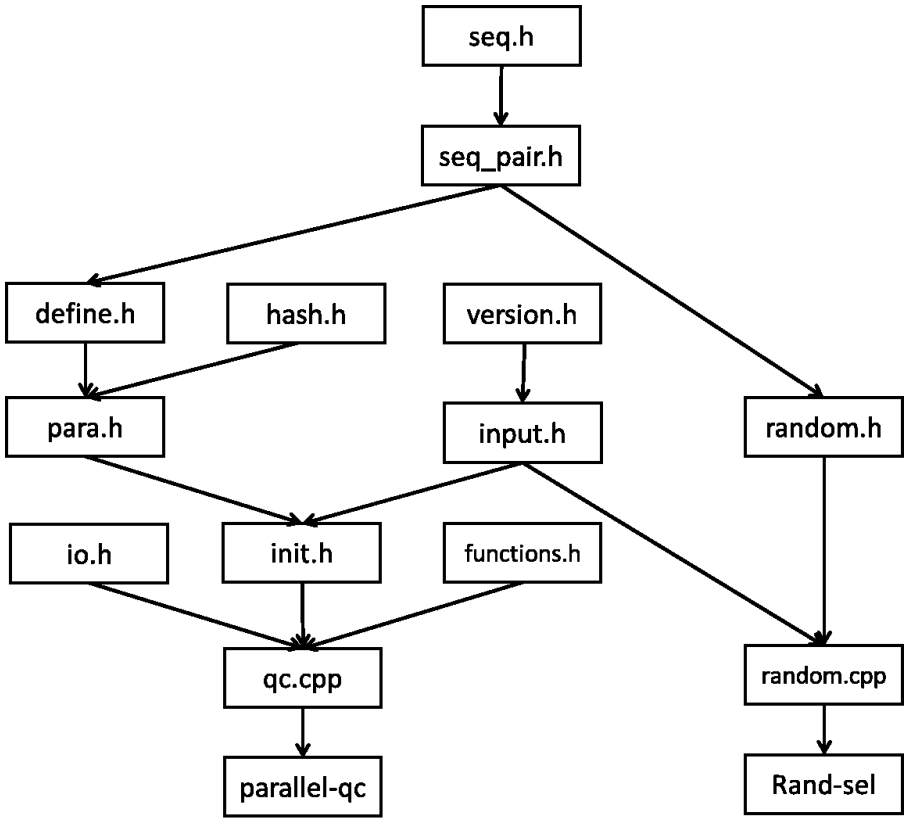
Parallel-QC development manual

Version 1.0

Parallel-QC is implemented by Linux C++ with POSIX thread. This manual is for further development of Parallel-QC or invocated into other pipeline.

1 PART I. PROJECT STRUCTURE



There are 13 source code files for the project.

seq.h: The definition of class Seq. Details can be found in Part II.

seq_pair.h: The definition of class Seq_pair. Details can be found in Part II.

define.h: The extern declaration of all global variables of parallel-qc.

hash.h: The definition of hash function.

version.h: The definition of version of parallel-qc.

para.h: The parameter parser of parallel-qc.

input.h: The sequence input functions for parallel-qc and rand-sel.

io.h: The I/O functions for parallel-qc.

init.h: The initialization functions for parallel-qc.

functions.h: The quality control functions for parallel-qc.

qc.cpp: The entrance of parallel-qc.

random.h: The random selection functions for rand-sel.

random.h: The entrance of rand-sel.

2 PART II. CLASSES

2.1 class Seq

The basic class of a sequence with quality value.

public:

Seq(string _seq, string _qual); The construction function with sequence _seq and quality values _qual.

void Trim(int _begin, int _end); To get the trimmed sub-read from _begin to _end.

void Qual_trim(int _qual, float _p); To do the quality trim with quality value _qual and proportion of _p.

*void Primer_trim(string * _primers, int n);* To do the tag-sequence trim with n tag sequences * _primers.

void GC_trim(float _p_min, float _p_max); To do the GC proportion trim with minimum GC proportion _p_min and maximum GC proportion _p_min.

bool Is_qual_trim(); If this sequence is trimmed by quality trimming.

bool Is_primer_trim(); If this sequence is trimmed by tag-sequence trimming.

bool Is_gc_trim(); If this sequence is trimmed by GC proportion trimming.

bool Is_dup_trim(); If this sequence is trimmed by duplication trimming.

bool Is_seq; If this sequence is trimmed.

2.2 class Seq_pair

The pair-ended sequences based on class Seq.

public:

Seq_pair(Seq _seq_1, Seq _seq_2); The construction function with 2 Seq objects.

void Add_seq_1(Seq _seq_1); Add sequence_1 to this sequence pair.

void Add_seq_2(Seq _seq_2); Add sequence_2 to this sequence pair.

void Add_qual_1(string qual_1); Add the quality values to sequence_1.

void Add_qual_2(string qual_2); Add the quality values to sequence_2.

*void Out_put(string _label, ostream * out_1, ostream * out_2, bool format);* Output the sequence to stream * out_1 for sequence_1 and *out_2 for sequence_2. If format= Ture the output format is in Fastq, else in Fasta.

3 PART III. VARIABLES AND FUNCTIONS

3.1 Global variables for parallel-qc

bool _in_pair; If the input is pair-ended.

bool _single_input ; If the input is in two files.

bool _keep_pair; If the output is kept in pair-ended.

bool _dup; If the duplication trimming is processed.

bool _ext_qual; If there are extern quality values.

bool _in_format ; Input format, true for Fastq, and false for Fasta.

bool _out_format = true; Output format, true for Fastq, and false for Fasta.

int _thread ; Thread number.

vector <string> order; Vector to keep the sequence order by recording the label.

hash_map <string, Seq_pair, std_string_hash> seq_pairs; Hash_map to store all sequences indexed by the sequence label.

3.2 Functions for quality control

void Input_reads_pair(); Input the raw reads of pair-ended format.

*void Input_reads();*Input the raw reads of single-ended format.

void Input_ext_qual_pair(); Input the quality values of pair-ended format for Fasta.

*void void Input_ext_qual();*Input the quality values of single-ended format for Fasta.

void Output_analysis_report(); Output the analysis report

void Output_reads_pair(); Output the analysis results in pair-ended format.

void Output_reads(); Output the analysis results in single-ended format.

void Para_args(); Parse the arguments.

void Init(); Initialization.

void Trim(); Trimming processes.

void Drop_dup(); Drop the duplication reads.

4 PARALLEL COMPUTING

Parallel-QC parallelizes the trimming processes and distribute into different threads based on POSIX thread technology.

*void * Trim_parallel(void * args);* The function pointer for multi-thread trimming processes.

*void * Drop_dup_1(void *);* The function pointer for parallel duplication trimming of sequence_1 for pair-ended data.

*void * Drop_dup_2(void *);* The function pointer for parallel duplication trimming of sequence_2 for pair-ended data.

5 CONTACT US

Any problem please contact

Xiaoquan Su

suxq@qibebt.ac.cn