PanGraphRNA Command-line User Manual

(version 1.0)

- PanGraphRNA is an efficient, flexible and web-based Galaxy platform that can be easily used to construct graph pangenomes from genetic variations at individual, subpopulation, and population levels. It can assist researchers to select appropriate graph pangenomes using various performance metrics for both real and simulation experiments.
- Currently, PanGraphRNA is composed of four functional modules: **Graph Pangenome Preparation Module, Construction Module, Evaluation Module, and Application Moudule**.
- PanGraphRNA was powered with an advanced packaging technology, which enables compatibility and portability.
- PanGraphRNA project is hosted on https://github.com/cma2015/PanGraphRNA
- PanGraphRNA docker image is available at https://hub.docker.com/r/malab/pangraphrna
- PanGraphRNA command-line version is available at https://github.com/cma2015/PanGraphRNA/command-line

Getting Started

Before using PanGraphRNA, ensure you have Python 3.9 or higher installed. Run commands in your terminal (Command Prompt on Windows, Terminal on Mac/Linux) using the format:

```
python pangraphrna.py <command> [options]
```

Commands

1. `sra2fastq`

What it does: Converts SRA (Sequence Read Archive) files to FASTQ format using `fastq-dump` (from SRA Toolkit).

Required Options:

- `--software_path`: Path to the folder containing `fastq-dump` (e.g., where SRA Toolkit is installed).
- `--input`: Folder containing your SRA files (`.sra` extension).
- `--output`: Folder where you want the output FASTQ files saved.

Optional Option:

- `--minread`: Minimum read length to keep (default: 20).

Example:

```
python pangraphrna.py sra2fastq \
        --software_path /path/to/sratoolkit/bin \
        --input /my_sra_files \
        --output /my_fastq_files \
        --minread 30
```

- Replace `/path/to/sratoolkit/bin` with where `fastq-dump` is installed.
- Replace `/my_sra_files` with the folder holding your `.sra` files.
- Replace `/my_fastq_files` with the folder where you want FASTQ files.

2. `quality_control`

What it does: Improves read quality using `fastp` (trims low-quality reads, removes adapters).

Required Options:

- `--software_path`: Path to the folder containing `fastp`.
- `--fastq_type`: Type of FASTQ files: `single` (single-end) or `paired` (paired-end).
- For single-end: `--single_file`: Path to your single FASTQ file.
- For paired-end: `--paired_file1` and `--paired_file2`: Paths to the two paired FASTQ files.
- `--output`: Folder for cleaned output files.
- `--name`: Name for your output files (e.g., "sample1").

Optional Options:

- `--minread`: Minimum read length to keep (default: 20).
- `--quality_value`: Minimum quality score for bases (default: 15).
- `--single_adapter` (single-end) or `--paired_adapter1` / `--paired_adapter2` (paired-end): Adapter sequences to remove (if known).

Examples:

- Single-end reads:

```
python pangraphrna.py quality_control \
        --software_path /path/to/fastp \
        --fastq_type single \
        --single_file /my_fastq/sample1.fastq \
        --output /my_clean_fastq \
        --name sample1 \
        --quality_value 20
```

- Paired-end reads (with auto-detected adapters):

```
python pangraphrna.py quality_control \
        --software_path /path/to/fastp \
        --fastq_type paired \
        --paired_file1 /my_fastq/sample1_R1.fastq \
        --paired_file2 /my_fastq/sample1_R2.fastq \
        --output /my_clean_fastq \
        --name sample1
```

3. `graph_pangenome_constructor`

What it does: Builds a graph-based pangenome using `HISAT2` (integrates reference genome and genetic variations).

Required Options:

- `--bcf_path`: Path to `bcftools` (for processing VCF files).
- `--hisat2_path`: Path to `HISAT2` (alignment tool).
- `--r_path`: Path to R (statistical software).
- `--fasta`: Path to the reference genome (FASTA format).
- `--vcf`: Path to the VCF file with genetic variations.
- `--method`: Method for building the pangenome (check tool docs for options).
- `--output`: Folder for the pangenome output.
- `--name`: Name for your pangenome.
- `--threads`: Number of CPU cores to use (e.g., 4).

Optional Options:

- `--gtf`: Path to a GTF file (gene annotations, if available).
- `--accession_list` or `--accession`: List of sample accessions (if needed).

Example:

```
python pangraphrna.py graph_pangenome_constructor \
        --bcf_path /path/to/bcftools \
        --hisat2_path /path/to/hisat2 \
        --r_path /path/to/R \
        --fasta /ref_genome/human.fasta \
        --vcf /variants/all_variants.vcf \
        --method basic \
        --output /my_pangenome \
        --name human_pangenome \
        --threads 8
```

4. `align`

What it does: Aligns FASTQ reads to a reference genome or pangenome using `HISAT2`.

Required Options:

- `--hisat2_path`: Path to `HISAT2`.
- `--perl_path`: Path to Perl (required by `HISAT2`).
- `--index_path`: Path to the pre-built `HISAT2` index (of your genome/pangenome).
- `--fastq_type`: `single` or `paired` (matching your FASTQ files).
- For single-end: `--single_file`: Path to cleaned FASTQ.
- For paired-end: `--paired_file1` and `--paired_file2`: Paths to cleaned paired FASTQs.
- `--strand_info`: Strand-specific info: `F`/`R` (single-end) or `FR`/`RF` (paired-end) (check your sequencing protocol).
- `--output`: Folder for alignment output (SAM/BAM files).
- `--name`: Name for your alignment files.
- `--threads`: Number of CPU cores (e.g., 4).

Optional Options:

- `--min_intron_length`/`--max_intron_length`: Minimum/maximum intron length (default: tool defaults).
- `--mismatch`: Maximum allowed mismatches (default: tool default).
- `--unique`: `yes` to keep only uniquely mapped reads (requires `--samtools_path`).

Example (paired-end):

```
python pangraphrna.py align \
        --hisat2_path /path/to/hisat2 \
        --perl_path /path/to/perl \
        --index_path /my_pangenome/index \
        --fastq_type paired \
        --paired_file1 /my_clean_fastq/sample1_clean_R1.fastq \
        --paired_file2 /my_clean_fastq/sample1_clean_R2.fastq \
        --strand_info FR \
        --output /my_alignments \
        --name sample1_aln \
        --threads 8 \
        --unique yes \
        --samtools_path /path/to/samtools
```

5. `alignment_statistics`

What it does: Calculates alignment statistics (e.g., uniquely mapped reads, alignment rate) from alignment files.

Required Options:

- `--input`: Folder containing alignment files (BAM/SAM).

- `--output`: Folder for statistics output (CSV/Text files).
- `--samtools_path`: Path to `samtools` (for processing BAM files).

Example:

```
python pangraphrna.py alignment_statistics \
        --input /my_alignments \
        --output /my_stats \
        --samtools_path /path/to/samtools
```

6. `expression_quantification`

What it does: Quantifies gene/transcript expression using `StringTie`.

Required Options:

- `--input_path`: Folder with alignment files (BAM).
- `--output_path`: Folder for expression results.
- `--r_path`: Path to R.
- `--samtools_path`: Path to `samtools`.
- `--stringtie_path`: Path to `StringTie`.
- `--threads`: Number of CPU cores.

Example:

```
python pangraphrna.py expression_quantification \
        --input_path /my_alignments \
        --output_path /my_expression \
        --r_path /path/to/R \
        --samtools_path /path/to/samtools \
        --stringtie_path /path/to/stringtie \
        --threads 4
```

7. `differential_expression`

What it does: Identifies differentially expressed genes (DEGs) using `DESeq2`.

Required Options:

- `--input_path`: Folder with expression quantification results.
- `--output_path`: Folder for DEG results.
- `--output_name`: Name for your DEG files.
- `--r_path`: Path to R.
- `--CG`: Number of control group samples.
- `--EG`: Number of experimental group samples.

Example:

```
python pangraphrna.py differential_expression \
        --input_path /my_expression \
        --output_path /my_degs \
        --output_name sample_degs \
        --r_path /path/to/R \
        --CG 3 \  # 3 control samples
        --EG 3   # 3 experimental samples
```

8. `quantitative_trait_locus`

What it does: Identifies eQTLs (expression quantitative trait loci) using `MatrixEQTL`.

Required Options:

- `--variation_genotype`: Path to genotype data (variations).
- `--variation_coordinate`: Path to variation coordinates (chromosome, position).
- `--gene_expression`: Path to gene expression data.
- `--gene_coordinate`: Path to gene coordinates (chromosome, position).
- `--output_path`: Folder for eQTL results.
- `--r_path`: Path to R.
- `--FDR`: False Discovery Rate cutoff (e.g., 0.05).

Example:

```
python pangraphrna.py quantitative_trait_locus \
        --variation_genotype /variants/genotypes.txt \
        --variation_coordinate /variants/variation_coords.txt \
        --gene_expression /my_expression/expression.txt \
        --gene_coordinate /genes/gene_coords.txt \
        --output_path /my_eqtls \
        --r_path /path/to/R \
        --FDR 0.05
```