Assembly through the command line.

- The following details how to run the analysis through the command line.
- The underlying analysis is the same, but we go into greater detail, breaking down each command that ran behind the scenes in the previous tutorial.

The Shell

A Shell is a program that provides a text only user interface for interacting with the computer. The shell consists of a command prompt, showing the user name and location, and the command line, where commands are entered.

The Command Prompt

- The command prompt shows basic information.

Name of logged in user.

Name of computer or server where the user is logged in.

```
[user_name@computer_name folder_name]$_
```

Current location in the computer's file structure.

The Command Line

- Run programs and navigate files by typing commands into the command line, next to the command prompt.

Program

```
[user_name@computer_name folder_name]$ fastqc --help
```

Option

--help

- Note that for most programs that run on the command line, the manual can be accessed by typing the name of the program, followed by a space and “-h” or “--help” as in the example below.

Program

```
[user_name@computer_name folder_name]$ fastqc --help
```

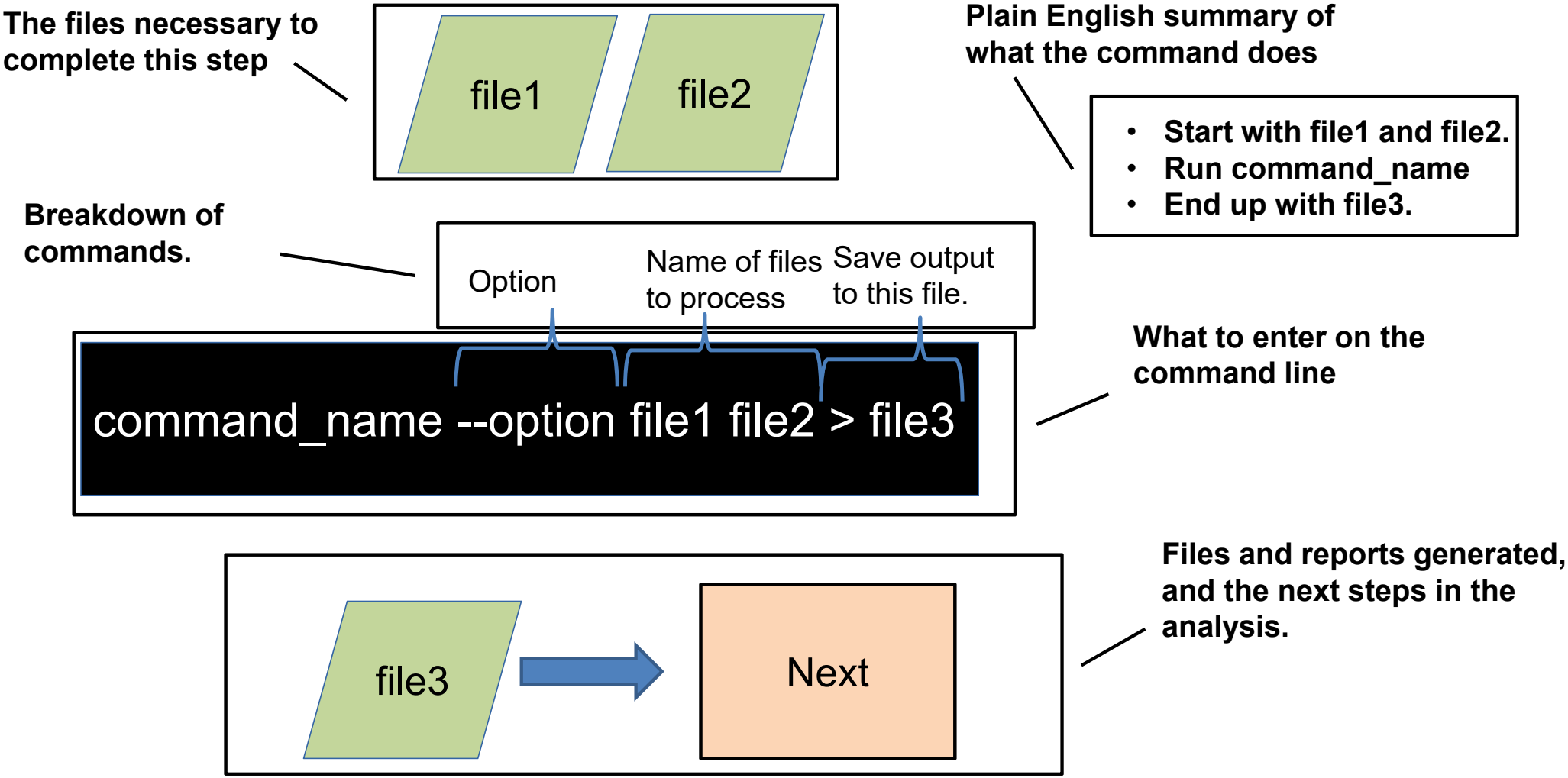
Option

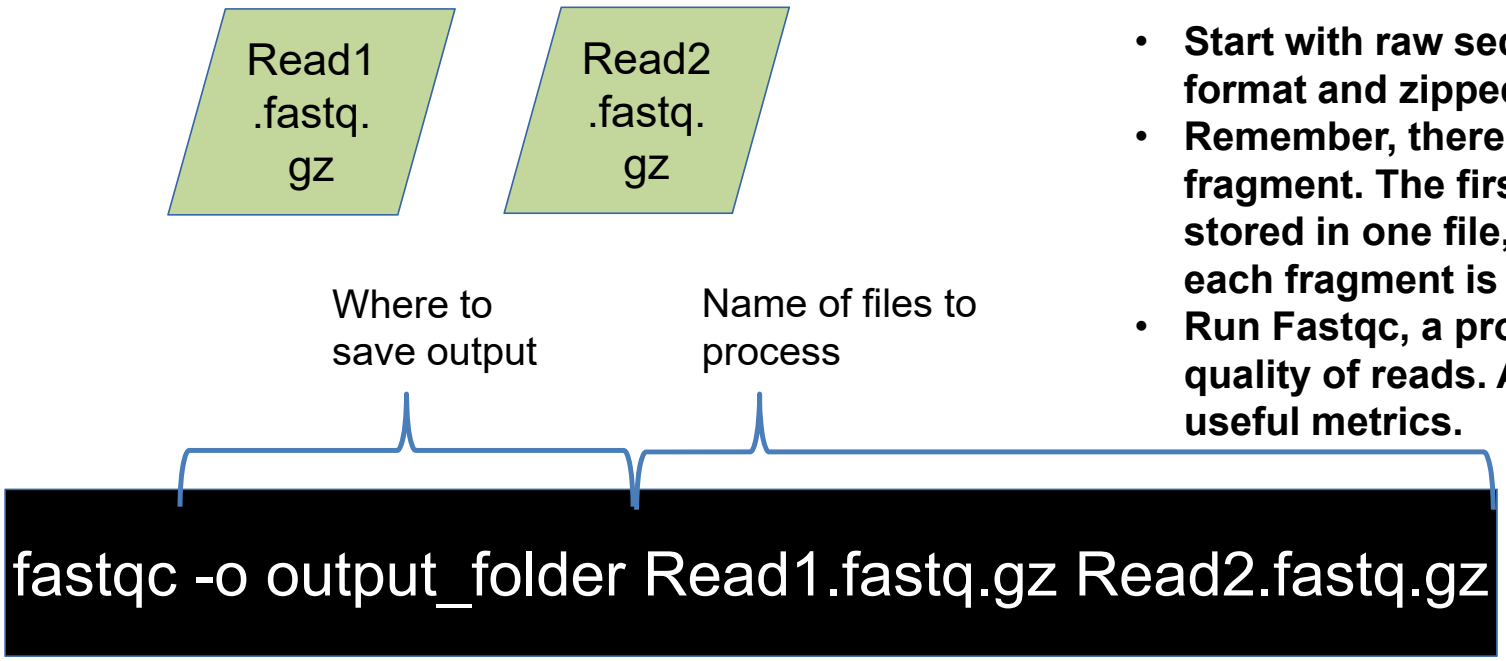
Command line caveats

- Commands are case sensitive. Enter commands exactly as written!
- Spacing and ordering of arguments are important, and can change the output of the command, so enter the commands exactly as written!
- Pressing enter runs the command as it appears in the command line. There is no warning or confirmation!
- Shells are text only interfaces. You cannot click on a space in the command line with the mouse. Use the arrow keys to navigate.

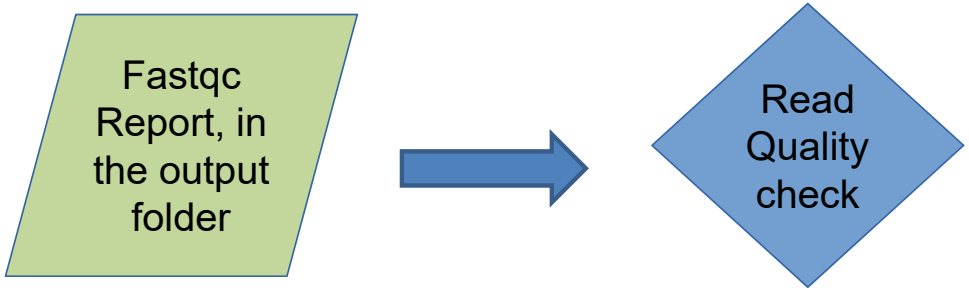
Benefits of using the command line.

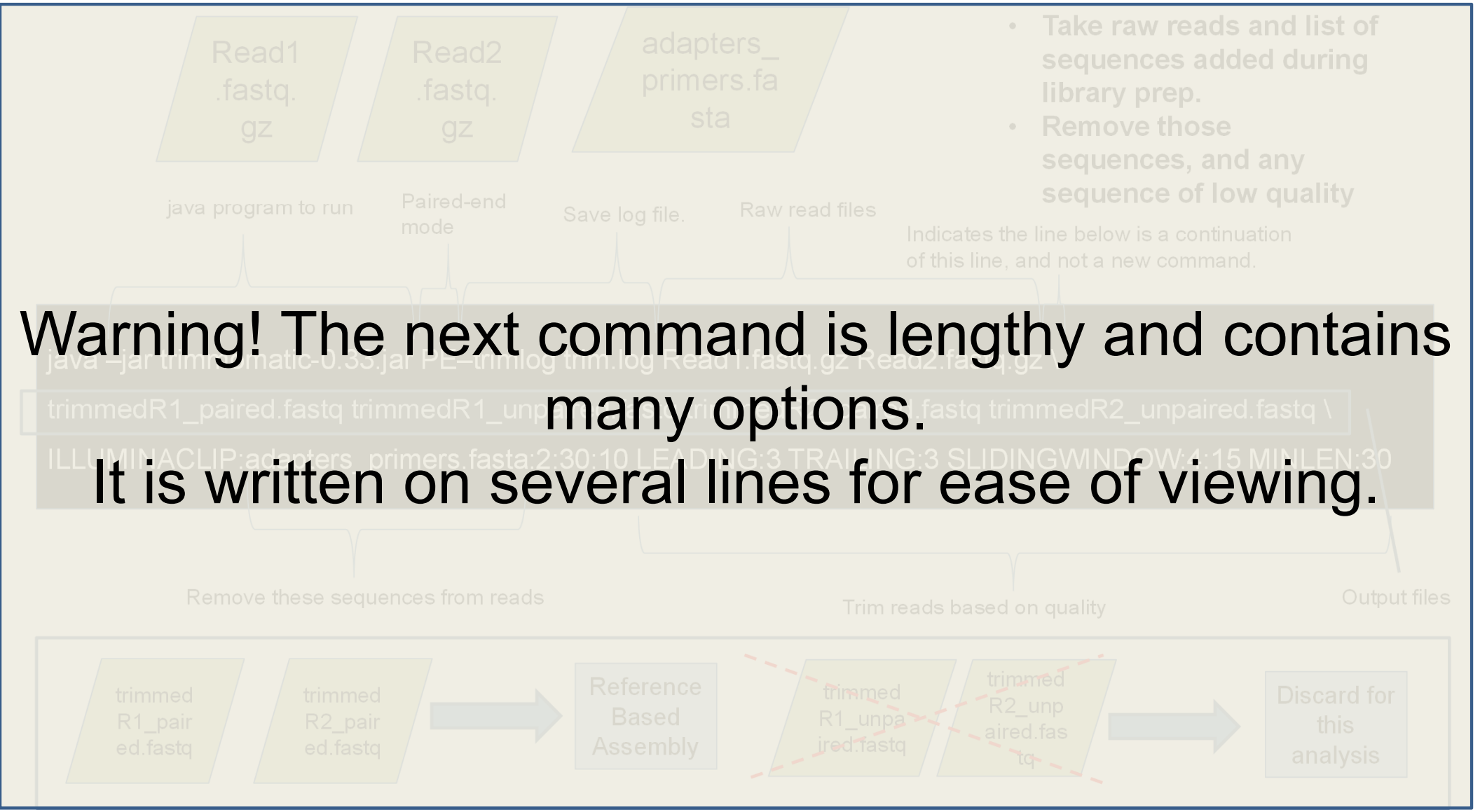
- Finer control of program parameters.
- Can string together multiple programs into analysis pipelines.
- Record of exactly what commands and parameters have been run.
- Increased portability and reproducibility.


The files necessary to complete this step
file1
file2
Plain English summary of what the command does
• Start with file1 and file2.
• Run command_name
• End up with file3.
Breakdown of commands.
Option
Name of files to process
Save output to this file.
command_name --option file1 file2 > file3
What to enter on the command line
file3
Next
Files and reports generated, and the next steps in the analysis.

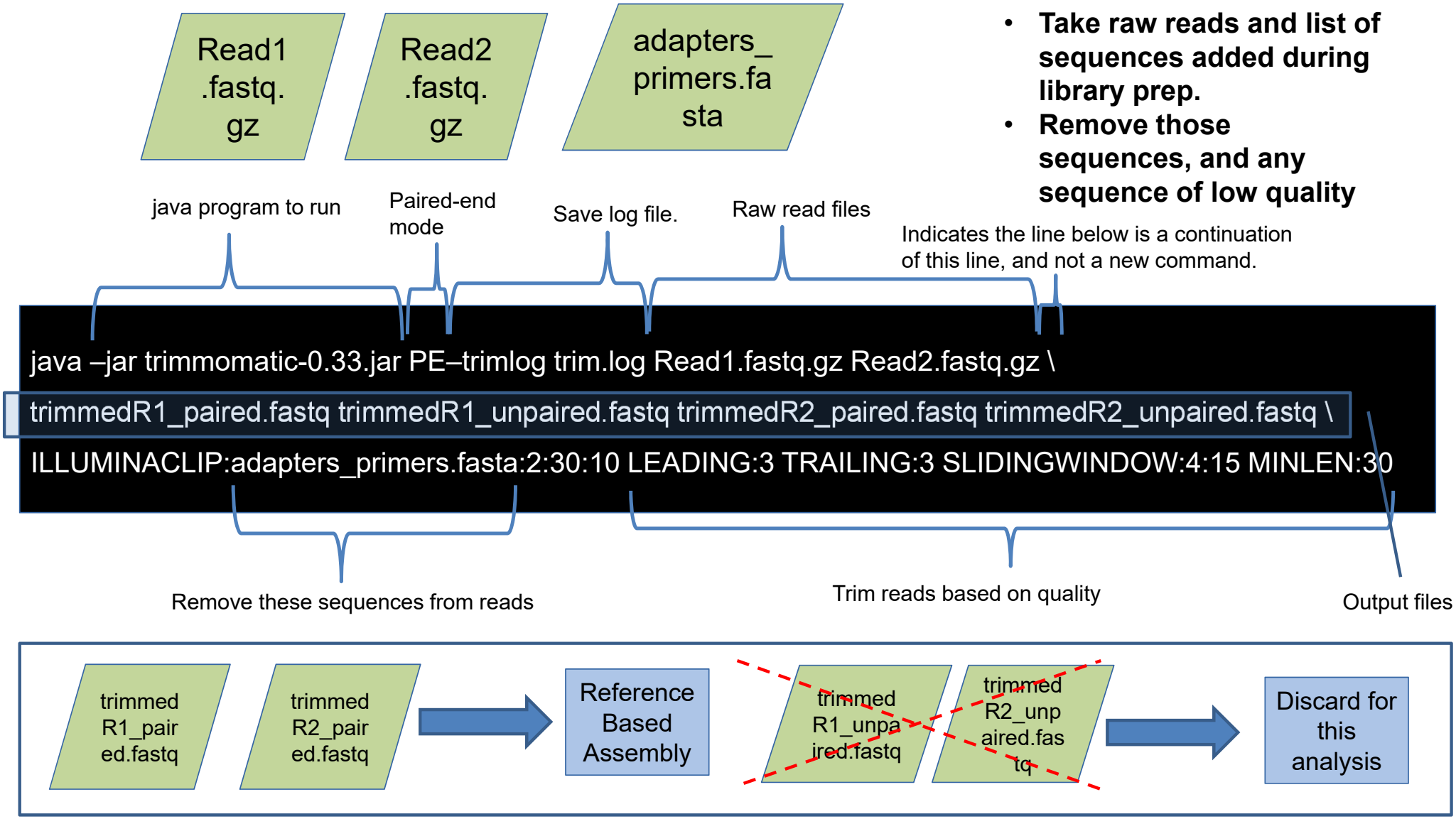


- **Start with raw sequencing data, in fastq format and zipped.**
- **Remember, there are two reads for each DNA fragment. The first read of each fragment is stored in one file, and the second read of each fragment is stored in another.**
- **Run Fastqc, a program that summarizes the quality of reads. Also outputs a number of useful metrics.**





Warning! The next command is lengthy and contains many options.
It is written on several lines for ease of viewing.



- **Take raw reads and list of sequences added during library prep.**
- **Remove those sequences, and any sequence of low quality**

Read1 .fastq. gz

Read2 .fastq. gz

adapters_ primers.fa sta

- Take raw reads and list of sequences added during library prep.
- Remove those sequences, and any sequence of low quality

java program to run

Paired-end mode

Save log file.

Raw read files

Indicates the line below is a continuation of this line, and not a new command.

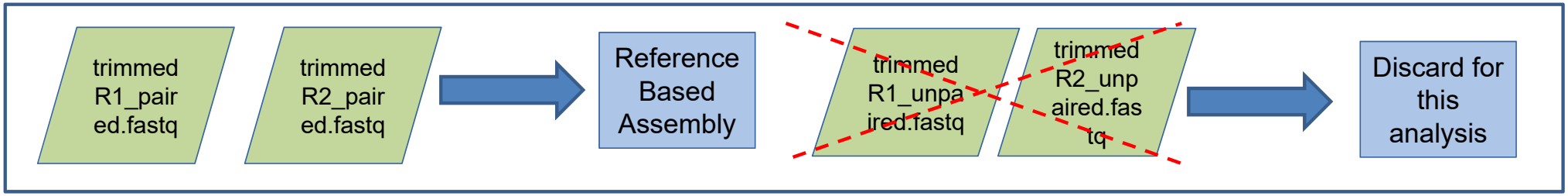
Note that this command requires outputting 4 files, but we will only use two in the subsequent steps.

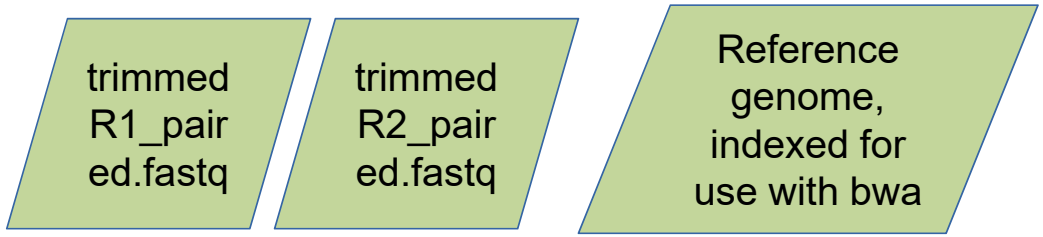
```
trimmedR1_paired.fastq trimmedR1_unpaired.fastq trimmedR2_paired.fastq trimmedR2_unpaired.fastq \
ILLUMINACLIP:adapters_primers.fasta:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:30
```

Remove these sequences from reads

Trim reads based on quality

Output files





- **Start with quality reads and a reference genome.**
- **Use bwa mem to align the reads to the reference.**
- **Save the output in a .sam file, which links the read to a location in the reference genome where the read aligns.**

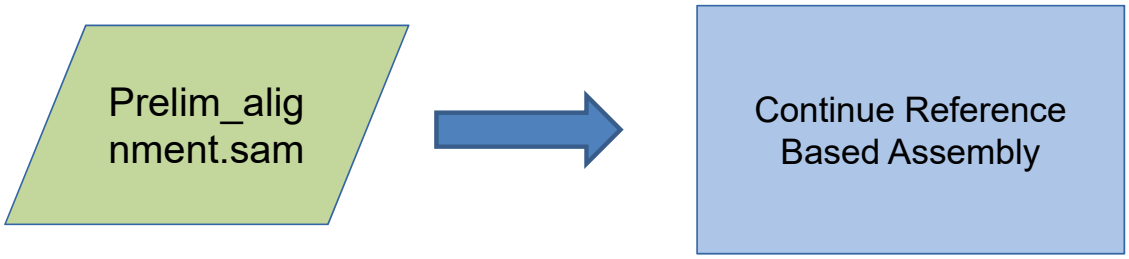
Mapping algorithm

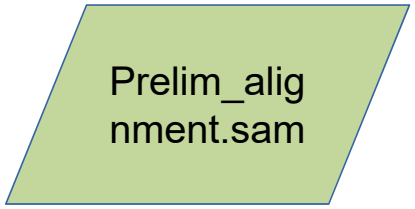
Name of reference.

Name of files to process

```
bwa mem ebola_ref trimmedR1_paired.fastq trimmedR2_paired.fastq \
> Prelim_alignment.sam
```

Save output in a .sam file.





- **Sort the alignment file (.sam) by location in the reference genome, and save as a bam.**
- **This makes the data easier to process by downstream programs.**

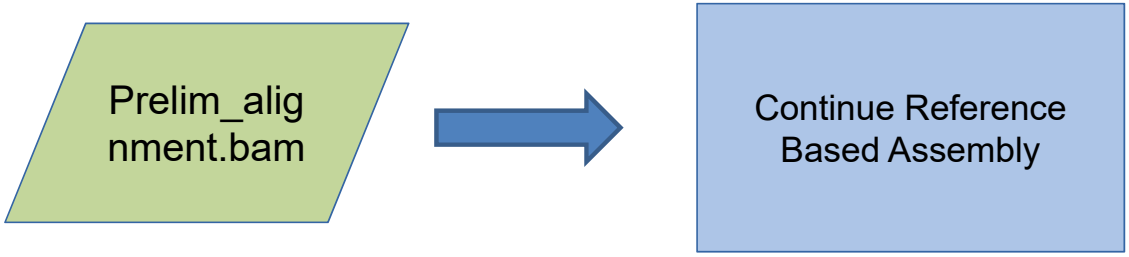
Sort the .sam by reference location.

Save the output in .bam format

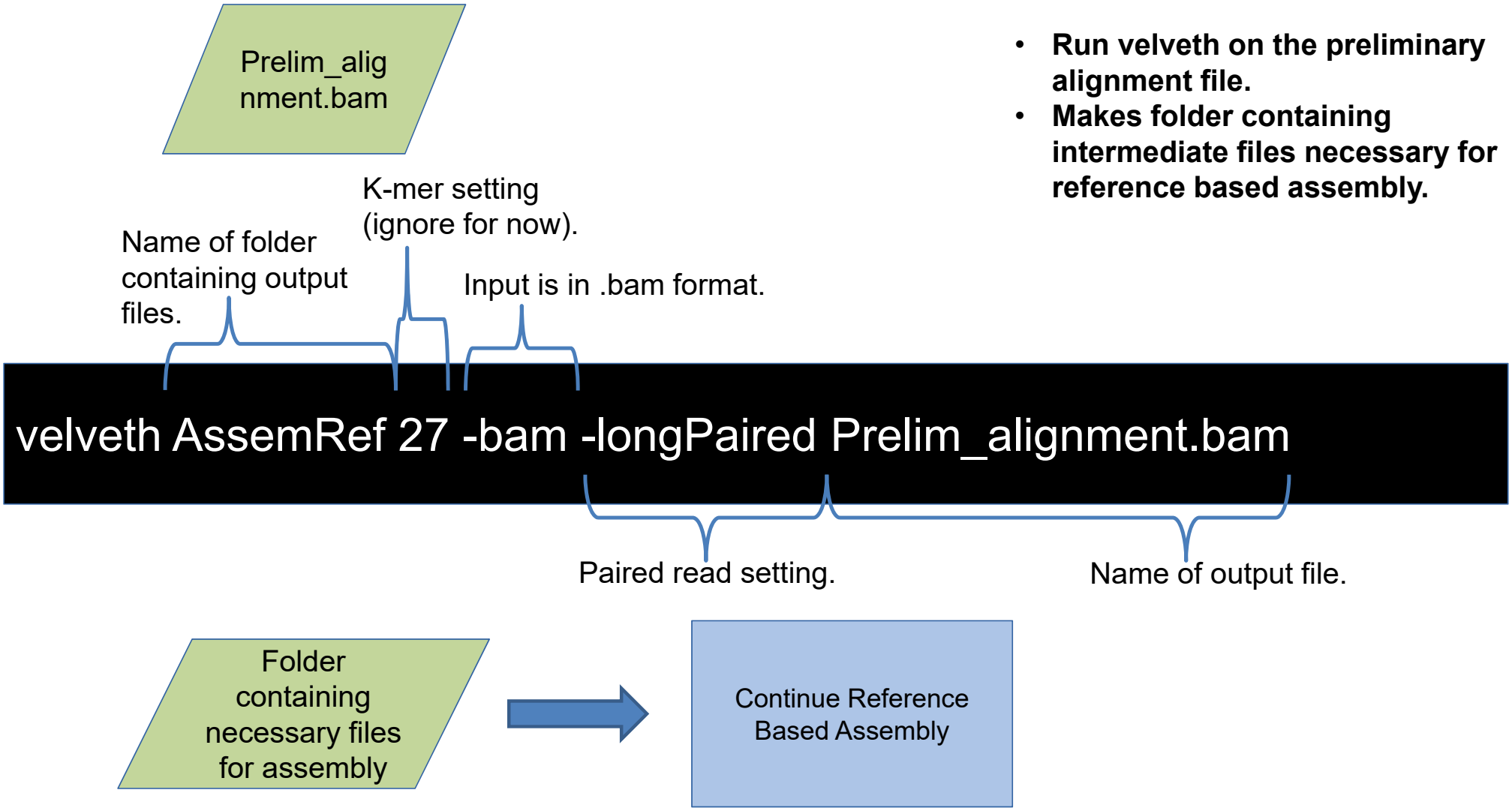
The input file

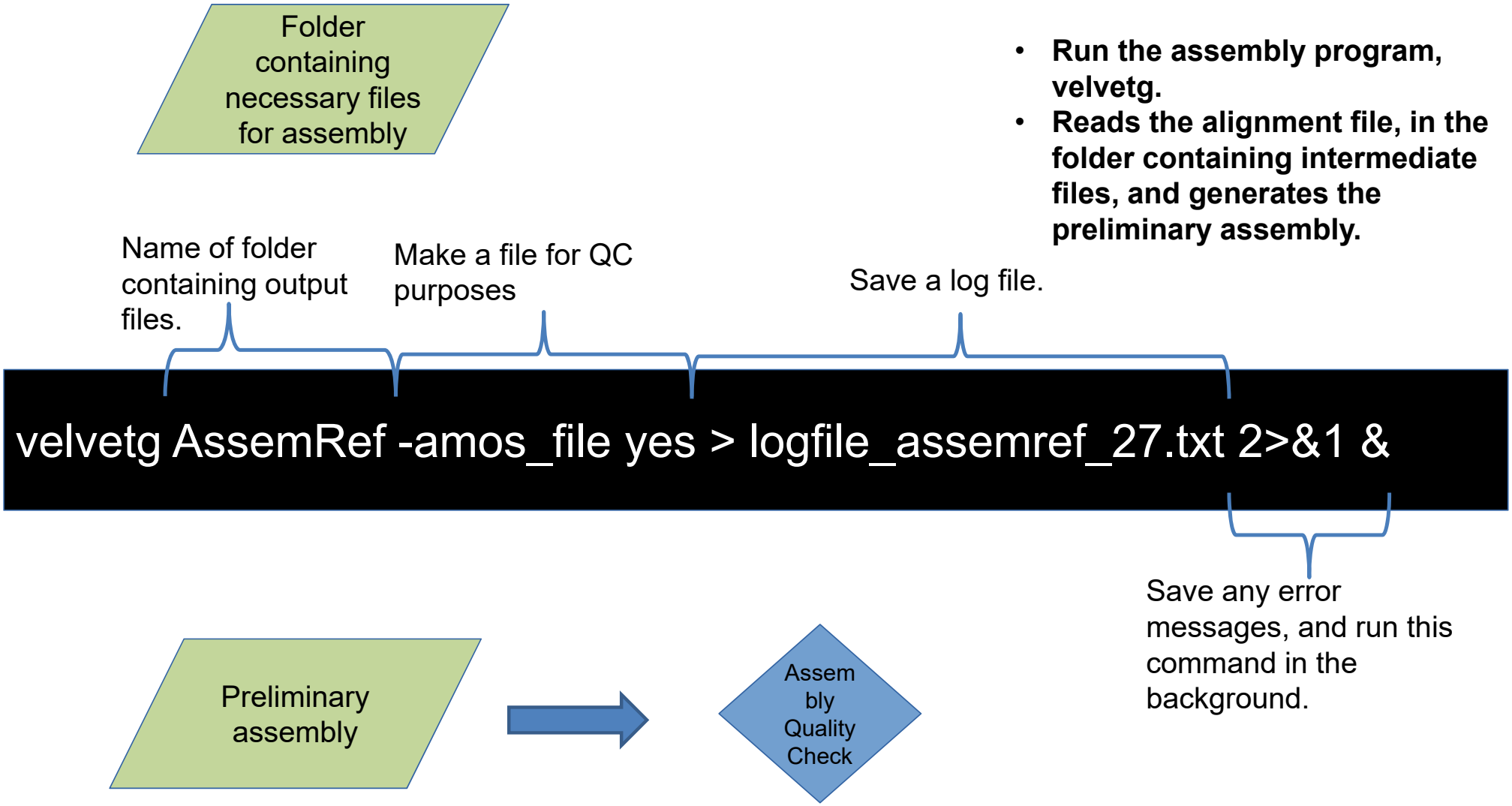
```
samtools sort -O BAM Prelim_alignment.sam > Prelim_alignment.bam
```

Save the output in a .bam file.

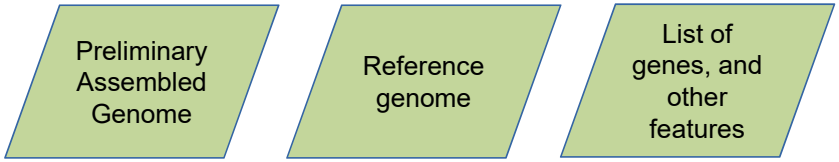


- **Run velveth on the preliminary alignment file.**
- **Makes folder containing intermediate files necessary for reference based assembly.**





- **Run the assembly program, velvetg.**
- **Reads the alignment file, in the folder containing intermediate files, and generates the preliminary assembly.**



- **Quast compares the preliminary assembly to a known genome from the same species.**
- **Also, identifies functional sequences, like genes and RNAs.**

Preliminary assembly

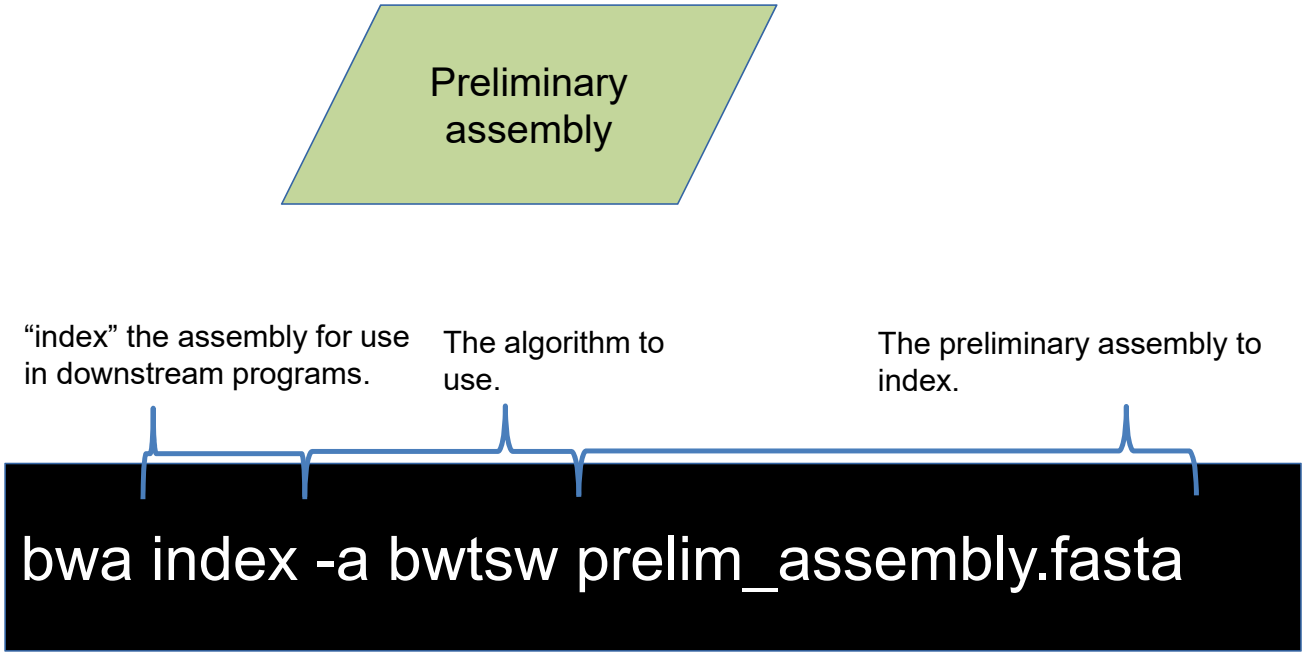
The reference .fasta file.

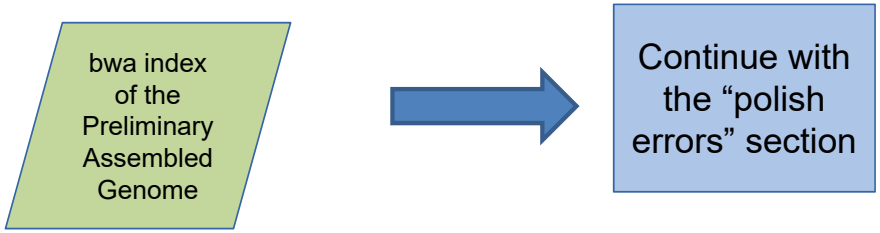
```
quast.py prelim_assembly.fasta –R ebola_ref.fasta \
–G ebola_ref_genes.gff –o output_folder –glimmer
```

List of genes in the reference.

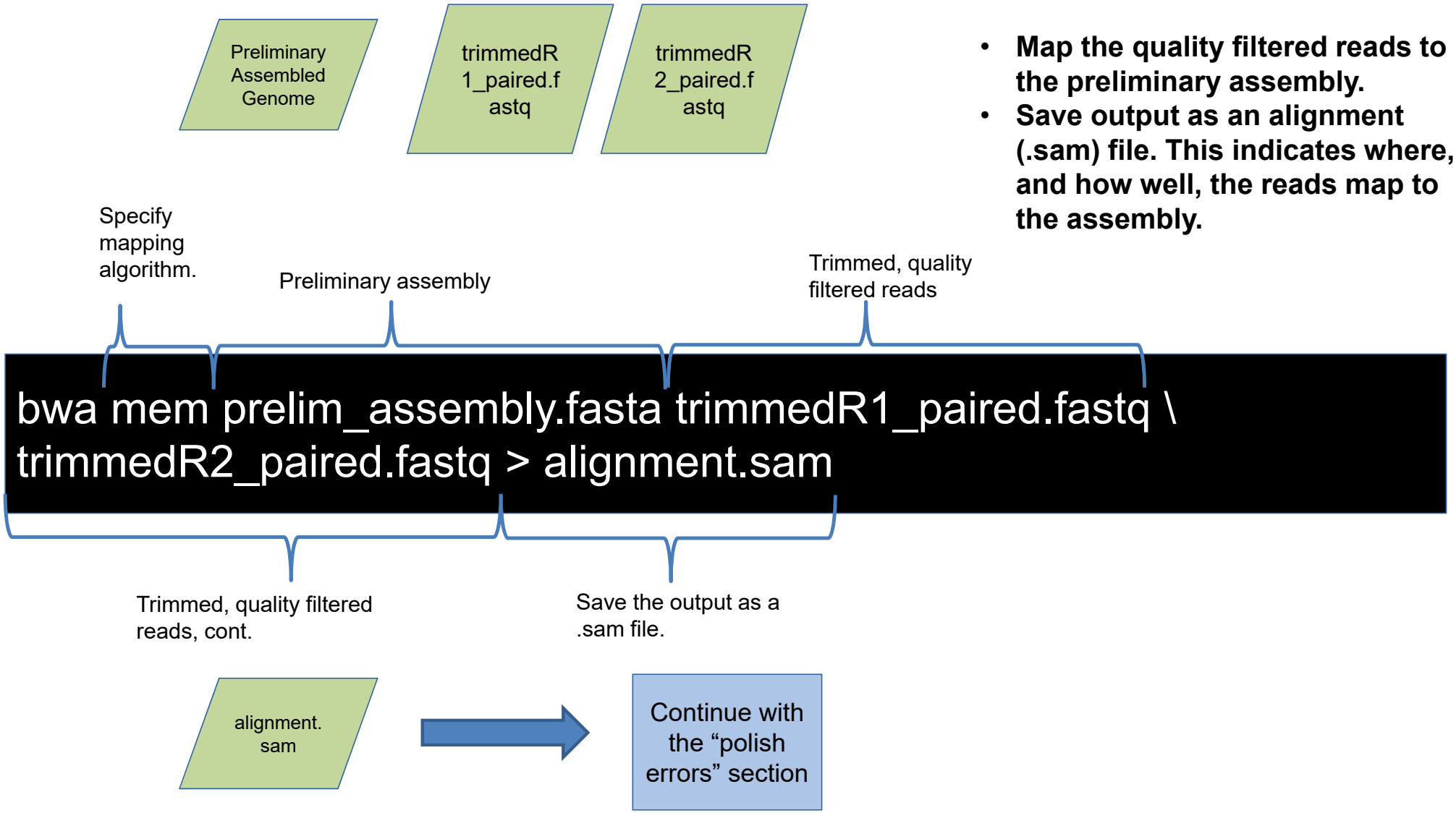
Output folder containing results

Provide list of gene locations in the preliminary assembly.



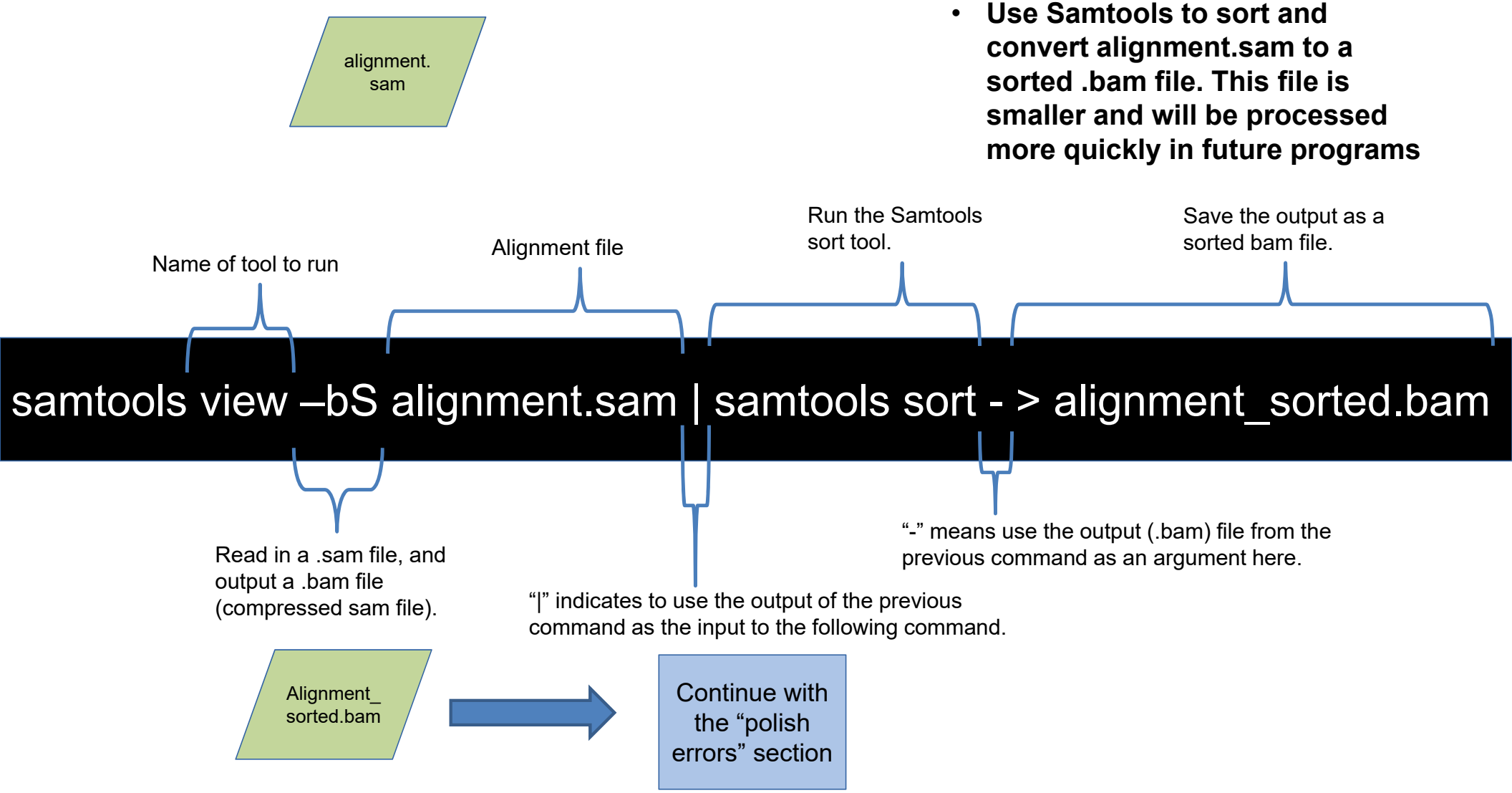


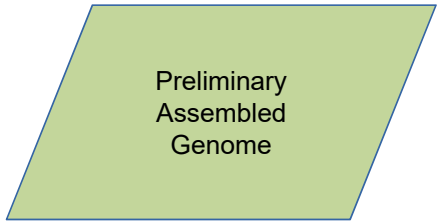
- **"Polish" out errors in the assembly by mapping the reads back to the assembly.**
- **This will take several steps.**
- **Map the quality filtered reads to the preliminary assembly.**
- **The index can then be used by the mapping program, bwa mem, in the next step.**



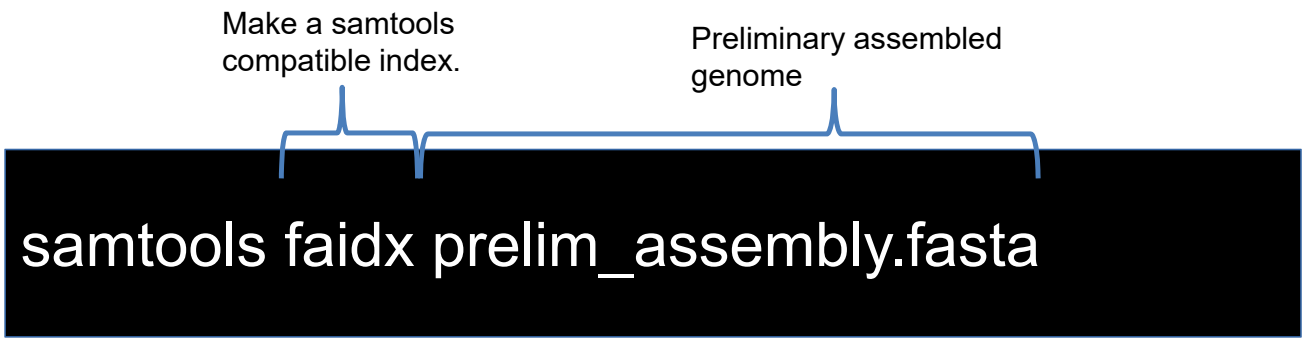
- **Map the quality filtered reads to the preliminary assembly.**
- **Save output as an alignment (.sam) file. This indicates where, and how well, the reads map to the assembly.**

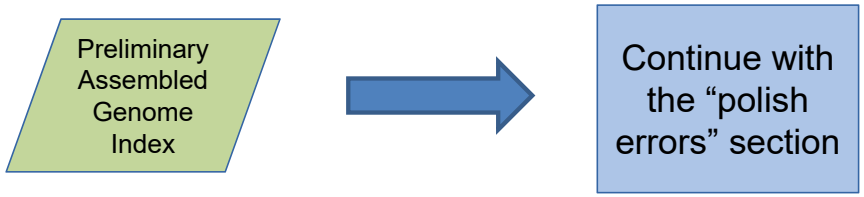
- **Use Samtools to sort and convert alignment.sam to a sorted .bam file. This file is smaller and will be processed more quickly in future programs**

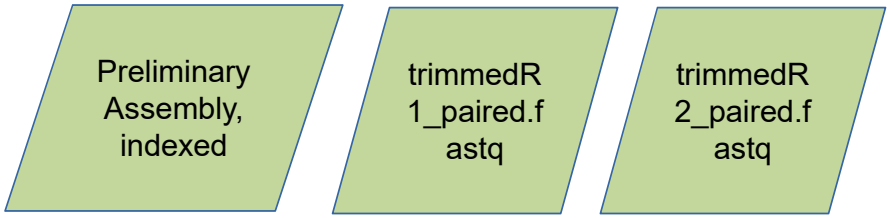




- **Make an index of the preliminary assembled genome.**
- **This is necessary to use the assembly in the next step.**







- **Samtools mpileup estimates the probability that each base in the preliminary assembly is correct, given the reads that map to it.**
- **bcftools call determines if the reads provide support for a different base at any given position in the assembly, based on these probabilities.**
- **The output is a list of positions in the preliminary assembly that should be changed, in the "variant call format (.vcf)."**

Tool for genotype estimation, based on alignment.

Output uncompressed, bcf format

The input reference file

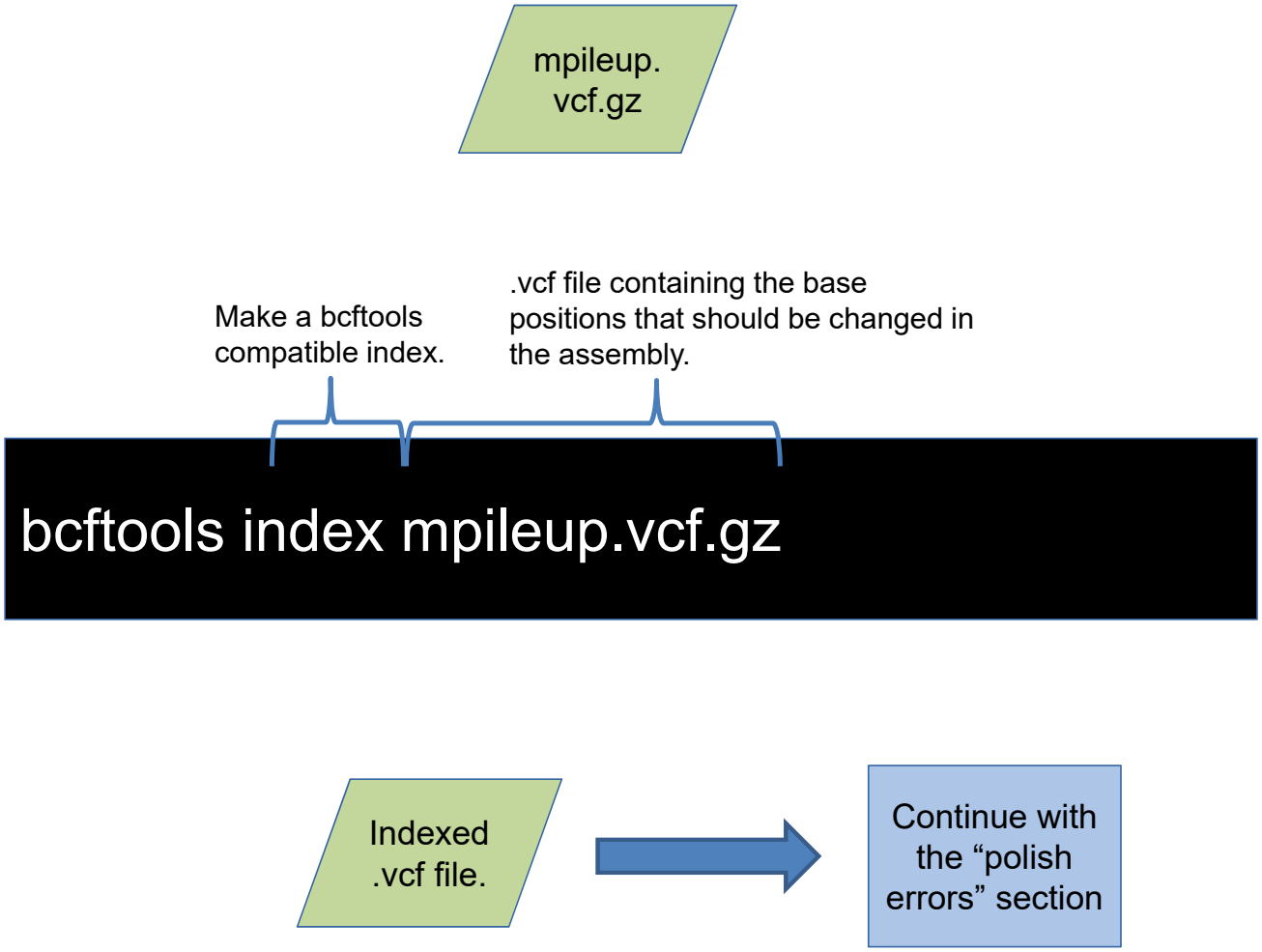
The alignment file

Output only positions that differ, in a compressed format.

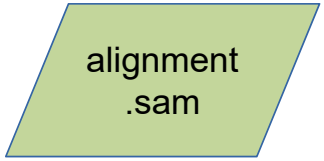
```
samtools mpileup -u -g -f prelim_assembly.fasta alignment_sorted.bam | bcftools call -v -m -O z -o mpileup.vcf.gz > logfile.txt
```

Save the output as mpileup.vcf.gz, and keep logfile.txt, a record of the steps run by each program.

Identifies bases in assembly not supported by reads.



- **Make an index of the .vcf file, which will make processing the file easier in the next step.**



- **Read in the preliminary assembly.**
- **Use "bcftools consensus" to make a new fasta with the corrected sequences.**

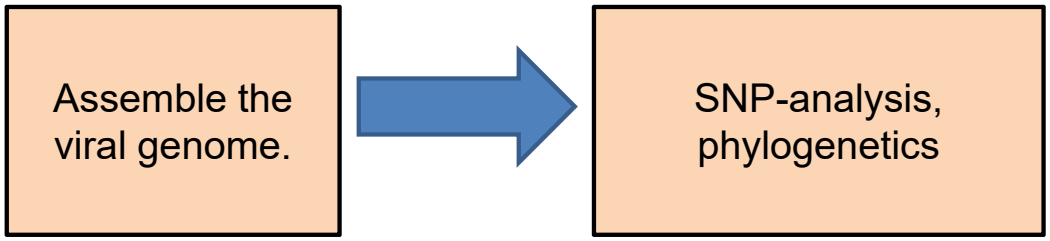
Read in the preliminary assembly

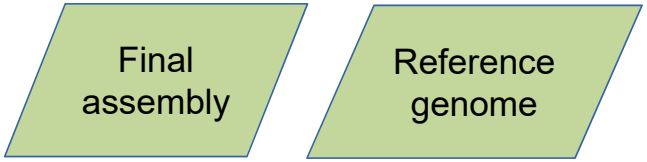
Make a new fasta containing the corrected sequences

The .vcf file containing list of changes to make.

Save the new assembly

```
cat prelim_assembly.fasta | bcftools consensus mpileup.vcf.gz > final_assembly.fasta
```





- **Align the final assembly to the reference genome.**
- **This alignment will be used as input for the phylogenetic analysis.**

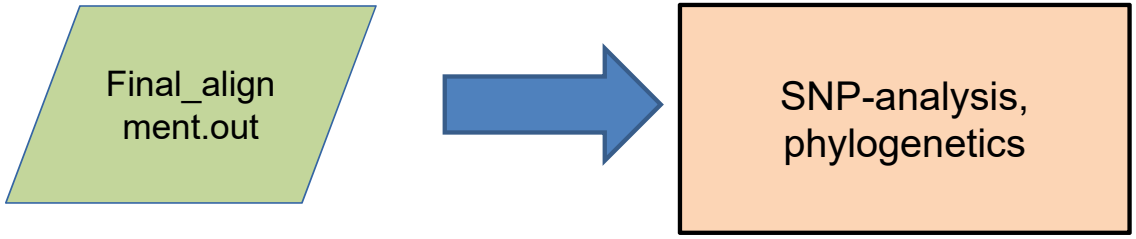
Read in the preliminary assembly

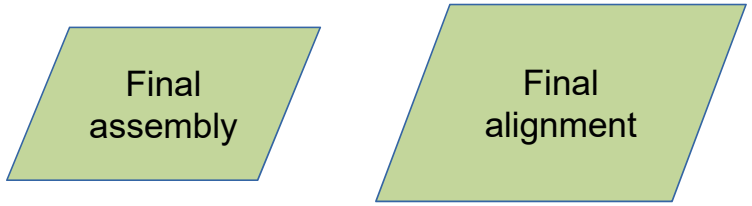
Run another aligner: mafft

Reference Genome

Save the alignment file.

```
cat final_assembly.fasta | mafft ebola_ref.fasta > Final_alignment.out
```



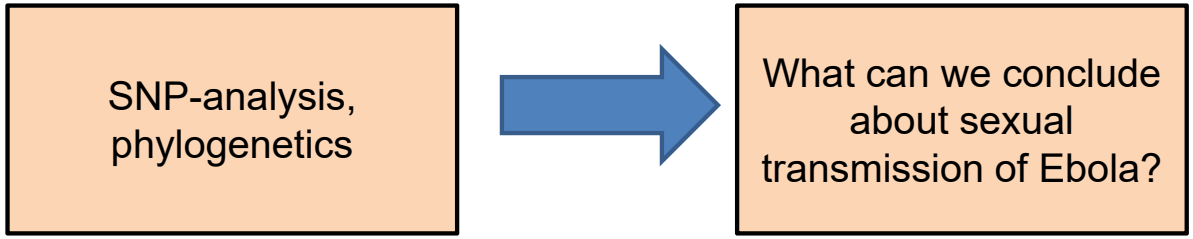


- **Run the haplonetwork_analysis.Rscript file, which contains instructions for running the haplotype network analysis in the R environment.**

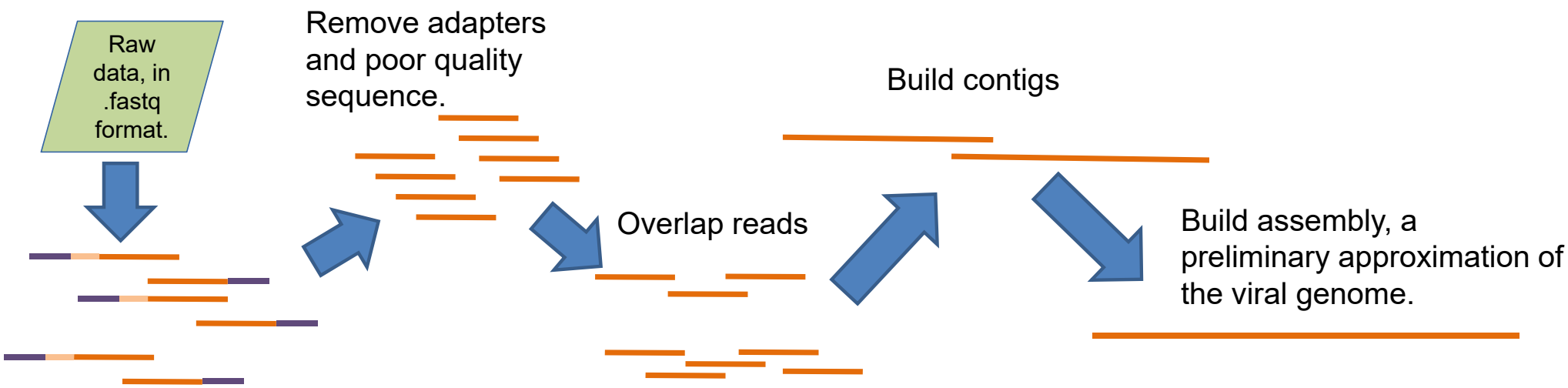
A script (code) that runs the phylogenetic analysis.

The final alignment file.

```
Rscript haplonetwork_analysis.Rscript Final_alignment.out
```



De Novo Assembly



Raw data consists of sequences containing fragments of the Ebola genome. Ultimately, we need to take these fragments and assemble them into the complete genome.

File types

.gz	Appended to files that are compressed
.fasta	Simple format for storing sequence information.
.fastq	Stores sequence and quality information
.gff	General Feature Format: a list of genes and other genomic features, and their location in a particular genome.
.sam	Sequence Alignment/Map format. Links sequences (as from reads) to a position in a reference genome.
.bam	The compressed version of a .sam file.
.vcf	Variant Call Format; stores information about variation between sequences, as between reads and a reference genome.