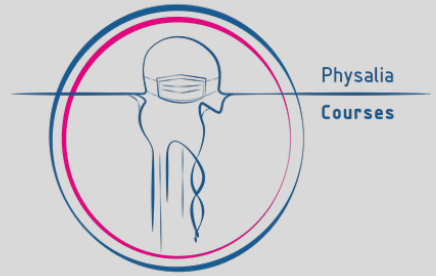
NGS workflows: Nextflow, Snakemake, ...

Epigenomics Data Analysis

Jacques Serizay

Physalia 2023



Why workflows?

- Workflows allow to write scalable and highly parallelizable scientific workflows.

Scalability
Parallelization

Why workflows?

- Workflows allow to write scalable and highly parallelizable scientific workflows.
- Workflows rely on software containers for reproducibility and portability.

Scalability
Parallelization
Reproducibility
Portability

Why workflows?

- Workflows allow to write scalable and highly parallelizable scientific workflows.
- Workflows rely on software containers for reproducibility and portability.
- Workflows can contain parts written in most common scripting languages.

Scalability
Parallelization
Reproducibility
Portability
Universality

Why workflows?

- Workflows allow to write scalable and highly parallelizable scientific workflows.
- Workflows rely on software containers for reproducibility and portability.
- Workflows can contain parts written in most common scripting languages.
- Continuous checkpoints are automatically tracked, and workflows can be resumed seamlessly.

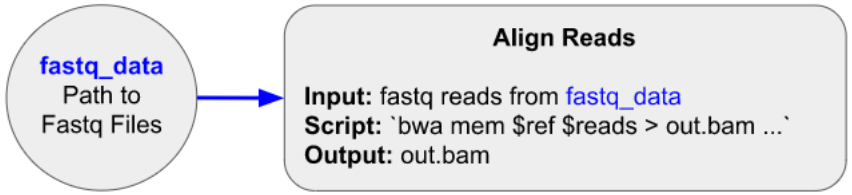
Scalability
Parallelization
Reproducibility
Portability
Universality
Progress tracking

Why workflows?

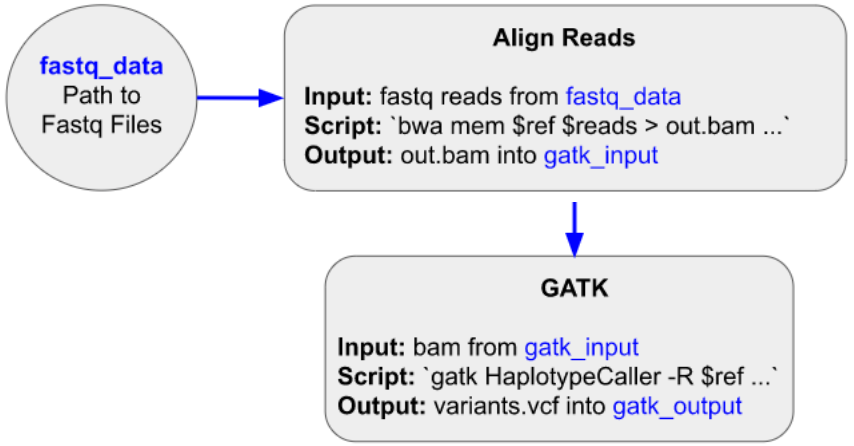
When/what can it be used for?

- To split fastq/bam/… files in smaller files
- To map hundreds of ChIP-seq datasets in parallel, in 1 keystroke
- To work efficiently in an HPC (High Performance Cluster) environment (e.g. AWS, Slurm, …)
- To reproduce a published analysis without having to wonder what parameter was used here or there
- …

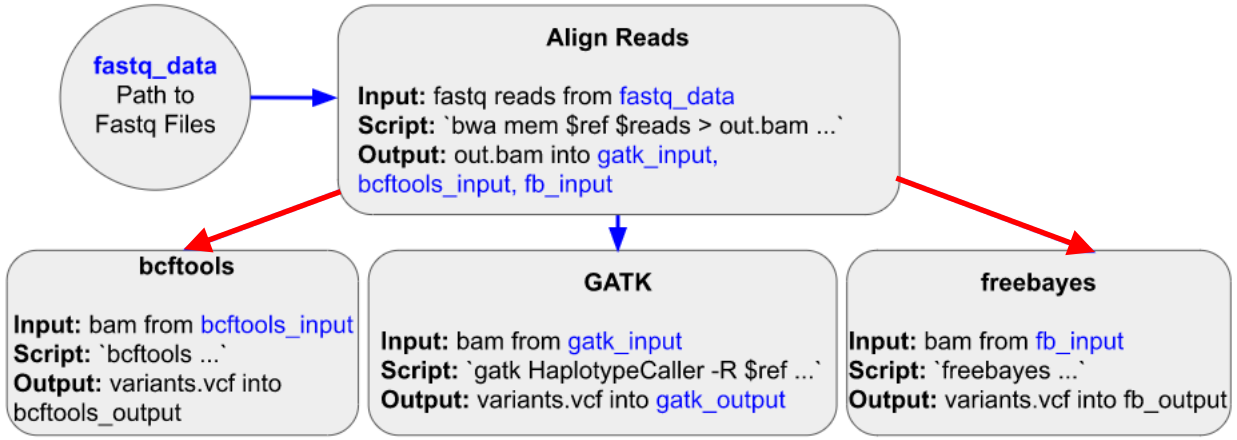
How it works


fastq_data
Path to
Fastq Files
Align Reads
Input: fastq reads from fastq_data
Script: `bwa mem $ref $reads > out.bam ...`
Output: out.bam

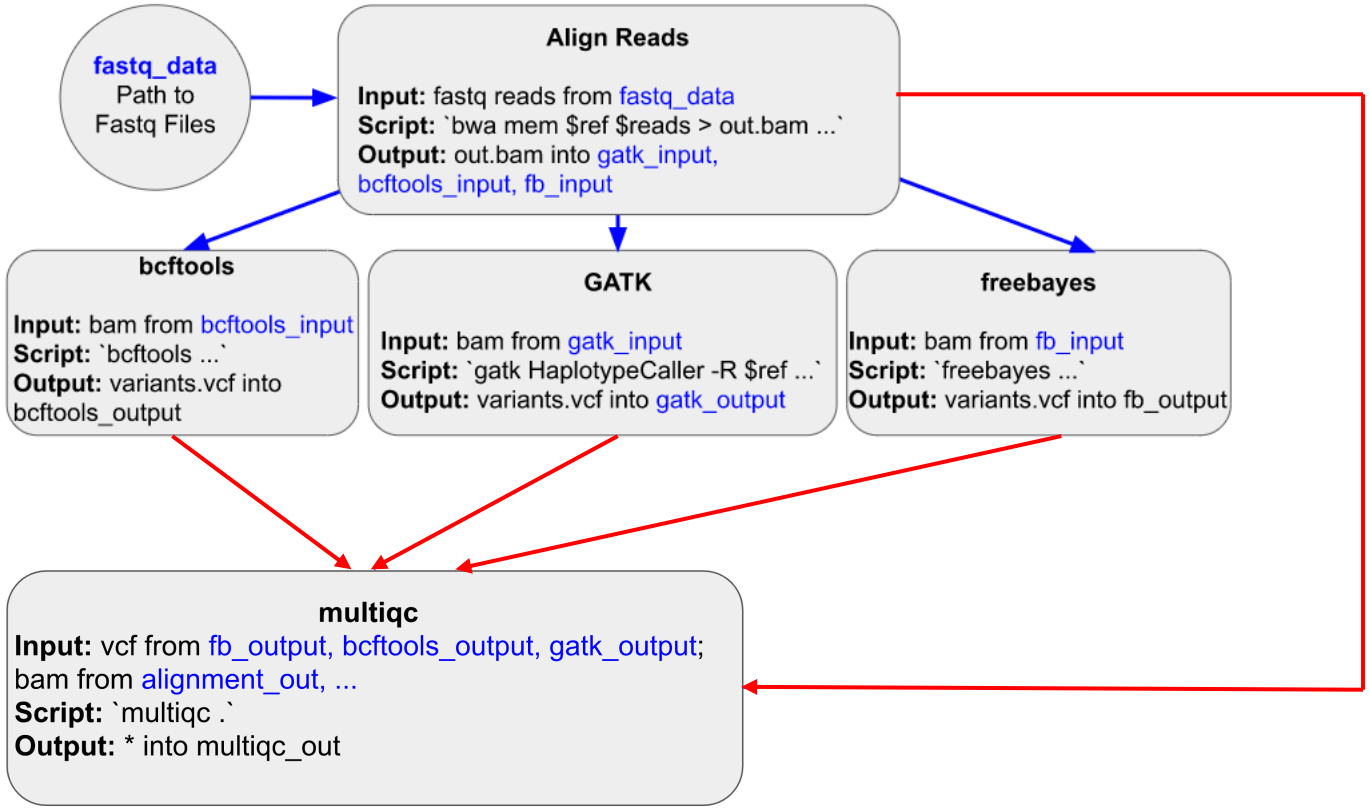
How it works


fastq_data
Path to
Fastq Files
Align Reads
Input: fastq reads from fastq_data
Script: `bwa mem $ref $reads > out.bam ...`
Output: out.bam into gatk_input
GATK
Input: bam from gatk_input
Script: `gatk HaplotypeCaller -R $ref ...`
Output: variants.vcf into gatk_output

How it works


fastq_data
Path to
Fastq Files
Align Reads
Input: fastq reads from fastq_data
Script: `bwa mem $ref $reads > out.bam ...`
Output: out.bam into gatk_input,
bcftools_input, fb_input
bcftools
Input: bam from bcftools_input
Script: `bcftools ...`
Output: variants.vcf into
bcftools_output
GATK
Input: bam from gatk_input
Script: `gatk HaplotypeCaller -R $ref ...`
Output: variants.vcf into gatk_output
freebayes
Input: bam from fb_input
Script: `freebayes ...`
Output: variants.vcf into fb_output

How it works


fastq_data
Path to
Fastq Files
Align Reads
Input: fastq reads from fastq_data
Script: `bwa mem $ref $reads > out.bam ...`
Output: out.bam into gatk_input,
bcftools_input, fb_input
bcftools
Input: bam from bcftools_input
Script: `bcftools ...`
Output: variants.vcf into
bcftools_output
GATK
Input: bam from gatk_input
Script: `gatk HaplotypeCaller -R $ref ...`
Output: variants.vcf into gatk_output
freebayes
Input: bam from fb_input
Script: `freebayes ...`
Output: variants.vcf into fb_output
multiqc
Input: vcf from fb_output, bcftools_output, gatk_output;
bam from alignment_out, ...
Script: `multiqc .`
Output: * into multiqc_out

Nextflow documentation

Latest documentation: https://www.nextflow.io/docs/latest/index.html

Working examples: https://github.com/nextflow-io/awesome-nextflow

Workflows for standard protocols: https://github.com/nf-core

Recurrent implementation patterns used in Nextflow applications:

https://nextflow-io.github.io/patterns/index.html