.

Methods:
configure_outputs(): Adjusts the required outputs based on the command-line argument.

If the argument is 'seq_len_dist' or 'over_seq', 'P' is removed from the required outputs.

Example:

```
>>> parsed_arg = ParsedArg('per_base_seq_qual', 'some_value')
>>> parsed_arg.configure_outputs()
['R', 'P', 'F']
```

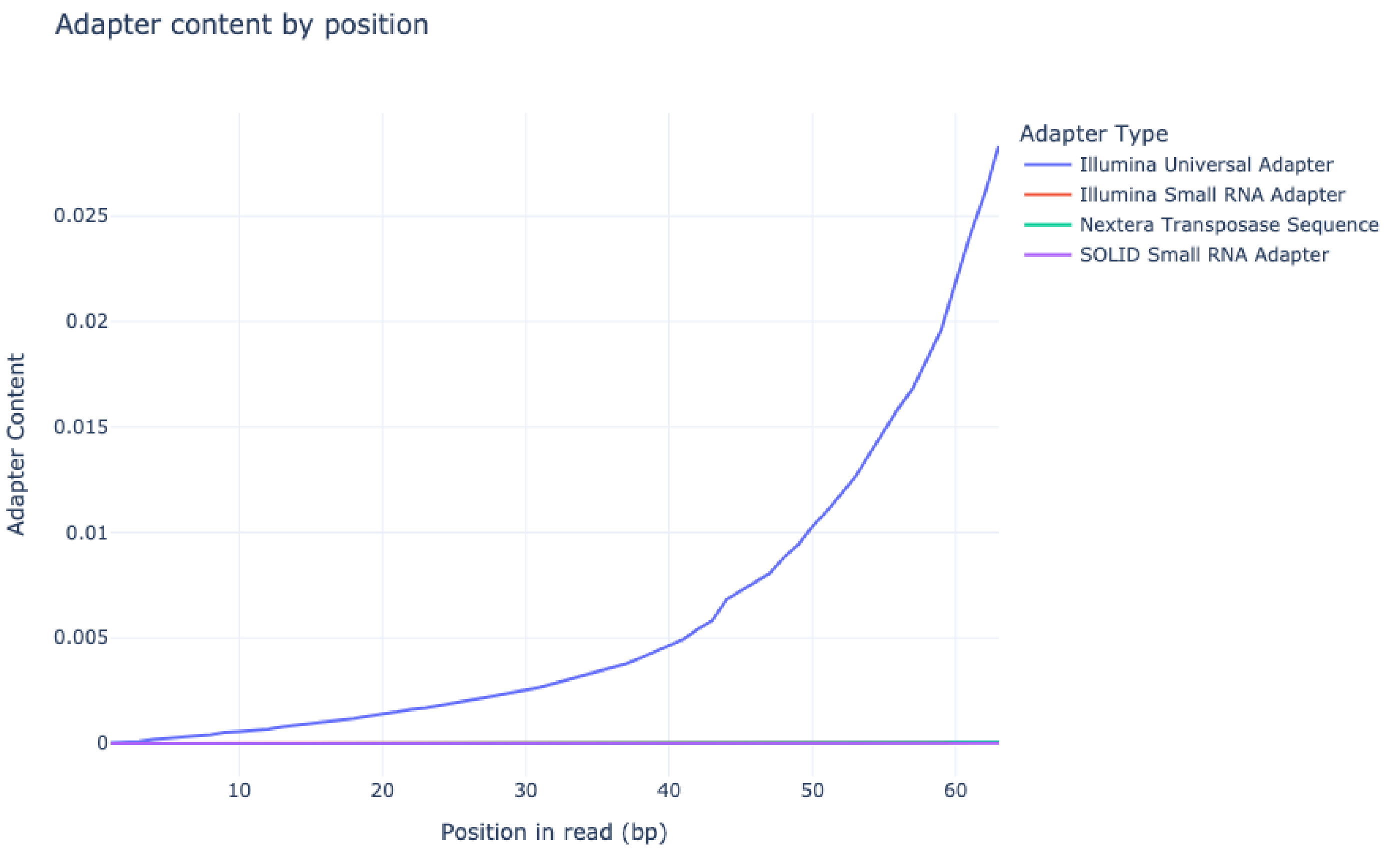
qotd_call()

Access an API that returns a random quote of the day and returns this string for printing

Example Plots

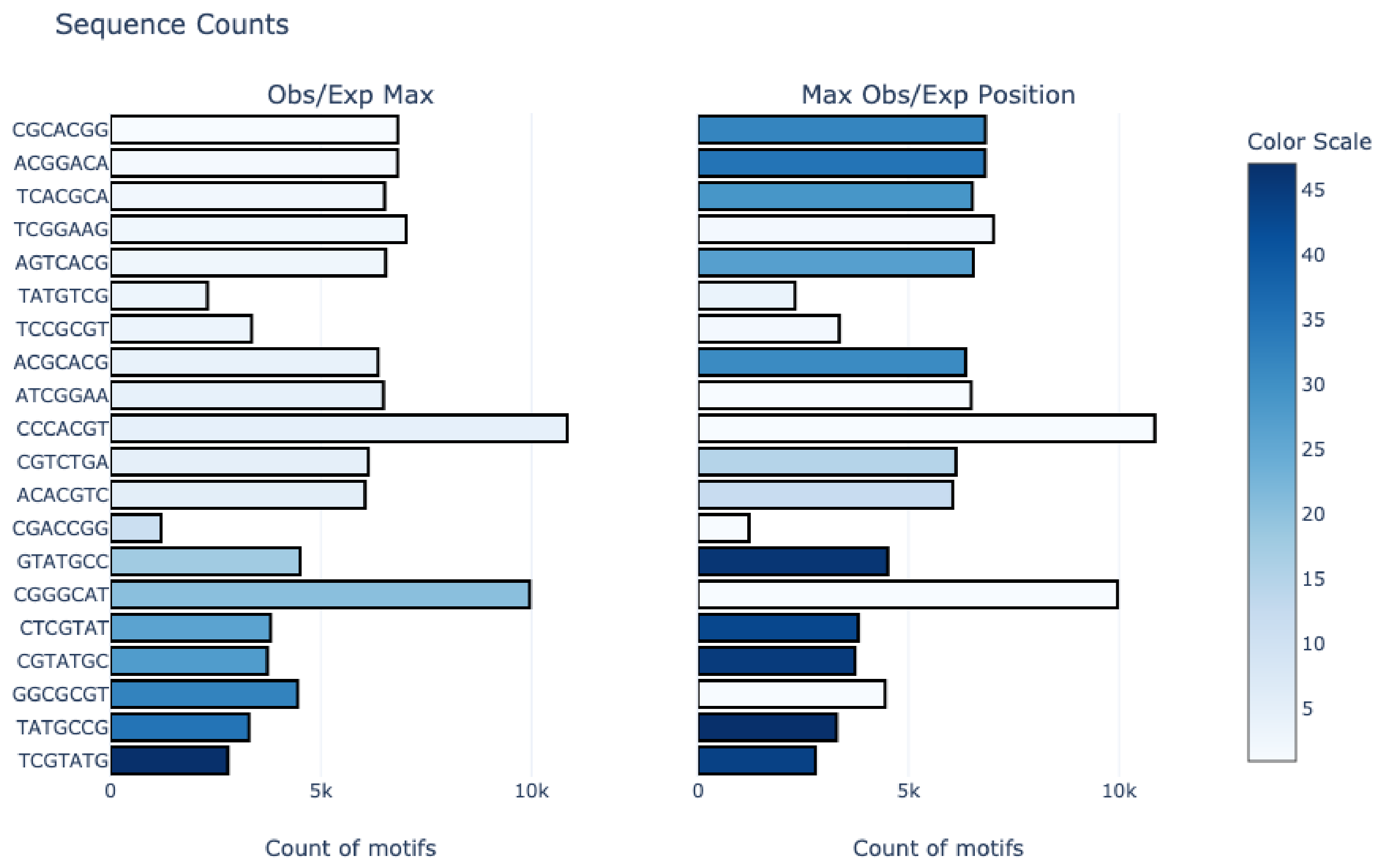
Adapter Content

Plot's adapter content by position.



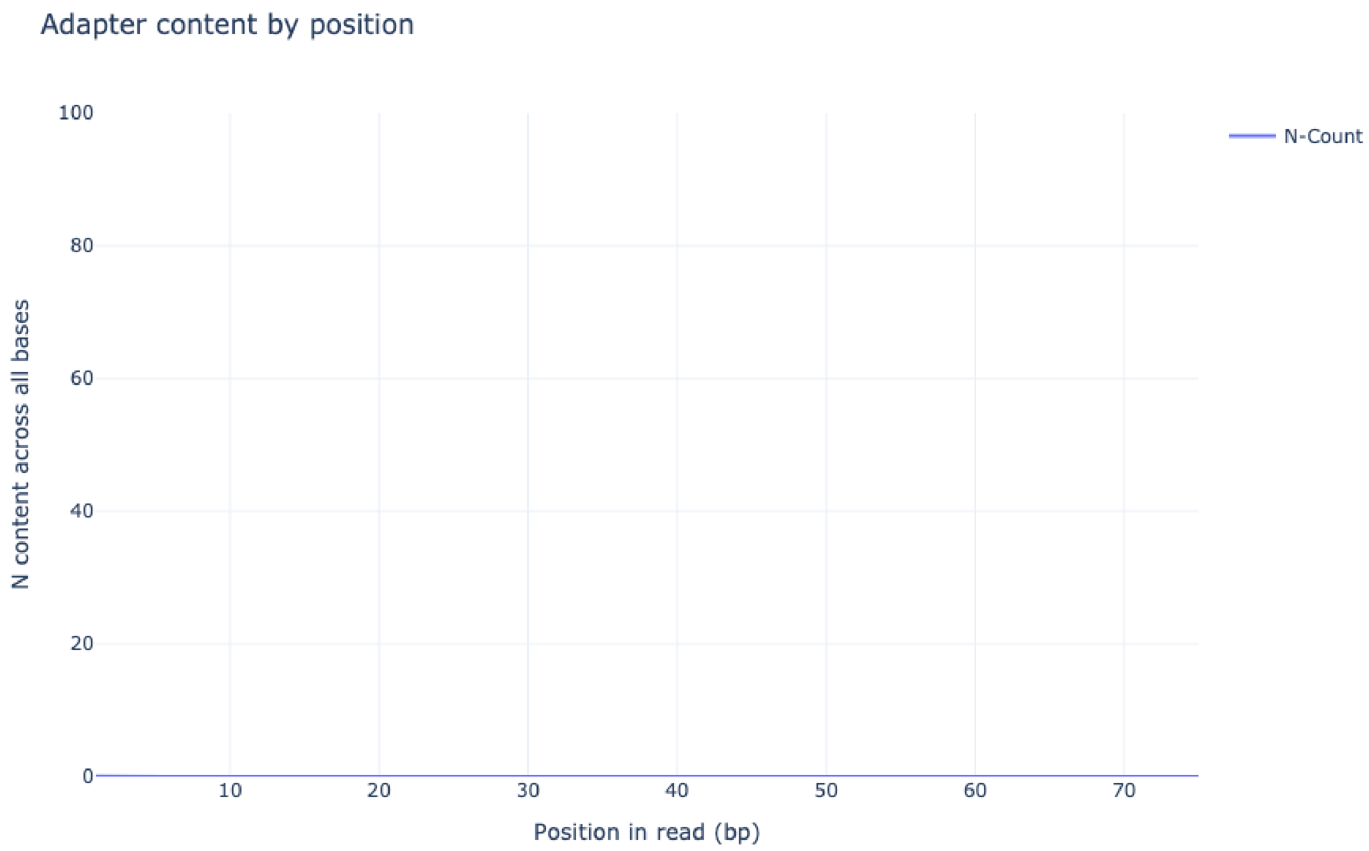
Kmer Content

Plot's kmer content by position.



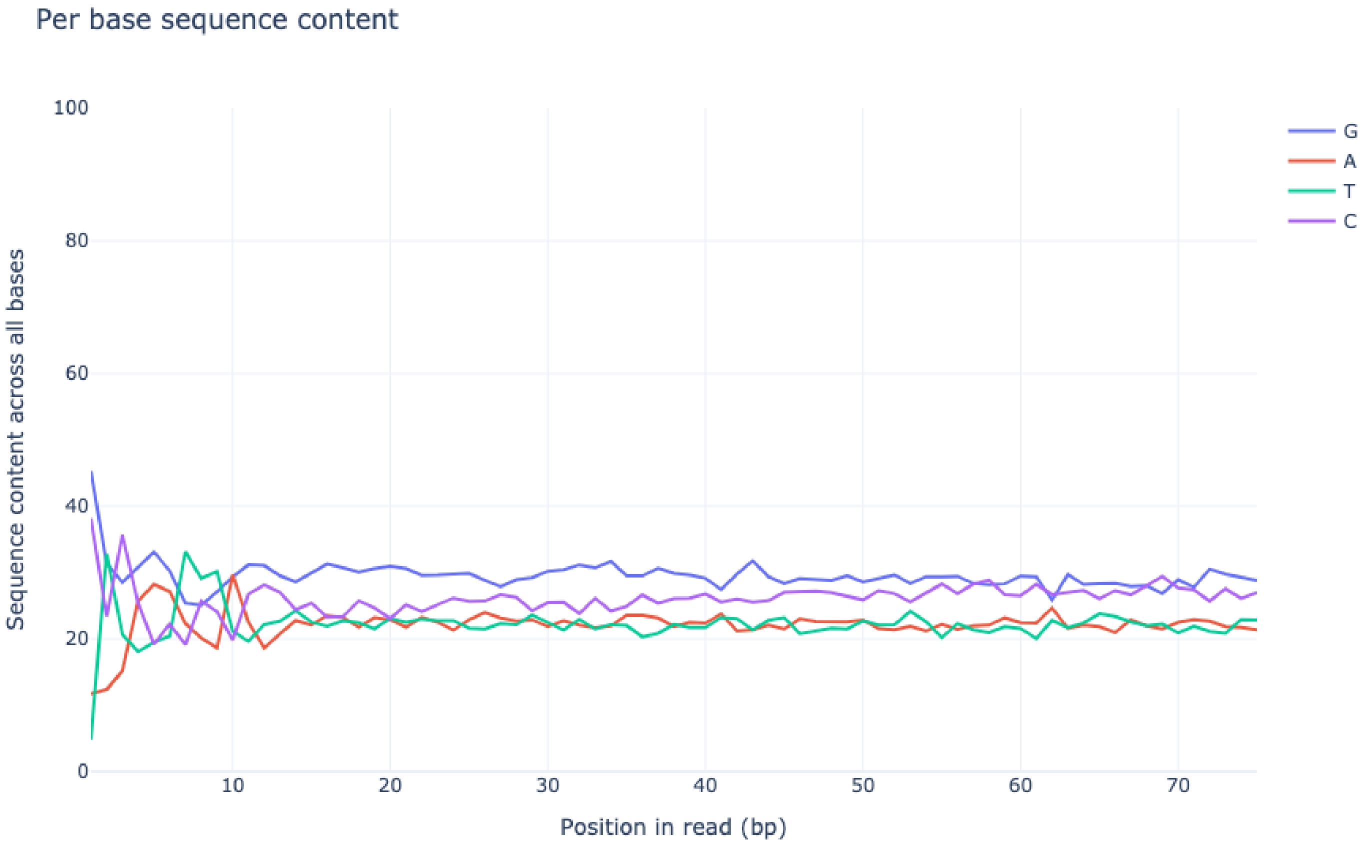
Overrepresented sequences

Plot's Per base N content.



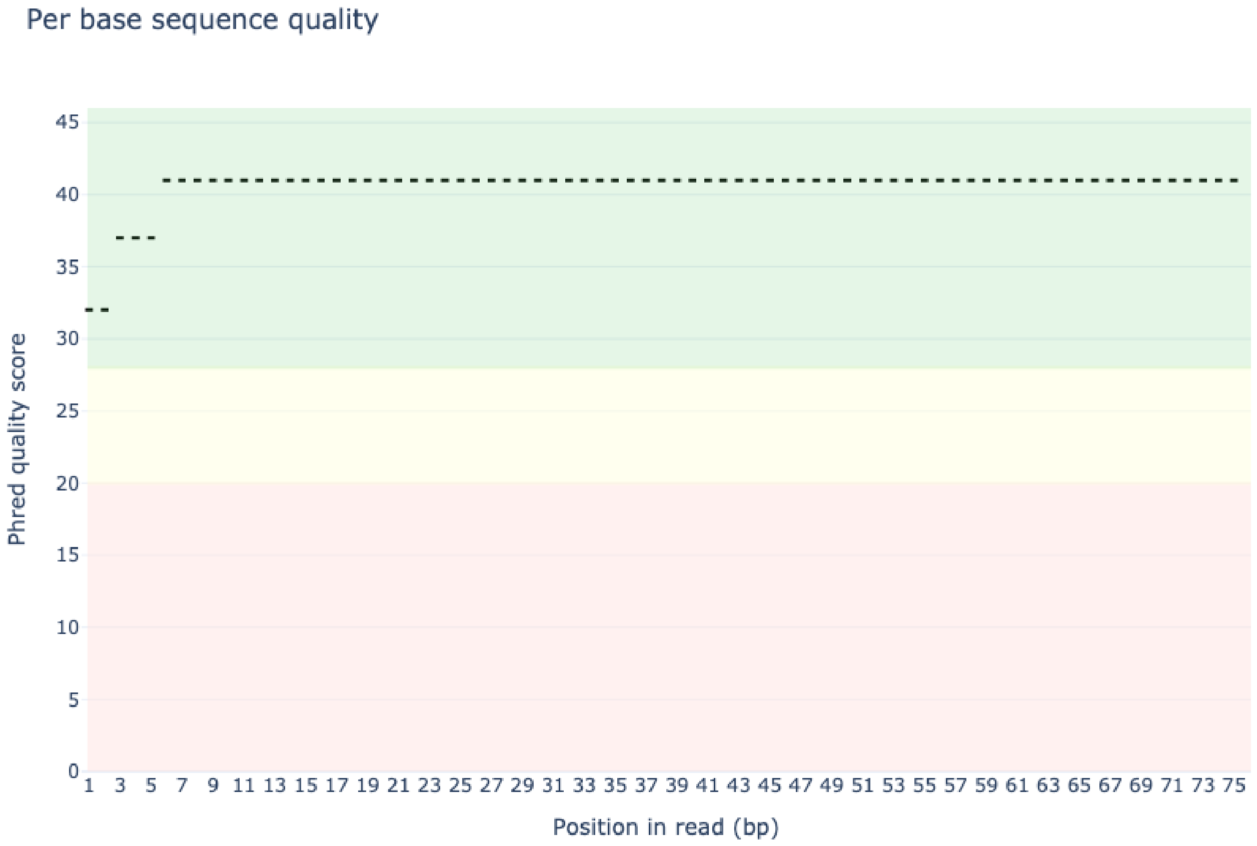
Per base sequence content

Plot's Per base sequence content.



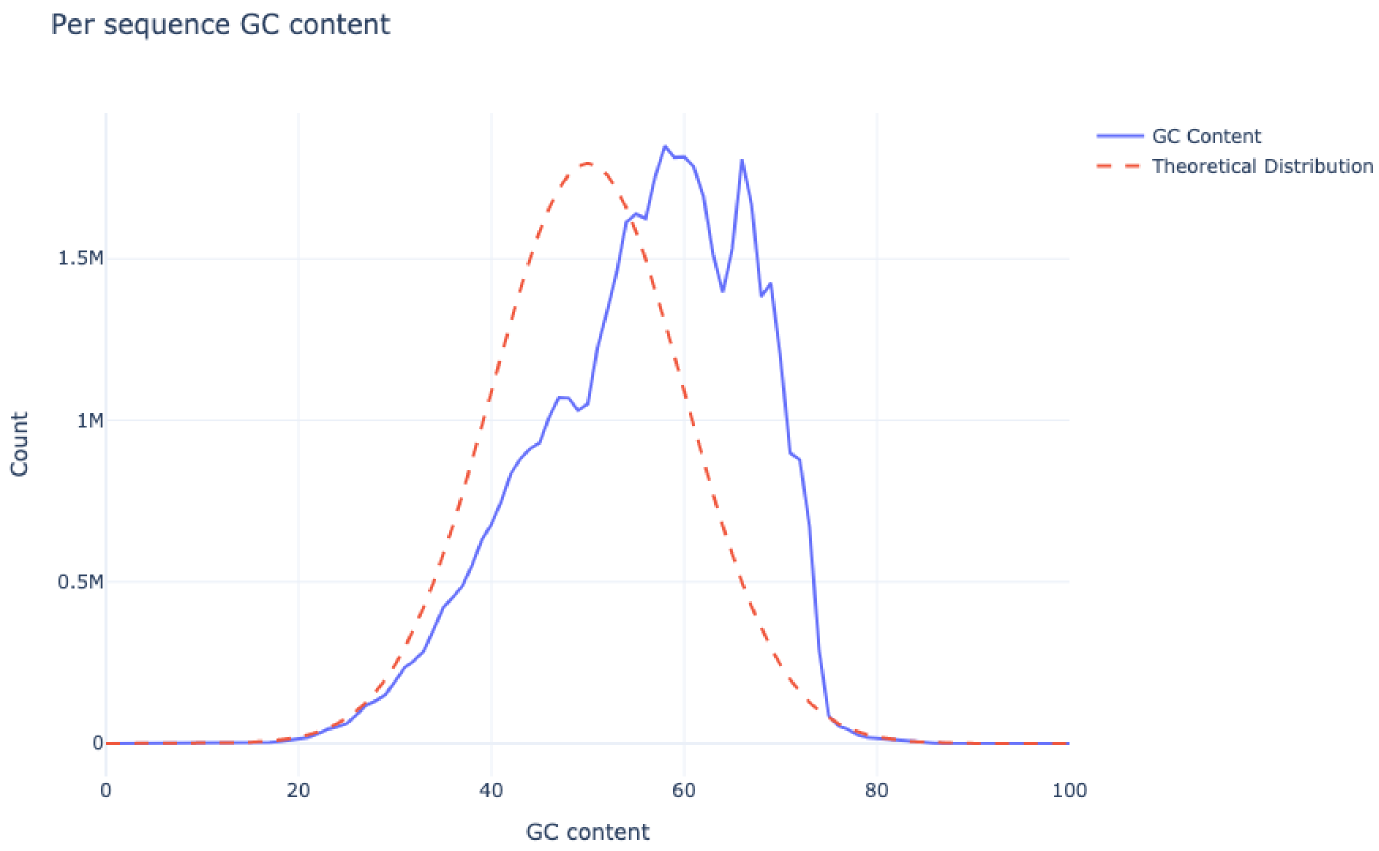
Per sequence quality scores

Plot's Per sequence quality scores.



Per sequence GC content

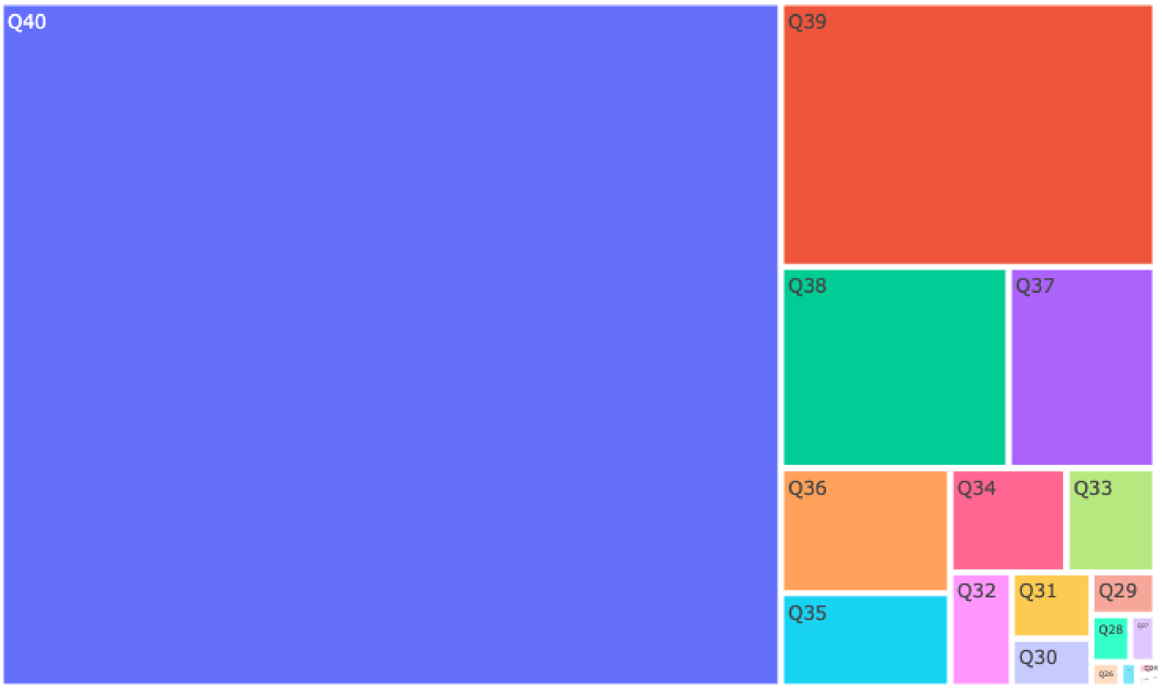
Plot's Per sequence GC content.



Per sequence quality scores

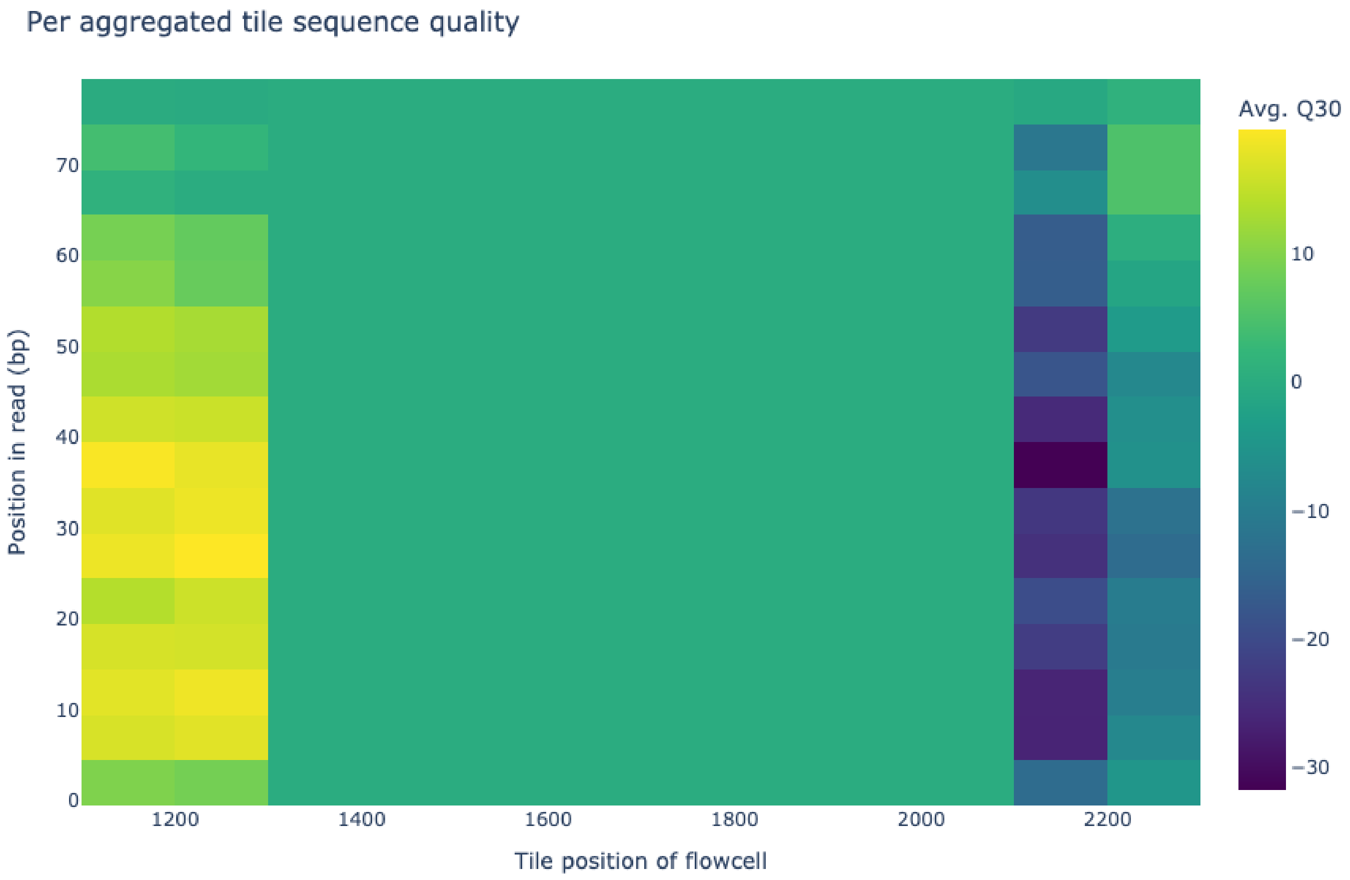
Plot's Per sequence quality scores.

Count of Phred score



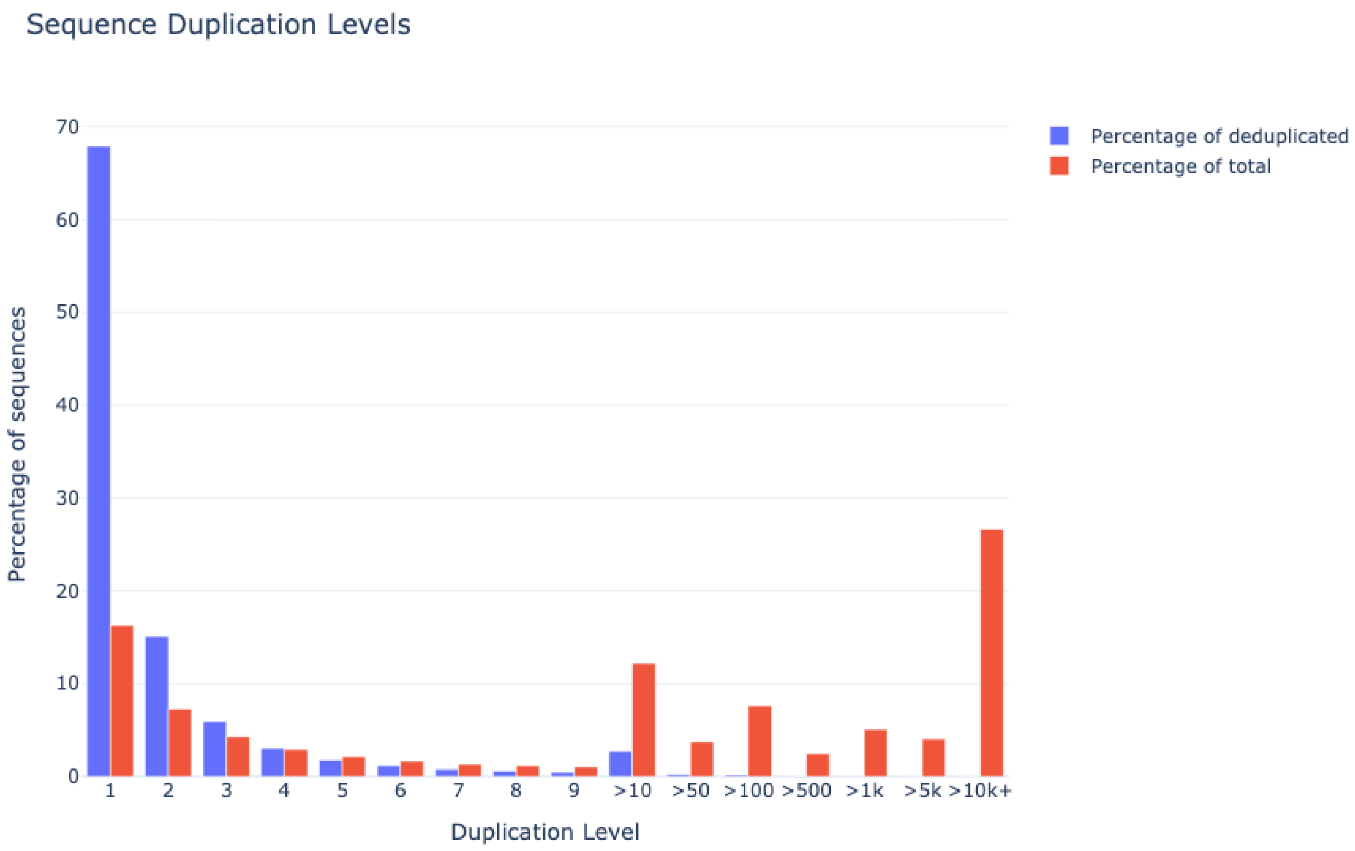
Per tile sequence quality

Plot's Per tile sequence quality.



Sequence Duplication Levels

Plot Sequence Duplication Levels.



GitHub

https://github.com/ms2206/FastQCParser.git