

Data Wrangling and Processing for Genomics

Elizabeth Everman

Postdoctoral Fellow, Macdonald Lab, Molecular Biosciences

Goals for today:

- Background and metadata (2_DataWrangling.md)
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Goals for today:

- [Background and metadata \(2_DataWrangling.md\)](#)
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Escherichia coli:

- Single-celled
- Fast generation time (doubles every 20 minutes)
- Low genetic complexity (comparatively)
- Plethora of genetic tools and protocols are available to study adaptation

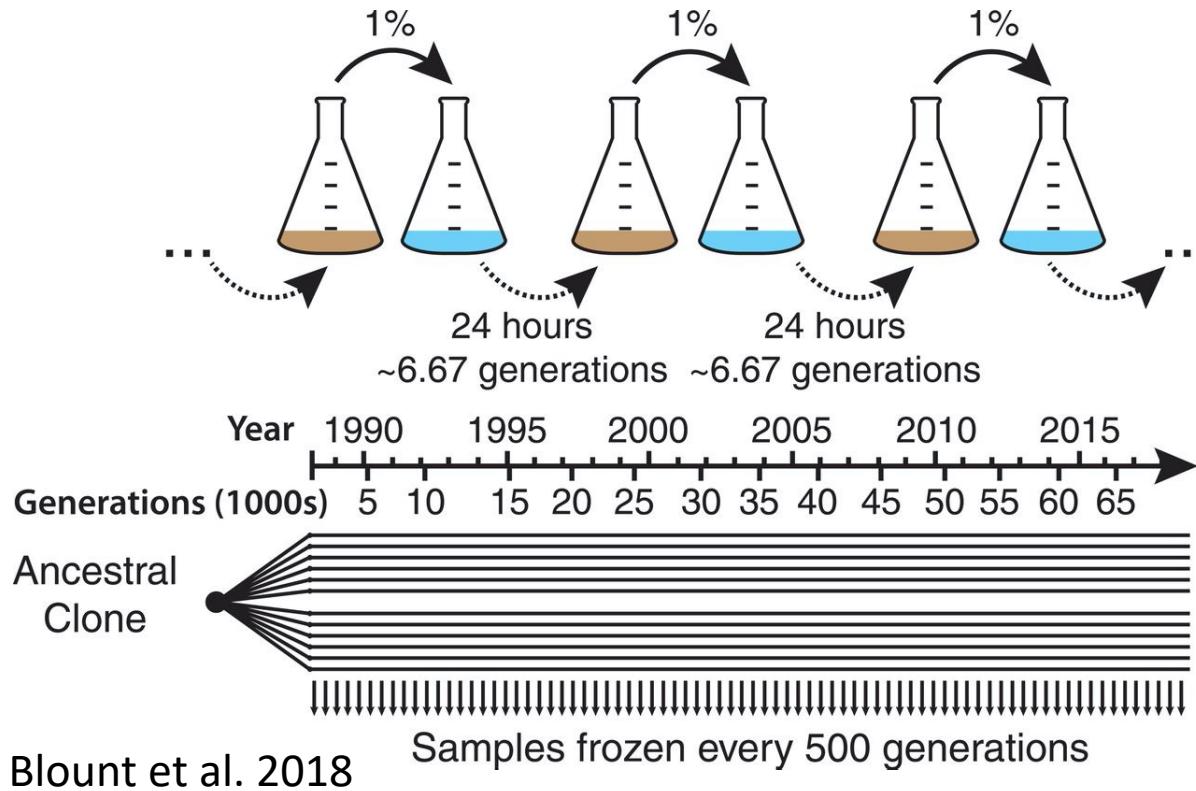
Data from:

<https://science.sciencemag.org/content/362/6415/eaam5979/tab-pdf>



STEVE GSCHMEISSNER—Getty Images/Science Photo Library RM

Long-Term Evolution Experiment with *E. coli*:



Wikipedia

- Glucose-limited media
- High concentration of citrate
 - Normally not usable under aerobic conditions.
 - Between generations 31,000 and 31,500, a spontaneous citrate using mutant appeared
 - Certain regions of the genome became hypermutable

Our data:

- Subset of the larger experiment
- We will work with three sample events of the Ara-3 strain
 - Generations 5,000, 15,000, and 50,000
- This population changed substantially through the experiment
- OUR GOAL:
 - Identify and characterize adaptive changes using a variant calling workflow
- Step 1: Access and examine the metadata associated with this project.
 - Download: https://raw.githubusercontent.com/datacarpentry/wrangling-genomics/gh-pages/files/Ecoli_metadata_composite.csv
 - View: https://github.com/datacarpentry/wrangling-genomics/blob/gh-pages/files/Ecoli_metadata_composite.csv

Metadata:

- **strain** = strain name
- **generation** = generation when sample was frozen
- **clade** = based on parsimony-based tree
- **reference** = study the samples were originally sequenced for
- **population** = ancestral population group
- **mutator** = hypermutability mutant status
- **facility** = facility samples were sequenced at
- **run** = sequence read archive sample ID
- **read_type** = library type of reads
- **read_length** = length of reads in sample
- **sequencing_depth** = depth of sequencing
- **cit** = citrate-using mutant status

Metadata:

- **strain** = strain name
- **generation** = generation when sample was frozen
- **clade** = based on parsimony-based tree
- **reference** = study the samples were originally sequenced for
- **population** = ancestral population group
- **mutator** = hypermutability mutant status
- **facility** = facility samples were sequenced at
- **run** = sequence read archive sample ID
- **read_type** = library type of reads
- **read_length** = length of reads in sample
- **sequencing_depth** = depth of sequencing
- **cit** = citrate-using mutant status

Based on the metadata, answer the following questions:

1. How many different generations exist in the data?
2. How many rows and how many columns are in this data?
3. How many citrate+ mutants have been recorded in Ara-3?
4. How many hypermutable mutants have been recorded in Ara-3?

Questions? Time for a break?

Goals for today:

- Background and metadata
- Assessing sequencing read quality (3_AssessingReadQuality.md)
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Objectives:

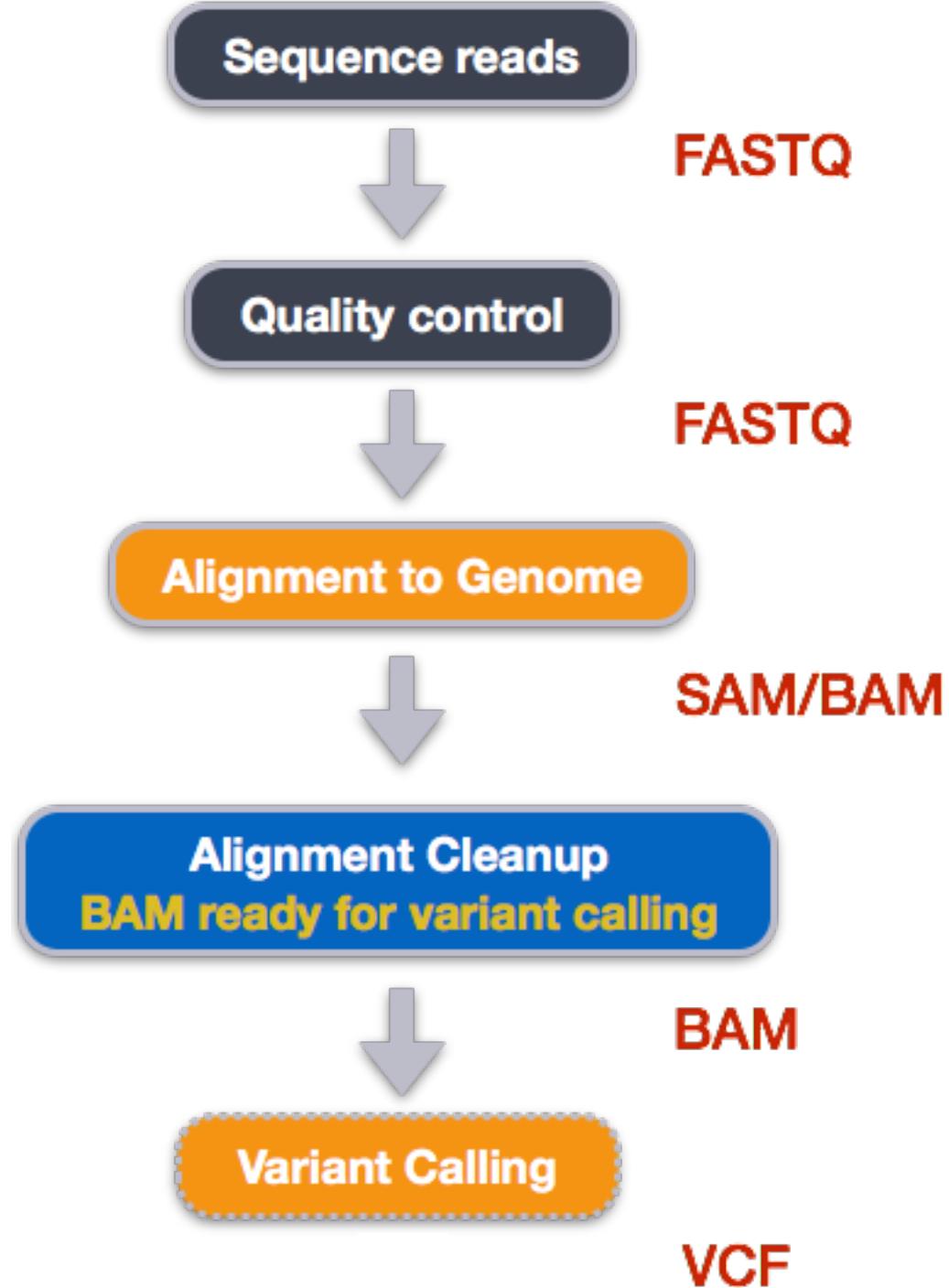
- Understand how a FASTQ file encodes per-base quality scores.
- Interpret a FastQC plot summarizing per-base quality across all reads.
- Use for loops to automate operations on multiple files.

Things that can affect read quality:

- The length of the read
 - Sequencing errors
 - Quality typically decreases toward the end of the read
- Upstream sample processing
 - Low quality DNA or very small amounts

Bioinformatic Pipeline:

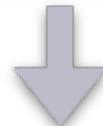
- Well-designed pipelines have the advantage of producing the correct input files for the next step in the pipeline
 - Files are generated according to a common standard format
- Pipelines can be fully automated once the bugs are worked out
- The backbone of the pipeline we will use can serve as a starting point for other bioinformatics pipelines
- There are other tools available for checking quality and performing alignments
 - Keep in mind you should always assess the suitability of each tool for your analysis



Bioinformatic Pipeline:

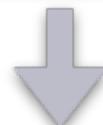
- Well-designed pipelines have the advantage of producing the correct input files for the next step in the pipeline
 - Files are generated according to a common standard format
- Pipelines can be fully automated once the bugs are worked out
- The backbone of the pipeline we will use can serve as a starting point for other bioinformatics pipelines
- There are other tools available for checking quality and performing alignments
 - Keep in mind you should always assess the suitability of each tool for your analysis

Sequence reads



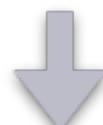
FASTQ

Quality control



FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

Pipelines can be highly versatile

- Our example is WGS
- Steps (especially quality and alignment) are similar for RNAseq

Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample

Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample
- We are using the European Nucleotide Archive to access our data.
 - The ENA "provides a comprehensive record of the world's nucleotide sequencing information, covering raw sequencing data, sequence assembly information and functional annotation."
 - Data are provided in FASTQ format

Download the data:

```
mkdir -p ~/dc_workshop/data/untrimmed_fastq/
cd ~/dc_workshop/data/untrimmed_fastq/
```

Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample
- We are using the European Nucleotide Archive to access our data.
 - The ENA "provides a comprehensive record of the world's nucleotide sequencing information, covering raw sequencing data, sequence assembly information and functional annotation."
 - Data are provided in FASTQ format

Download the data:

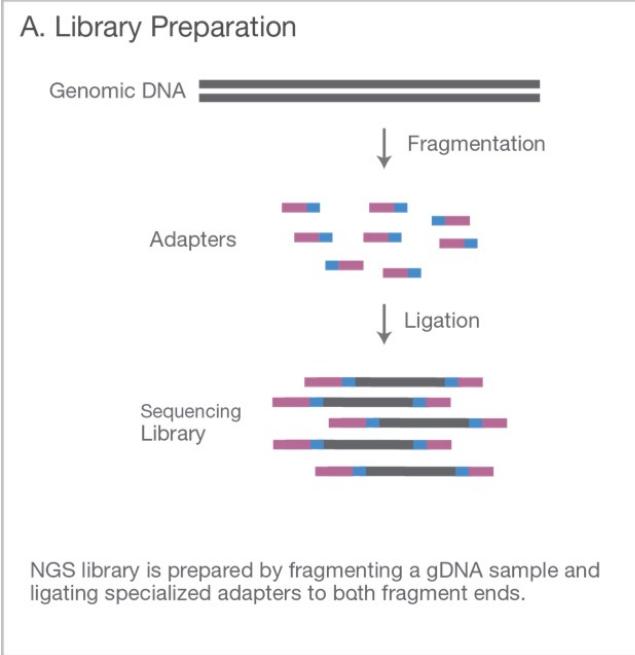
```
mkdir -p ~/dc_workshop/data/untrimmed_fastq/
cd ~/dc_workshop/data/untrimmed_fastq/

curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_2.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_2.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_2.fastq.gz
```

Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample

A. Sequencing starts with library preparation. Genomic DNA (or cDNA if starting from RNA) is fragmented size-selected. 5' and 3' adapters are ligated. (Adapters can include sample-specific indices).

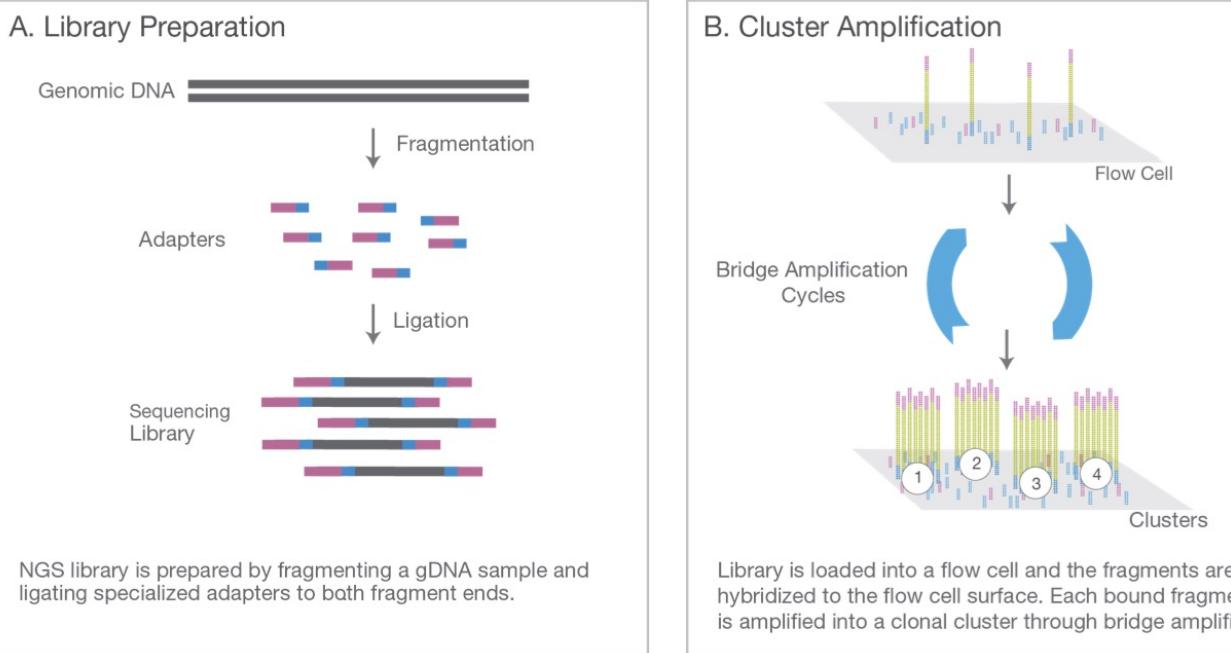


Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample

A. Sequencing starts with library preparation. Genomic DNA (or cDNA if starting from RNA) is fragmented size-selected. 5' and 3' adapters are ligated. (Adapters can include sample-specific indices).

B. Library is loaded into a flow cell and the fragments bind to short sequences that are complementary to the adapters. Bridge amplification generates clusters of primed templates across the flow cell.



Our data files:

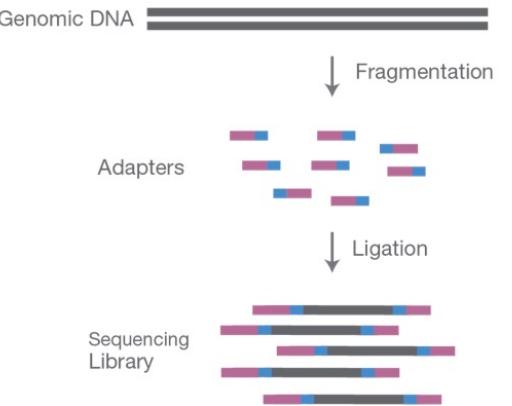
- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample

A. Sequencing starts with library preparation. Genomic DNA (or cDNA if starting from RNA) is fragmented size-selected. 5' and 3' adapters are ligated. (Adapters can include sample-specific indices).

B. Library is loaded into a flow cell and the fragments bind to short sequences that are complementary to the adapters. Bridge amplification generates clusters of primed templates across the flow cell.

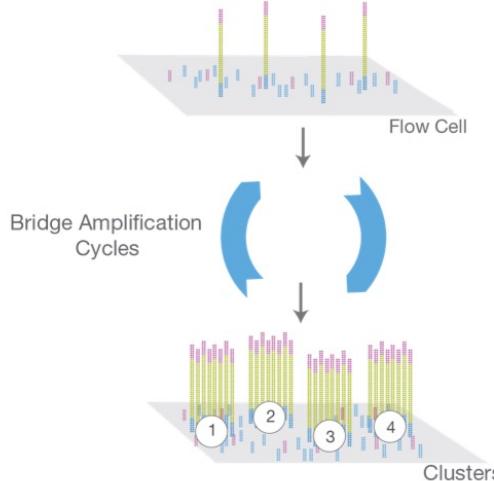
C. Sequencing by synthesis occurs as bases are added to the sequencing primer in a series of cycles. Bases are fluorescently labeled and when they are incorporated, they emit a base-specific wavelength that is decoded in an output file. One base is incorporated each cycle. The number of cycles determines the length of the read.

A. Library Preparation



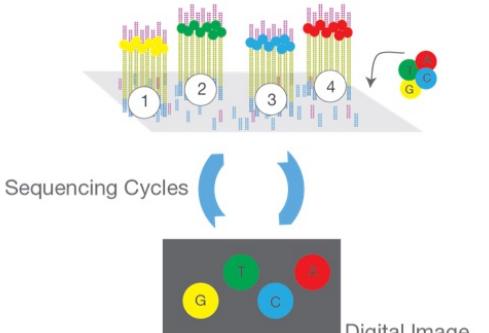
NGS library is prepared by fragmenting a gDNA sample and ligating specialized adapters to both fragment ends.

B. Cluster Amplification



Library is loaded into a flow cell and the fragments are hybridized to the flow cell surface. Each bound fragment is amplified into a clonal cluster through bridge amplification.

C. Sequencing



Cluster 1 > Read 1: GAGT...
Cluster 2 > Read 2: TTGA...
Cluster 3 > Read 3: CTAG...
Cluster 4 > Read 4: ATAC...
Text File

Sequencing reagents, including fluorescently labeled nucleotides, are added and the first base is incorporated. The flow cell is imaged and the emission from each cluster is recorded. The emission wavelength and intensity are used to identify the base. This cycle is repeated "n" times to create a read length of "n" bases.

Our data files:

- We will work with 3 samples from the LTEE *E. coli* experiment
 - Generations 5000, 15000, and 50000
 - Expect that the population has evolved over this time
- Data are paired-end, so there will be 2 files for every sample
- We are using the European Nucleotide Archive to access our data.
 - The ENA "provides a comprehensive record of the world's nucleotide sequencing information, covering raw sequencing data, sequence assembly information and functional annotation."
 - Data are provided in FASTQ format

Download the data:

```
mkdir -p ~/dc_workshop/data/untrimmed_fastq/
cd ~/dc_workshop/data/untrimmed_fastq/

curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_2.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_2.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_1.fastq.gz
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_2.fastq.gz
```

Quality Control:

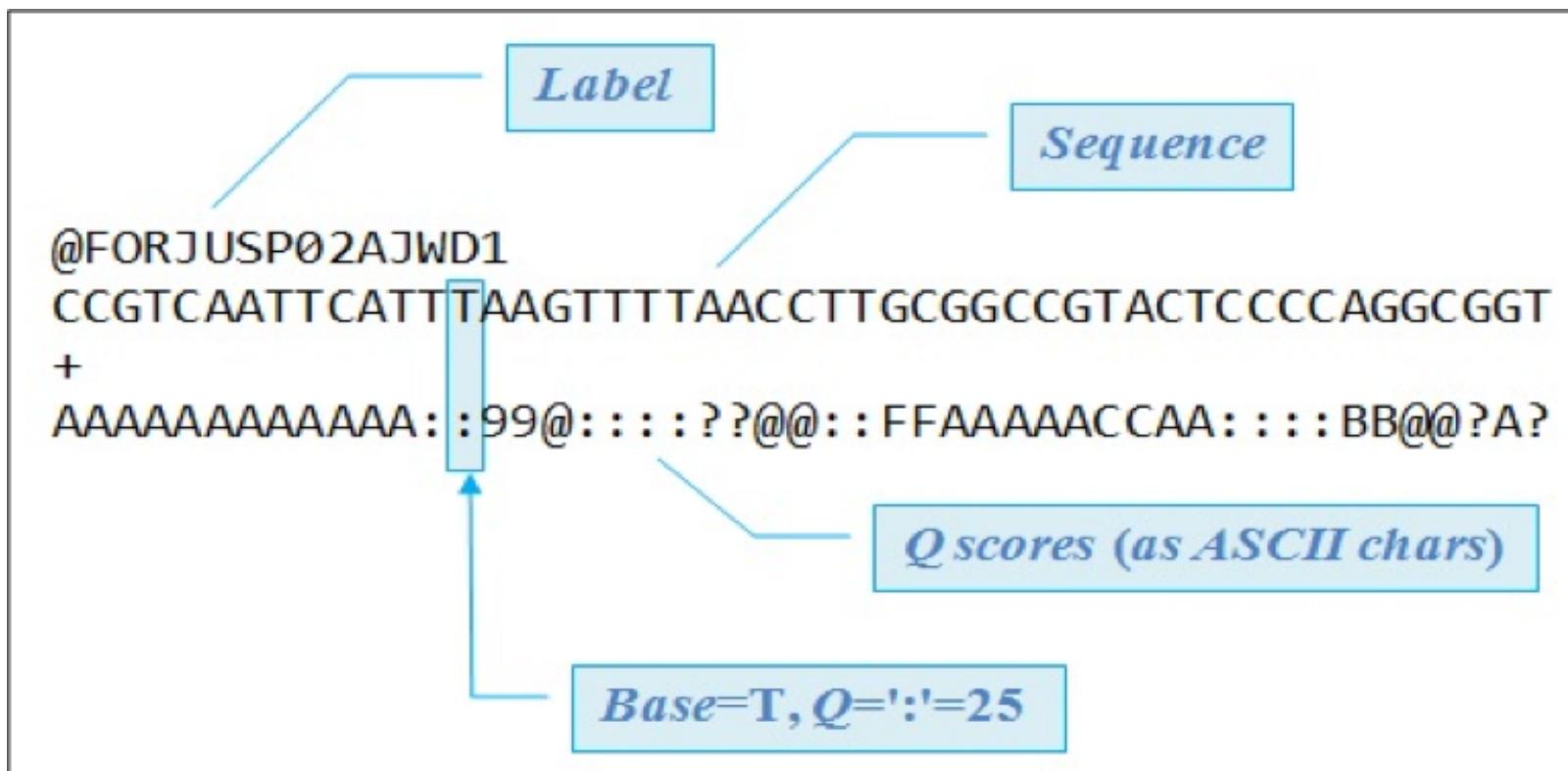
- FASTQ files have a specific format:

LINE 1 @ followed by information about the read

LINE 2 DNA sequence

LINE 3 + and sometimes same info as line 1

LINE 4 Quality scores for each base



Quality Control:

- FASTQ files have a specific format:

LINE 1 @ followed by information about the read

LINE 2 DNA sequence

LINE 3 + and sometimes same info as line 1

LINE 4 Quality scores for each base

Output

Quality encoding: !"#\$%&'()*+,-./0123456789:;=>?@ABCDEFGHIJ

Quality score: 01.....11.....21.....31.....41

Quality Control:

- FASTQ files have a specific format:

LINE 1 @ followed by information about the read

LINE 2 DNA sequence

LINE 3 + and sometimes same info as line 1

LINE 4 Quality scores for each base

Output

Quality encoding: !"#\$%&'()*+,-./0123456789:;=>?@ABCDEFGHIJ

Quality score: 01.....11.....21.....31.....41

Quality score to accuracy probability:

$$Q = -10 \log_{10} P$$

Q Search this file...

1	Phred Quality Score	Incorrect base call prob	Base call accuracy
2	10	1 in 10	90%
3	20	1 in 100	99%
4	30	1 in 1000	99.9%
5	40	1 in 10000	99.99%

phred_quality_score_1.csv hosted with ❤ by GitHub [view raw](#)

Quality Control:

- The tool we will use to interpret and report the quality scores in a more human-readable output is FASTQC
- Verify that you have FASTQC installed

SYNOPSIS

```
fastqc seqfile1 seqfile2 .. seqfileN
```

Basic Syntax

```
fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
       [-c contaminant file] seqfile1 .. seqfileN
```

Syntax with options

DESCRIPTION

FastQC reads a set of sequence files and produces from each one a quality control report consisting of a number of different modules, each one of which will help to identify a different potential type of problem in your data.

If no files to process are specified on the command line then the program will start as an interactive graphical application. If files are provided on the command line then the program will run with no user interaction required. In this mode it is suitable for inclusion into a standardised analysis pipeline.

The options for the program are as follows:

-h --help Print this help file and exit

-v --version Print the version of the program and exit

-o --outdir Create all output files in the specified output directory. Please note that this directory must exist as the program will not create it. If this option is not set then the output file for each sequence file is created in the same directory as the sequence file which was processed.

Quality Control:

- The tool we will use to interpret and report the quality scores in a more human-readable output is FASTQC
- Verify that you have FASTQC installed
- Assessing quality with FASTQC:
 - Gives a quick impression of problems with data so that you can take them into consideration in subsequent steps
 - Summarizes quality information by SAMPLE rather than by READ
 - We will access the summary using an output of FASTQC

Downloading data from your AMI:

Windows:

1. If you haven't already done so, download pscp from <http://the.earth.li/~sgtatham/putty/latest/x86/pscp.exe>
2. Make sure that the PSCP program is in your downloads folder
3. Open PowerShell:
 - Start menu → Search 'cmd' → Open PowerShell
4. Change directory to the location you want to store the file (e.g. Downloads, Desktop)
 - cd DIRECTORY
5. Download the fastqc html files:

```
C:\USER\DIRECTORY> pscp.exe dcuser@ecX-XX-XX-XXX-XX.compute-1.amazonaws.com:~/dc_workshop/results/fastqc_untrimmed_reads/*.html .
```

Replace with your information

Mac:

1. Open a new tab on your terminal (command t)
2. Confirm that you are using the bash interactive shell:
`ps -p $$`
3. Change directory to the location you want to store the file (e.g. Downloads, Desktop)
 - cd DIRECTORY
4. Download the fastqc html files:

```
scp dcuser@ecX-XX-XX-XXX-XX.compute-1.amazonaws.com:~/dc_workshop/results/fastqc_untrimmed_reads/*.html .
```

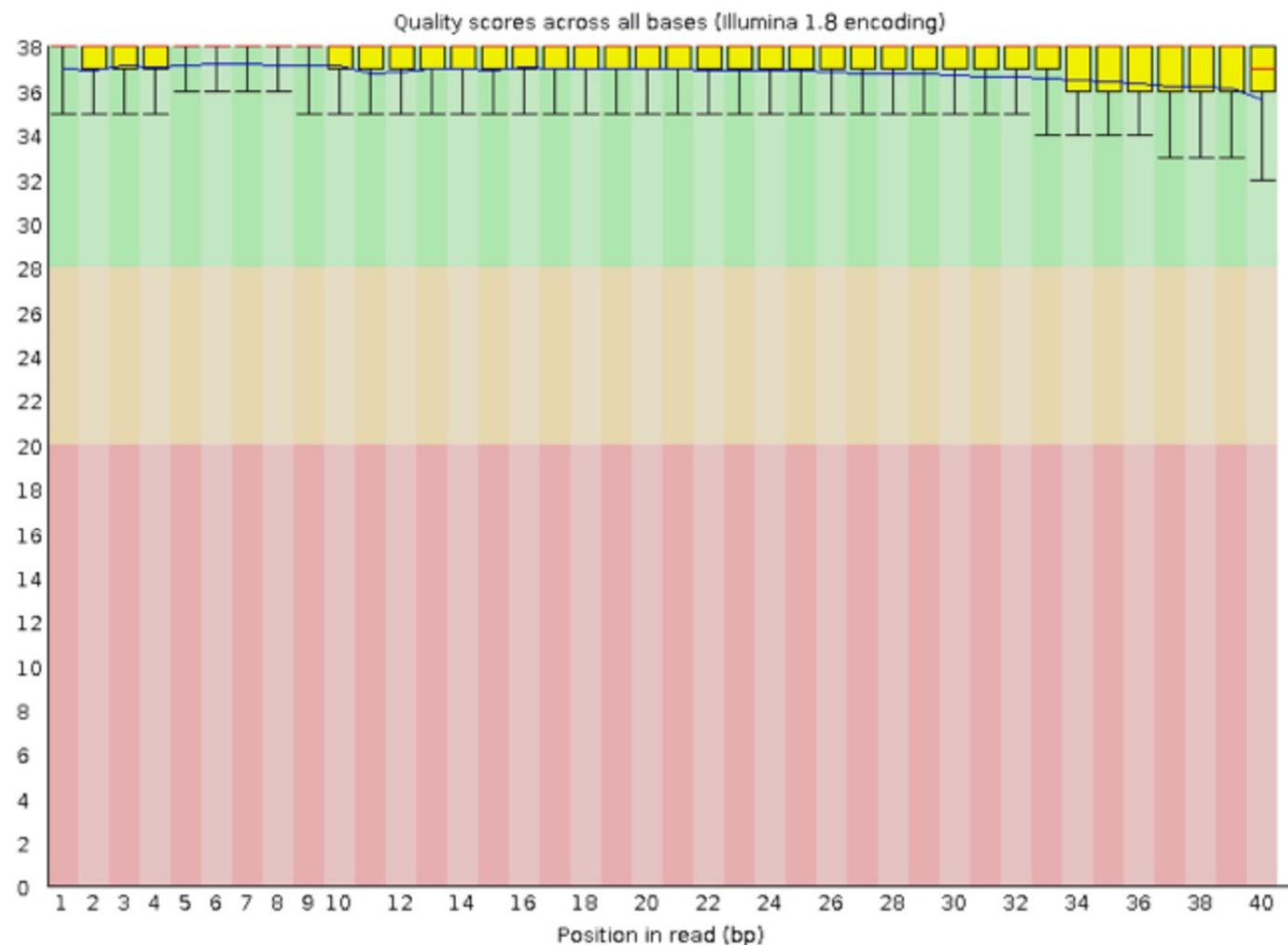
Replace with your information

Quality Control:

- The tool we will use to interpret and report the quality scores in a more human-readable output is FASTQC
- Verify that you have FASTQC installed
- Assessing quality with FASTQC:
 - Gives a quick impression of problems with data so that you can take them into consideration in subsequent steps
 - Summarizes quality information by SAMPLE rather than by READ
 - We will access the summary using an output FASTQC

High Quality Sample:

- 40 bp read
- Each position is summarized with a box and whisker plot
 - 50% of reads have a quality score that falls within the range of the yellow box at that position

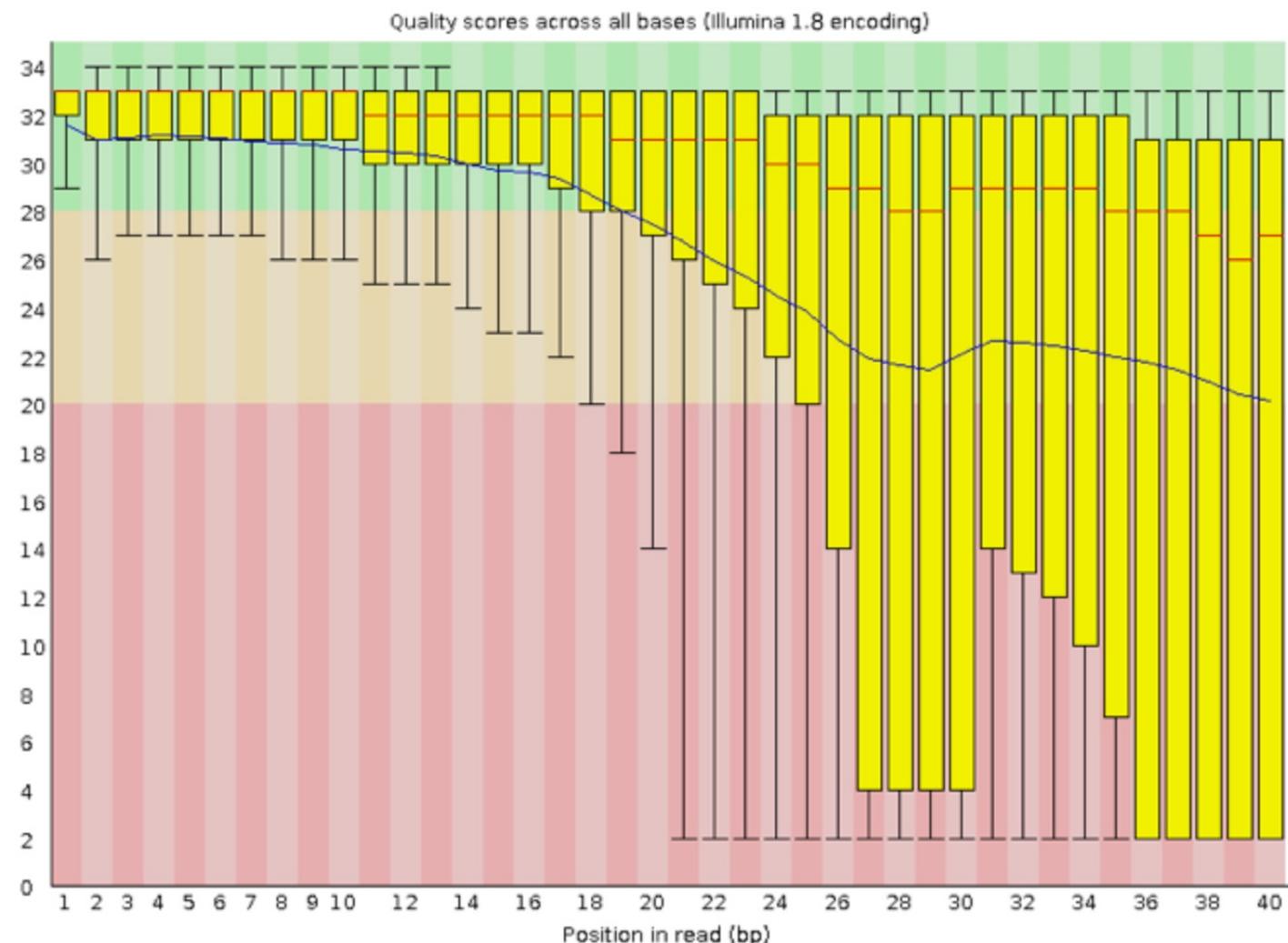


Quality Control:

- The tool we will use to interpret and report the quality scores in a more human-readable output is FASTQC
- Verify that you have FASTQC installed
- Assessing quality with FASTQC:
 - Gives a quick impression of problems with data so that you can take them into consideration in subsequent steps
 - Summarizes quality information by SAMPLE rather than by READ
 - We will access the summary using an output FASTQC

Low Quality Sample:

- 40 bp read
- Quality at each position is more variable
- Quality scores drop very low, particularly at the end of reads.



For loop syntax:

```
for OBJECT in LIST_OF_OBJECTS
do
  OPERATION $OBJECT
done
```

- For loops use an indexing system to move through each **element** in a **list**

For loop syntax:

```
for OBJECT in LIST_OF_OBJECTS
do
  OPERATION $OBJECT
done
```

- For loops use an indexing system to move through each **element** in a **list**

For loop syntax:

```
for filename in *.zip
do
  unzip $filename
done
```

- We have 6 .zip files.
- The first time the loop iterates, the variable **filename** is assigned as **SRR2584863_1_fastqc.zip** and the file is unzipped
- For the second iteration, the variable **filename** is assigned as **SRR2584863_2_fastqc.zip**

Building our Pipeline:

Sequence reads



FASTQ

Quality control



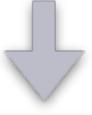
FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

```
# STEP 1: FASTQC raw data
fastqc *.fastq*                                # Run FASTQC

for filename in *.zip                         # unzip fastqc .zip files
do
unzip $filename
done

cat */summary.txt > fastqc_summaries.txt      # obtain fastqc summary
grep FAIL fastqc_summaries.txt > fastqc_FAIL.txt # identify problem samples
```

Goals for today:

- Background and metadata
- Assessing sequencing read quality
- **Trimming and filtering reads based on read quality**
- Variant calling workflow
- Automating a variant calling workflow

Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Objectives:

- Remove sequence data that doesn't meet quality standards
- Select and set multiple options for command-line bioinformatics tools
- Write for loops with two variables

Cleaning reads:

- FASTQC revealed there is variability in sequencing quality
- We want to base downstream analyses on data that has high quality and high confidence in accuracy
 - Low confidence base calls can inflate false positive and negatives
 - Variant calling is sensitive to low confidence in base calls.

Cleaning reads:

- FASTQC revealed there is variability in sequencing quality
- We want to base downstream analyses on data that has high quality and high confidence in accuracy
 - Low confidence base calls can inflate false positive and negatives
 - Variant calling is sensitive to low confidence in base calls.

Specify whether data are
Paired-end or Single-end reads

```
$ trimomatic
Usage:
    PE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary
<statsSummaryFile>] [-quiet] [-validatePairs] [-basein <inputBase> | <inputFile1> <inputFile2>] [-baseout
<outputBase> | <outputFile1P> <outputFile1U> <outputFile2P> <outputFile2U>] <trimmer1>...
    or:
    SE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary
<statsSummaryFile>] [-quiet] <inputFile> <outputFile> <trimmer1>...
    or:
    -version
```

Cleaning reads:

- FASTQC revealed there is variability in sequencing quality
- We want to base downstream analyses on data that has high quality and high confidence in accuracy
 - Low confidence base calls can inflate false positive and negatives
 - Variant calling is sensitive to low confidence in base calls.

Specify whether data are
Paired-end or Single-end reads

```
$ trimmomatic
Usage:
  PE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary
<statsSummaryFile>] [-quiet] [-validatePairs] [-basein <inputBase> | <inputFile1> <inputFile2>] [-baseout
<outputBase> | <outputFile1P> <outputFile1U> <outputFile2P> <outputFile2U>] <trimmer1>...
  or:
  SE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary
<statsSummaryFile>] [-quiet] <inputFile> <outputFile> <trimmer1>...
  or:
  -version
```

Specify the two files that have
the paired data for each sample

Name the output files. Two types:
Output file with surviving read pairs (1/2P)
Output file with orphaned reads (1/2/U)

Cleaning reads:

- FASTQC revealed there is variability in sequencing quality
- We want to base downstream analyses on data that has high quality and high confidence in accuracy
 - Low confidence base calls can inflate false positive and negatives
 - Variant calling is sensitive to low confidence in base calls.

step	meaning
ILLUMINACLIP	Perform adapter removal.
SLIDINGWINDOW	Perform sliding window trimming, cutting once the average quality within the window falls below a threshold.
LEADING	Cut bases off the start of a read, if below a threshold quality.
TRAILING	Cut bases off the end of a read, if below a threshold quality.
CROP	Cut the read to a specified length.
HEADCROP	Cut the specified number of bases from the start of the read.
MINLEN	Drop an entire read if it is below a specified length.
TOPHRED33	Convert quality scores to Phred-33.
TOPHRED64	Convert quality scores to Phred-64.

Long commands can be separated across lines with \

```
$ trimmomatic PE \n\n    -threads 4 \n    Input_1.fastq Input_2.fastq \n    Output_1.trimmed.fastq Output_1un.trimmed.fastq Output_2.trimmed.fastq Output_2un.trimmed.fastq \n    ILLUMINACLIP:SRR_adapters.fa \ # clip Illumina adapters from the input file \n    SLIDINGWINDOW:4:20 \ # use a sliding window size of 4 bases, remove bases \n                                # with phred score < 20
```

Long commands can be separated across lines with \

```
$ trimmomatic PE \
  -threads 4 \
  Input_1.fastq Input_2.fastq \
  Output_1.trimmed.fastq Output_1un.trimmed.fastq Output_2.trimmed.fastq Output_2un.trimmed.fastq \
  ILLUMINACLIP:SRR_adapters.fa \
  SLIDINGWINDOW:4:20
  # clip Illumina adapters from the input file
  # use a sliding window size of 4 bases, remove bases
  # with phred score < 20
```

```
trimmomatic PE \
> SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
> SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
> SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
> SLIDINGWINDOW:4:20 \
  # sliding window size = 4, remove bases with phred score < 20
> MINLEN:25 \
  # discard reads with < 25 bases after trimming
> ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
  # Usage <adapters.fa>:<seed matches>:
  # <palindrome clip threshold>:<simple clip threshold>
```

```
for infile in *_1.fastq.gz
do
  base=$(basename ${infile} _1.fastq.gz)
  trimmomatic PE ${infile} ${base}_2.fastq.gz \
    ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
    ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
    SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done
```

Building our Pipeline:

Sequence reads



FASTQ

Quality control



FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

```
# STEP 1: FASTQC raw data
fastqc *.fastq*                                # Run FASTQC

for filename in *.zip                         # unzip fastqc .zip files
do
unzip $filename
done

cat */summary.txt > fastqc_summaries.txt      # obtain fastqc summary
grep FAIL fastqc_summaries.txt > fastqc_FAIL.txt # identify problem samples

# STEP 2: TRIM and FILTER

for infile in *_1.fastq.gz                      # Run trimmomatic
do
base=$(basename ${infile} _1.fastq.gz)
trimmomatic PE ${infile} ${base}_2.fastq.gz \
${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done

# STEP 3: FASTQC filtered data

fastqc ~/dc_workshop/data/trimmed_fastq*.fastq*  # Run FASTQC
```

Building our Pipeline:

Sequence reads



FASTQ

Quality control



FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

```
# STEP 1: FASTQC raw data
fastqc *.fastq*                                # Run FASTQC

for filename in *.zip                         # unzip fastqc .zip files
do
unzip $filename
done

cat */summary.txt > fastqc_summaries.txt      # obtain fastqc summary
grep FAIL fastqc_summaries.txt > fastqc_FAIL.txt # identify problem samples

# STEP 2: TRIM and FILTER

for infile in *_1.fastq.gz                      # Run trimmomatic
do
base=$(basename ${infile} _1.fastq.gz)
trimmomatic PE ${infile} ${base}_2.fastq.gz \
${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done

# STEP 3: FASTQC filtered data

fastqc ~/dc_workshop/data/trimmed_fastq*.fastq*  # Run FASTQC
```

Questions?

Building our Pipeline:

Sequence reads



FASTQ

Quality control



FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

```
# STEP 1: FASTQC raw data
fastqc *.fastq*                                # Run FASTQC

for filename in *.zip                         # unzip fastqc .zip files
do
unzip $filename
done

cat */summary.txt > fastqc_summaries.txt      # obtain fastqc summary
grep FAIL fastqc_summaries.txt > fastqc_FAIL.txt # identify problem samples

# STEP 2: TRIM and FILTER

for infile in *_1.fastq.gz                      # Run trimmomatic
do
base=$(basename ${infile} _1.fastq.gz)
trimmomatic PE ${infile} ${base}_2.fastq.gz \
${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done

# STEP 3: FASTQC filtered data

fastqc ~/dc_workshop/data/trimmed_fastq*.fastq*  # Run FASTQC
```

Questions?

Take a break or complete Step 3

Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- **Variant calling workflow**
- Automating a variant calling workflow

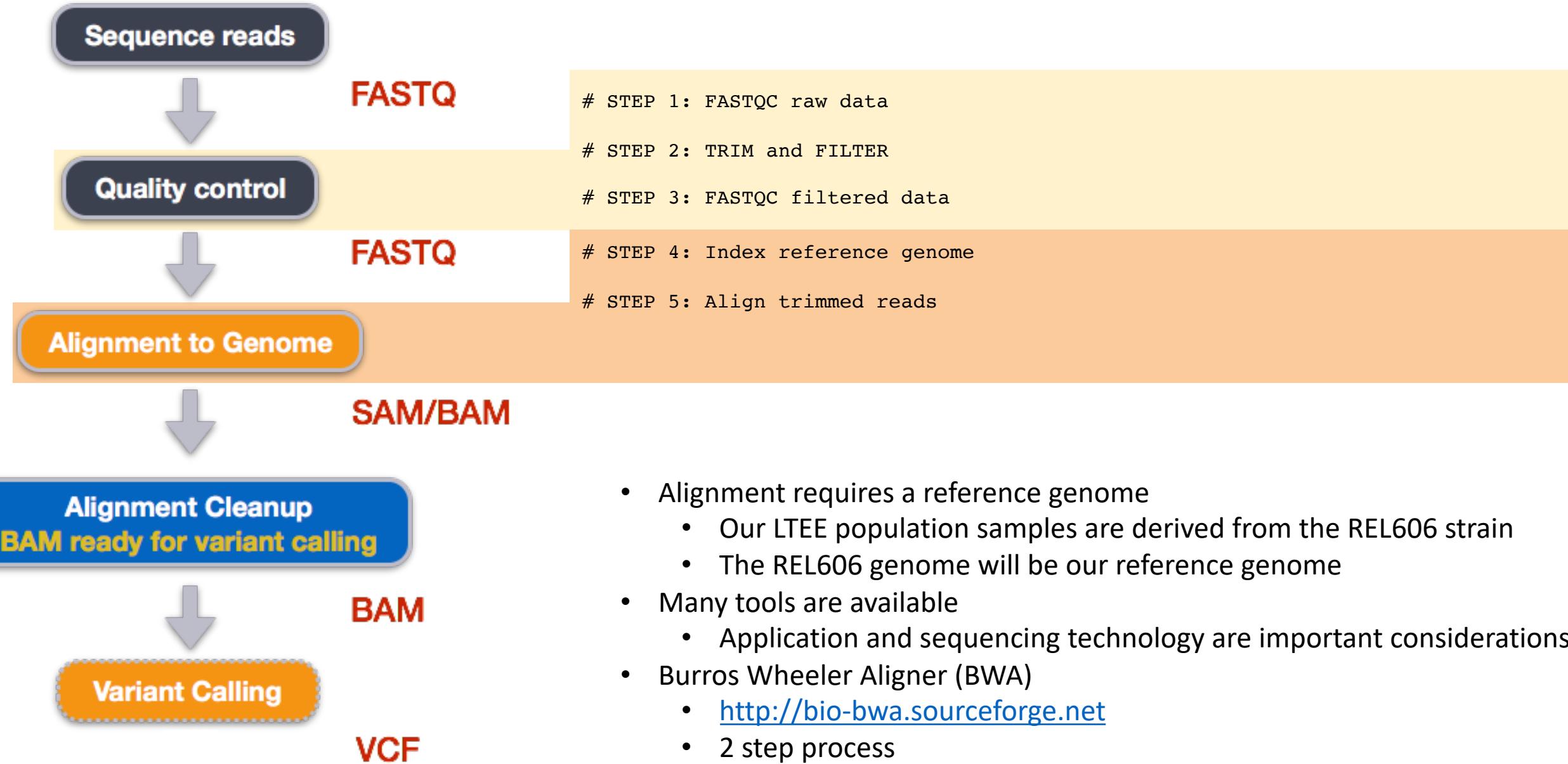
Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- **Variant calling workflow**
- Automating a variant calling workflow

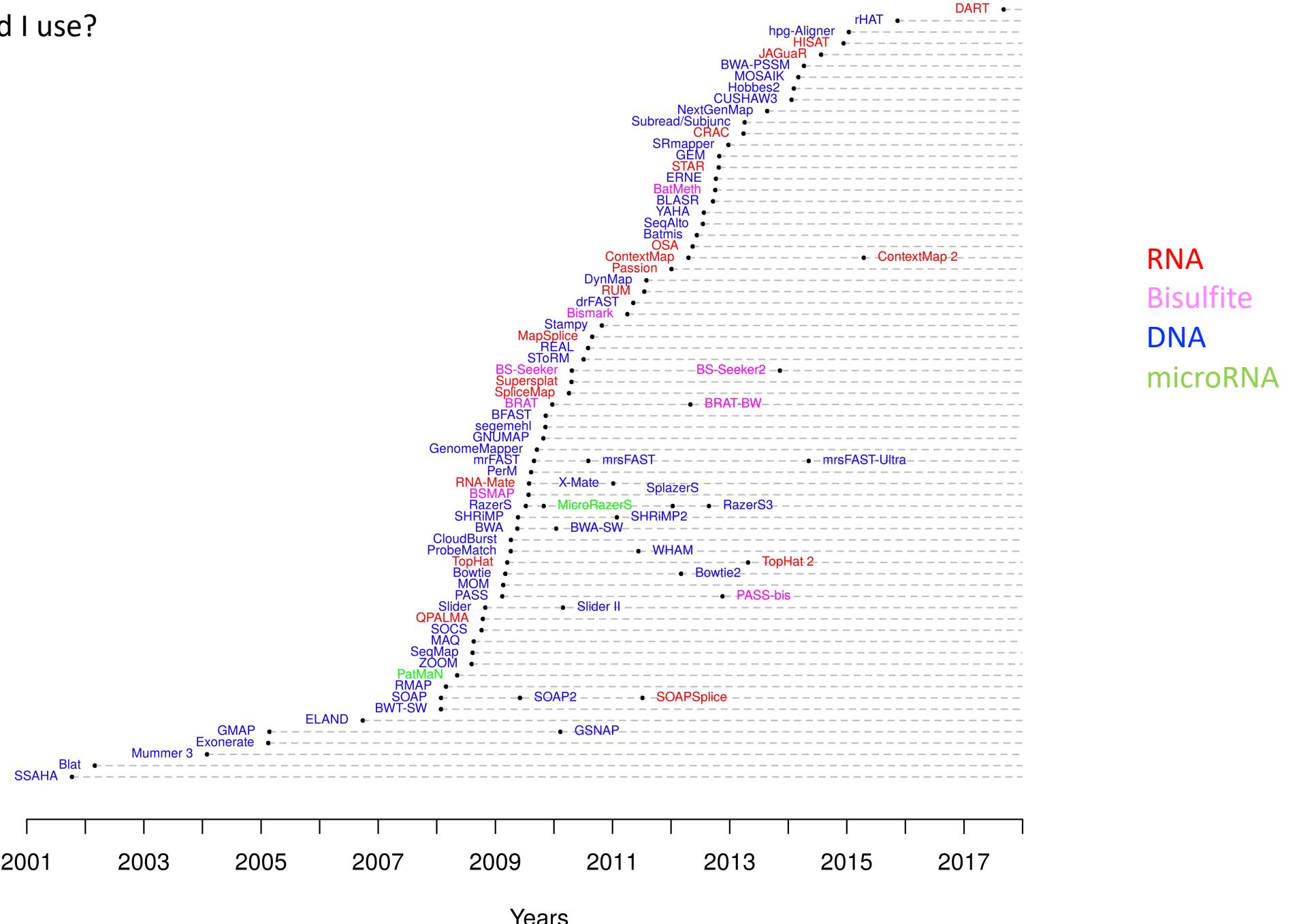
Objectives:

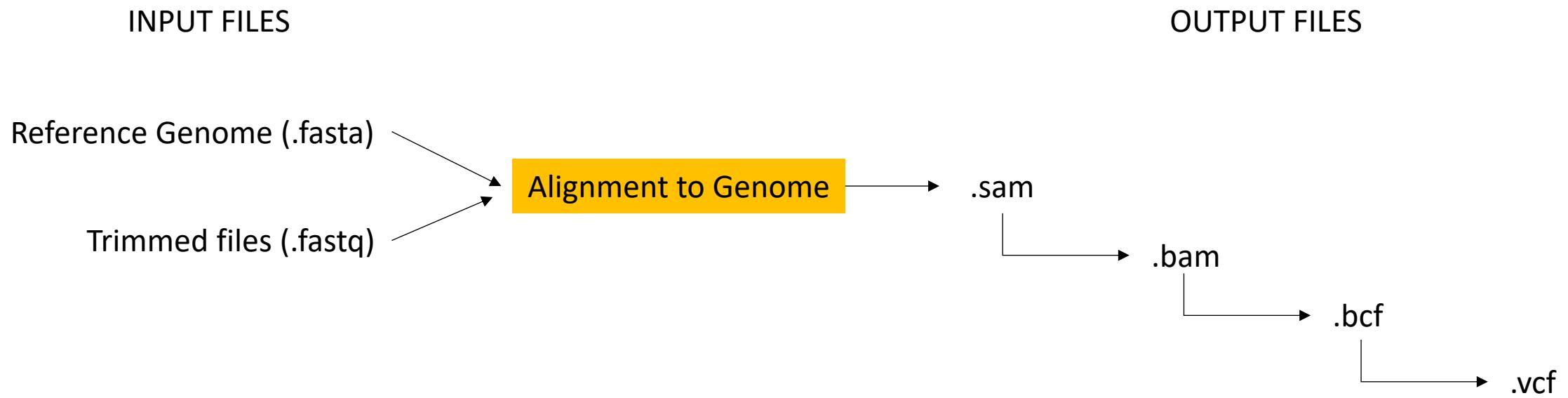
- Find sequence variants between samples and reference genome
- Describe types of data formats encountered during variant calling
- Use command line tools to perform variant calling

Building our Pipeline:



Which aligner should I use?





- To help with organization, we will make directories ahead of time to hold each file type.

Indexing the reference genome:

- Indexing allows the aligner to quickly find potential alignment sites for query sequences
- Indexing varies by aligner tool. This method is specific to BWA
- Indexing only needs to be run once
- Like an index in a book:
 - Indexing allows the aligner to narrow down the genome to a smaller region before trying to match reads to genome



- Alignment is about fitting individual pieces (reads) into the correct part of the puzzle
- The human genome project gave us the picture on the box cover (the reference genome)
- Imperfections in how the pieces fit can indicate changes to a copy of the picture

Reference:

AGCCTGAGACC GTAAAAAA **A**GTCAAG

||||| ||||| ||||| |||||

GAGACC GTAAAAAA **C**GT C

A variant!

A read sequence:

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

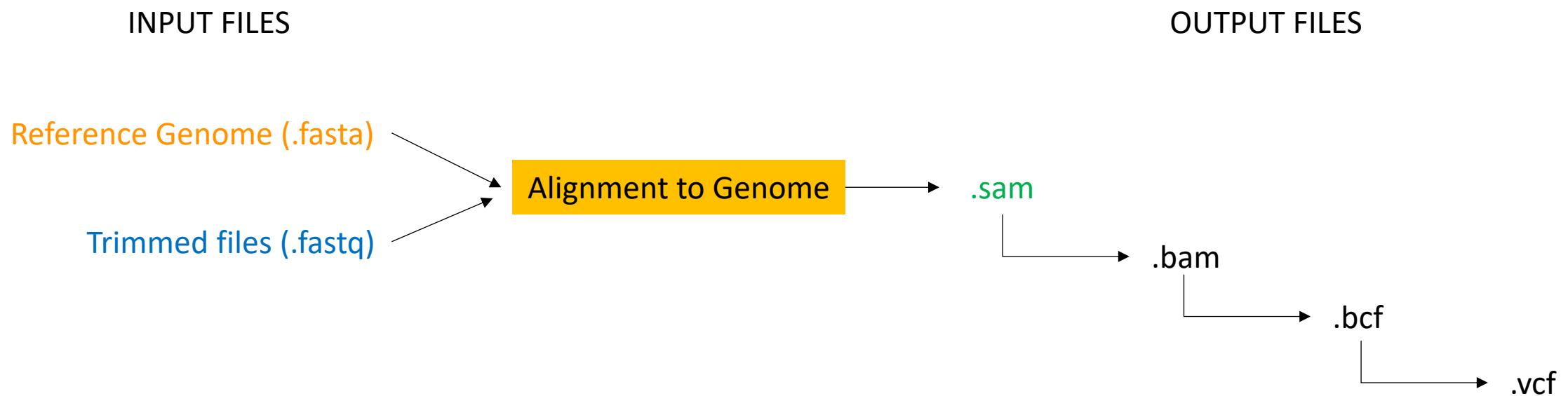
.bam

.bcf

.vcf

OUTPUT FILES

```
$ bwa mem ref_genome.fasta input_file_R1.fastq input_file_R2.fastq > output.sam
```

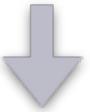


```
$ bwa mem ref_genome.fasta input_file_R1.fastq input_file_R2.fastq > output.sam
```

- Our input and output files are organized across directories, so we will include the paths in our command

Building our Pipeline:

Sequence reads



FASTQ

Quality control



```
# STEP 1: FASTQC raw data  
# STEP 2: TRIM and FILTER  
# STEP 3: FASTQC filtered data
```

FASTQ

Alignment to Genome



SAM/BAM

```
# STEP 4: Index reference genome  
bwa index PATH/ref_genome/ref_genome.fasta  
# STEP 5: Align trimmed reads  
bwa mem ref_genome.fasta input_file_R1.fastq input_file_R2.fastq > output.sam
```

Alignment Cleanup

BAM ready for variant calling



BAM

Variant Calling

VCF

Building our Pipeline:

Sequence reads



FASTQ

```
# STEP 1: FASTQC raw data  
# STEP 2: TRIM and FILTER  
# STEP 3: FASTQC filtered data
```

Quality control



FASTQ

```
# STEP 4: Index reference genome  
# STEP 5: Align trimmed reads
```

Alignment to Genome



SAM/BAM

```
# STEP 6: Convert from SAM to BAM  
# STEP 7: Sort BAM file
```

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

.bam

.bcf

.vcf

OUTPUT FILES

```
$ samtools view -S -b PATH/sam/output.sam > PATH/bam/output.bam
```

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

.bam

.bcf

.vcf

OUTPUT FILES

```
samtools sort -o PATH/bam/output.sorted.bam PATH/bam/output.bam
```

Building our Pipeline:

Sequence reads



FASTQ

```
# STEP 1: FASTQC raw data  
# STEP 2: TRIM and FILTER  
# STEP 3: FASTQC filtered data
```

Quality control



FASTQ

```
# STEP 4: Index reference genome  
# STEP 5: Align trimmed reads
```

Alignment to Genome



SAM/BAM

```
# STEP 6: Convert from SAM to BAM  
samtools view -S -b PATH/sam/output.sam > PATH/bam/output.bam  
  
# STEP 7: Sort BAM file  
samtools sort -o PATH/bam/output.sorted.bam PATH/bam/output.bam
```

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

Variant Calling:

- A variant is a different nucleotide at a given position compared to the reference
- Typically accompanied by an estimate of variant frequency and confidence

Reference:

AGCCTGAGACCGTAAAAAA**AG**TCAAG

||||||||||||||||||

GAGACCGTAAAAAA**C**GT

A read sequence:

A variant!

Building our Pipeline:

Sequence reads



FASTQ

Quality control

```
# STEP 1: FASTQC raw data  
# STEP 2: TRIM and FILTER  
# STEP 3: FASTQC filtered data
```



FASTQ

```
# STEP 4: Index reference genome  
# STEP 5: Align trimmed reads
```

Alignment to Genome



SAM/BAM

```
# STEP 6: Convert from SAM to BAM  
# STEP 7: Sort BAM file
```

Alignment Cleanup
BAM ready for variant calling



BAM

```
# STEP 8: Calculate read coverage of positions in genome  
# STEP 9: Detect SNPs  
# STEP 10: Filter and report SNPs in VCF
```

Variant Calling

VCF

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

.bam

.bcf

.vcf

OUTPUT FILES

```
bcftools mpileup -O b -o PATH/bcf/SAMPLE_raw.bcf  
-f PATH/ref_genome/ref_genome.fasta PATH/bam/output.sorted.bam
```

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

.bam

.bcf

.vcf

OUTPUT FILES

```
bcftools call --ploidy 1 -m -v -o PATH/bcf/SAMPLE_variants.vcf PATH/bcf/SAMPLE_raw.bcf
```

INPUT FILES

Reference Genome (.fasta)

Trimmed files (.fastq)

Alignment to Genome

.sam

.bam

.bcf

.vcf

OUTPUT FILES

```
vcfutils.pl varFilter PATH/bcf/SAMPLE_variants.vcf > PATH/vcf/SAMPLE_final_variants.vcf
```

VCF decoding:

Contig location of variant

Position of variant

. until annotated

Reference genotype

Sample genotype

Phred-scaled probability that the observed variant exists

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	results/bam/SRR2584866.aligned.sorted.bam	GT:PL	GQ
CP000819.1	1521	.	C	T	207	.	DP=9;VDB=0.993024;SGB=-0.662043;MQSB=0.974597;MQ0F=0;AC=1;AN=1;DP4=0,0,4,5;MQ=60			GT:PL	1:237,0
CP000819.1	1612	.	A	G	225	.	DP=13;VDB=0.52194;SGB=-0.676189;MQSB=0.950952;MQ0F=0;AC=1;AN=1;DP4=0,0,6,5;MQ=60			GT:PL	1:255,0
CP000819.1	9092	.	A	G	225	.	DP=14;VDB=0.717543;SGB=-0.670168;MQSB=0.916482;MQ0F=0;AC=1;AN=1;DP4=0,0,7,3;MQ=60			GT:PL	1:255,0
CP000819.1	9972	.	T	G	214	.	DP=10;VDB=0.022095;SGB=-0.670168;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,2,8;MQ=60			GT:PL	1:244,0
CP000819.1	10563	.	G	A	225	.	DP=11;VDB=0.958658;SGB=-0.670168;MQSB=0.952347;MQ0F=0;AC=1;AN=1;DP4=0,0,5,5;MQ=60			GT:PL	1:255,0
CP000819.1	22257	.	C	T	127	.	DP=5;VDB=0.0765947;SGB=-0.590765;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,2,3;MQ=60			GT:PL	1:157,0
CP000819.1	38971	.	A	G	225	.	DP=14;VDB=0.872139;SGB=-0.680642;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,4,8;MQ=60			GT:PL	1:255,0
CP000819.1	42306	.	A	G	225	.	DP=15;VDB=0.969686;SGB=-0.686358;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,5,9;MQ=60			GT:PL	1:255,0
CP000819.1	45277	.	A	G	225	.	DP=15;VDB=0.470998;SGB=-0.680642;MQSB=0.95494;MQ0F=0;AC=1;AN=1;DP4=0,0,7,5;MQ=60			GT:PL	1:255,0
CP000819.1	56613	.	C	G	183	.	DP=12;VDB=0.879703;SGB=-0.676189;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,8,3;MQ=60			GT:PL	1:213,0
CP000819.1	62118	.	A	G	225	.	DP=19;VDB=0.414981;SGB=-0.691153;MQSB=0.906029;MQ0F=0;AC=1;AN=1;DP4=0,0,8,10;MQ=59			GT:PL	1:255,0
CP000819.1	64042	.	G	A	225	.	DP=18;VDB=0.451328;SGB=-0.689466;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,7,9;MQ=60			GT:PL	1:255,0
CP000819.1	78808	.	C	T	225	.	DP=23;VDB=0.885435;SGB=-0.691153;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,0,13,5;MQ=60			GT:PL	1:255,0

Depth per allele by sample and coverage

GT:PL:GQ = Genotype, likelihoods of
genotypes, Phred-scaled confidence of
genotype

Building our Pipeline:

Sequence reads



FASTQ

```
# STEP 1: FASTQC raw data  
# STEP 2: TRIM and FILTER  
# STEP 3: FASTQC filtered data
```

Quality control



FASTQ

```
# STEP 4: Index reference genome  
# STEP 5: Align trimmed reads
```

Alignment to Genome



SAM/BAM

```
# STEP 6: Convert from SAM to BAM  
# STEP 7: Sort BAM file
```

Alignment Cleanup
BAM ready for variant calling



BAM

```
# STEP 8: Calculate read coverage of positions in genome  
bcftools mpileup -O b -o PATH/SAMPLE_raw.bcf -f PATH/ref_genome.fasta PATH/output.sorted.bam
```

Variant Calling

VCF

```
# STEP 7: Detect SNPs  
bcftools call --ploidy 1 -m -v -o PATH/bcf/SAMPLE_variants.vcf PATH/bcf/SAMPLE_raw.bcf  
# STEP 9: Filter and report SNPs in VFC  
vcfutils.pl varFilter PATH/bcf/SAMPLE_variants.vcf > PATH/vcf/SAMPLE_final_variants.vcf
```


Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- **Automating a variant calling workflow**

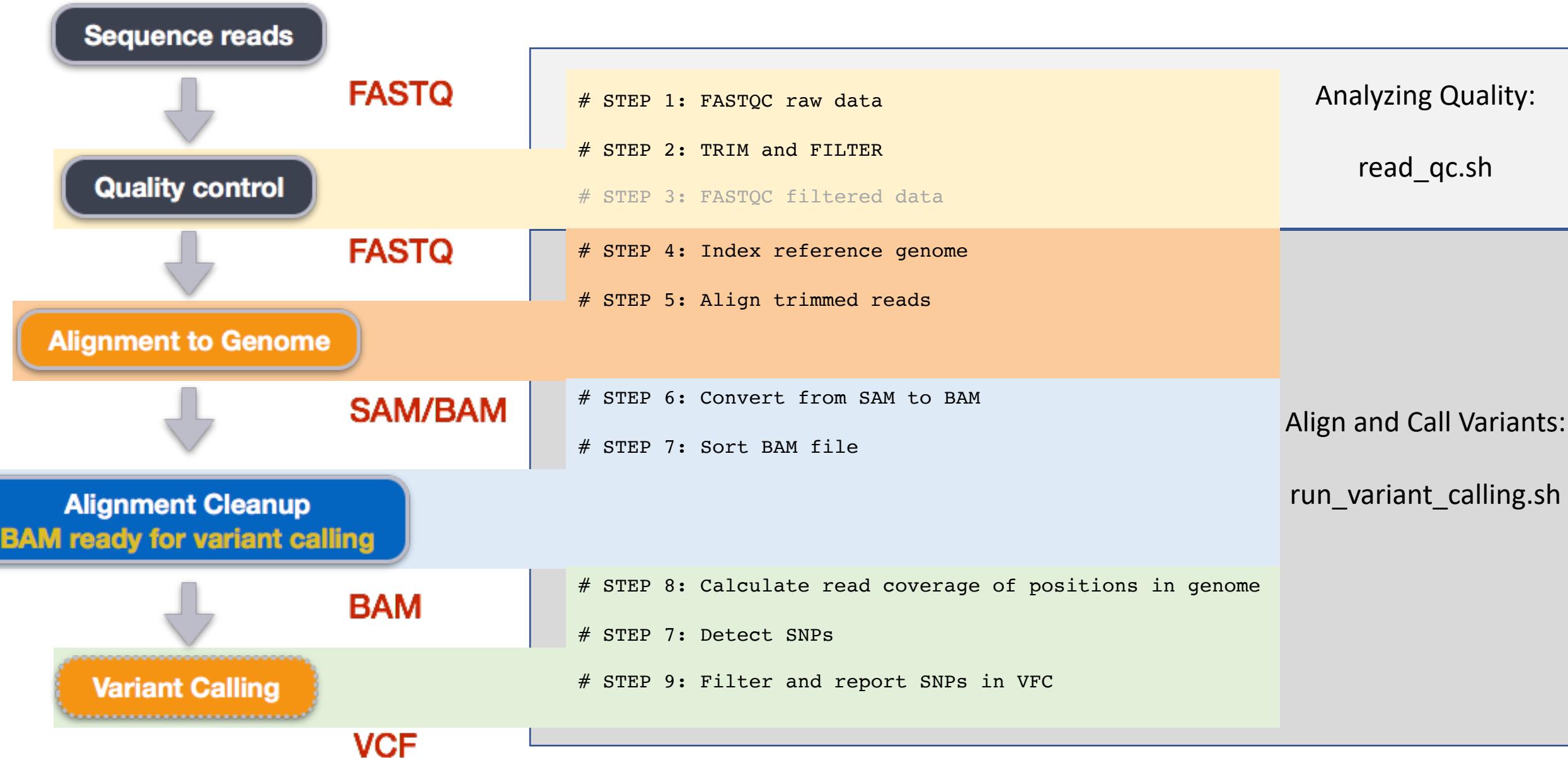
Goals for today:

- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

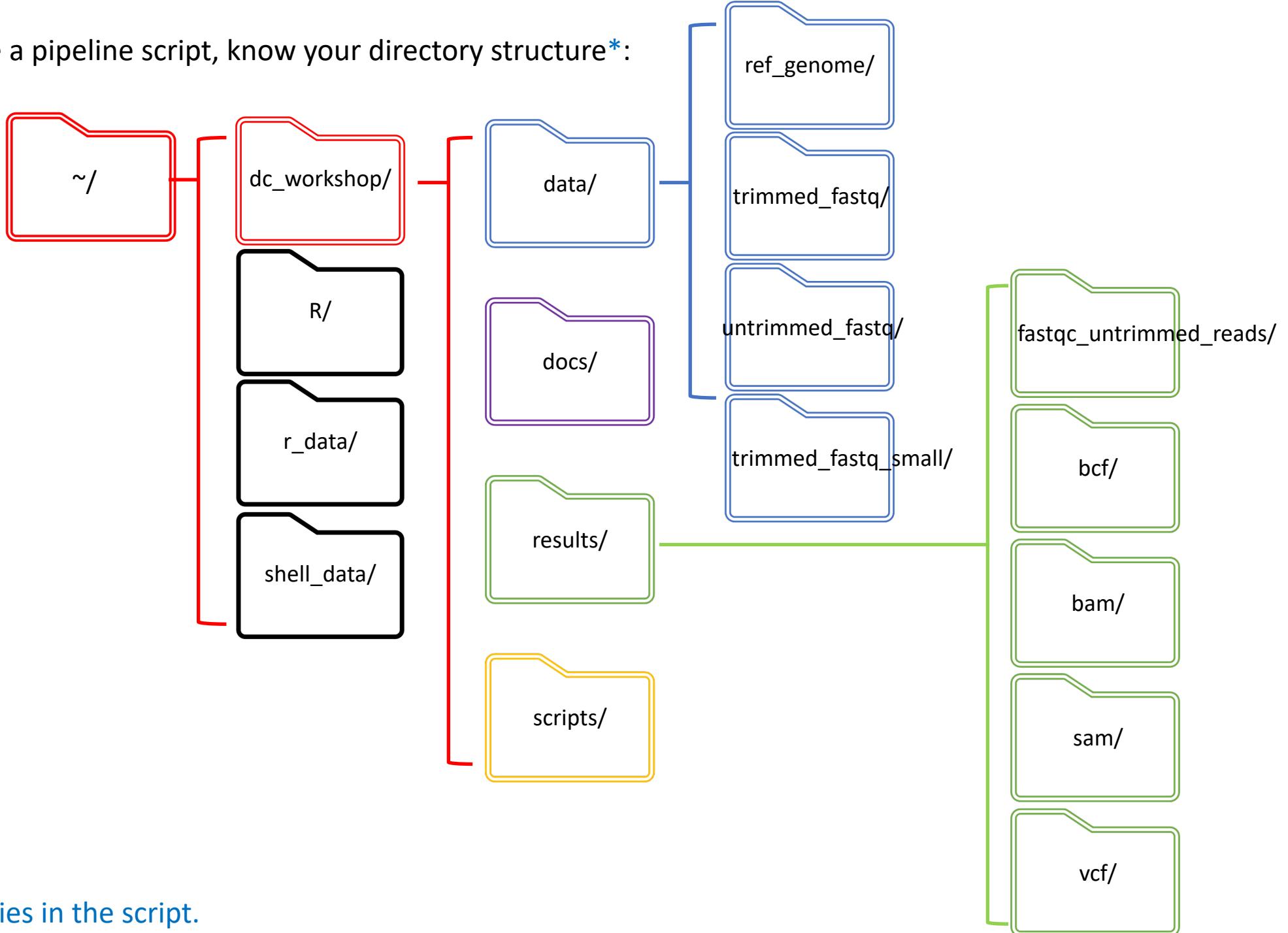
Objectives:

- Make a more efficient and less error-prone pipeline
- Write a shell script with multiple variables
- Incorporate a for loop into a shell script

Automating our Pipeline:



BEFORE you write a pipeline script, know your directory structure*:



*or create directories in the script.

BEFORE you write a pipeline script, test elements of your script:

Unzipping .zip files:

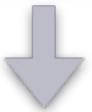
```
for filename in *.zip
do
unzip $filename
done
```

Running Trimmomatic:

```
for infile in *_1.fastq.gz
do
base=$(basename ${infile} _1.fastq.gz)
trimmmatic PE ${infile} ${base}_2.fastq.gz \
    ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
    ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
    SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done
```

Automating our Pipeline:

Sequence reads



FASTQ

Quality control



FASTQ

Alignment to Genome



SAM/BAM

Alignment Cleanup
BAM ready for variant calling



BAM

Variant Calling

VCF

```
# STEP 1: FASTQC raw data
```

```
# STEP 2: TRIM and FILTER
```

```
# STEP 3: FASTQC filtered data
```

```
# STEP 4: Index reference genome
```

```
# STEP 5: Align trimmed reads
```

```
# STEP 6: Convert from SAM to BAM
```

```
# STEP 7: Sort BAM file
```

```
# STEP 8: Calculate read coverage of positions in genome
```

```
# STEP 9: Detect SNPs
```

```
# STEP 10: Filter and report SNPs in VCF
```

Analyzing Quality:
read_qc.sh

Align and Call Variants:
run_variant_calling.sh

```

set -e
cd ~/dc_workshop/results

genome=~/dc_workshop/data/ref_genome/ecoli_rel606.fasta

bwa index $genome

mkdir -p sam bam bcf vcf

for fq1 in ~/dc_workshop/data/trimmed_fastq_small/*_1.trim.sub.fastq
do
echo "working with file $fq1"

base=$(basename $fq1 _1.trim.sub.fastq)
echo "base name is $base"

fq1=~/dc_workshop/data/trimmed_fastq_small/${base}_1.trim.sub.fastq
fq2=~/dc_workshop/data/trimmed_fastq_small/${base}_2.trim.sub.fastq
sam=~/dc_workshop/results/sam/${base}.aligned.sam
bam=~/dc_workshop/results/bam/${base}.aligned.bam
sorted_bam=~/dc_workshop/results/bam/${base}.aligned.sorted.bam
raw_bcf=~/dc_workshop/results/bcf/${base}_raw.bcf
variants=~/dc_workshop/results/bcf/${base}_variants.vcf
final_variants=~/dc_workshop/results/vcf/${base}_final_variants.vcf

bwa mem $genome $fq1 $fq2 > $sam
samtools view -S -b $sam > $bam
samtools sort -o $sorted_bam $bam
samtools index $sorted_bam
bcftools mpileup -O b -o $raw_bcf -f $genome $sorted_bam
bcftools call --ploidy 1 -m -v -o $variants $raw_bcf
vcfutils.pl varFilter $variants > $final_variants

done

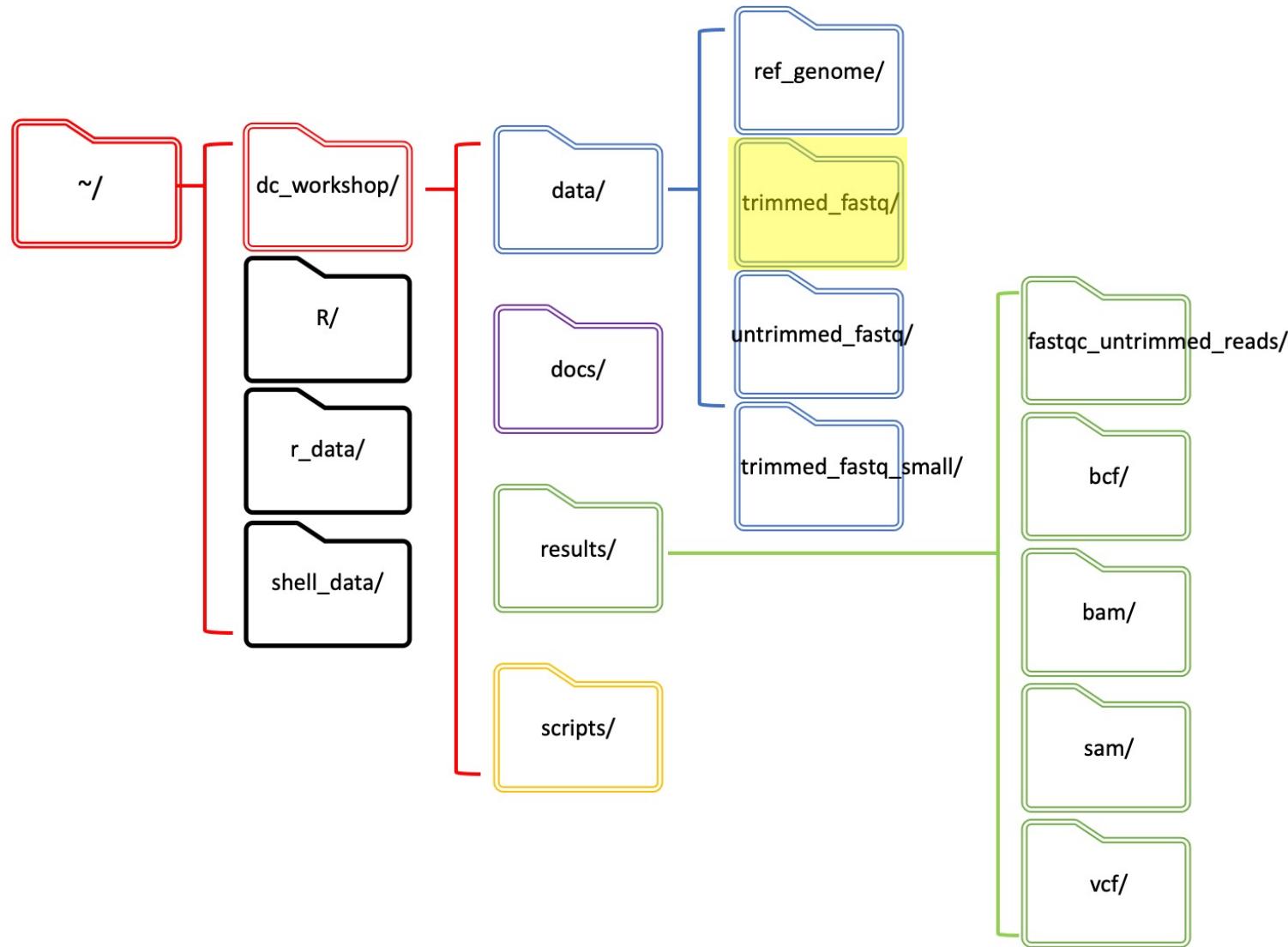
```

Define variables

Run alignment

Challenge/Homework:

- We were working with subsetted trimmed fastq data files for efficiency
- Re-run the alignment on the un-subsetted fastq files you generated earlier



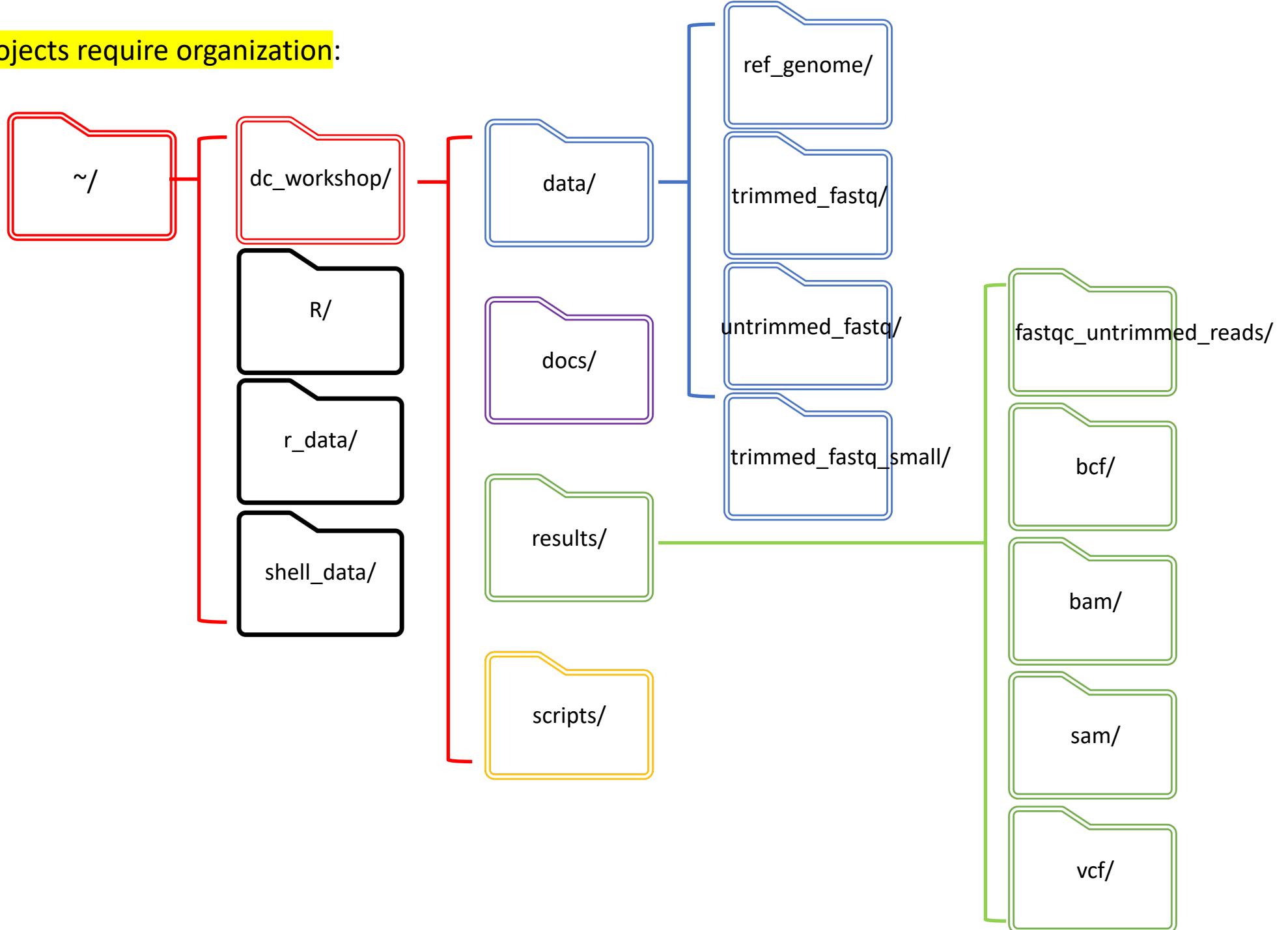
Goals for today:

- Project Organization
- Background and metadata
- Assessing sequencing read quality
- Trimming and filtering reads based on read quality
- Variant calling workflow
- Automating a variant calling workflow

Objectives:

- Create an organized file system for a bioinformatics project
- Identify which types of files should be stored in different directories
- Learn tools to document your work

Bioinformatics projects require organization:



Exercise: Make the following directories in your home directory

dc_workshop

dc_workshop/docs

dc_workshop/data

dc_workshop/results

dc_workshop/scripts

Exercise: Make the following directories in your home directory

dc_workshop

dc_workshop/docs

dc_workshop/data

dc_workshop/results

dc_workshop/scripts

Improve efficiency and reproducibility:

- 2 tools:
 - nano
 - Shell script or custom program

```
# Bioinformatics project template

# Instructions:
# CHANGE the PROJECT_DIR directory path in the script below:

PROJECT_DIR=/home/dcuser/dc_workshop/

# DATA:
DATA_DIR=${PROJECT_DIR}/data
DATA_RAW=${DATA_DIR}/raw
DATA_PROCESSED=${DATA_DIR}/processed

# RESULTS:
RESULTS_DIR=${PROJECT_DIR}/results

# DOCS:
DOCS_DIR=${PROJECT_DIR}/docs

# SCRIPTS:
SCRIPTS_DIR=${PROJECT_DIR}/scripts

mkdir -p ${PROJECT_DIR} \
${DATA_DIR} \
${DATA_RAW} \
${DATA_PROCESSED} \
${RESULTS_DIR} \
${DOCS_DIR} \
${SCRIPTS_DIR} \
```

denotes a comment
and is not run as part of
the script

```
# Bioinformatics project template
# Instructions:
# CHANGE the PROJECT_DIR directory path in the script below:
```

```
PROJECT_DIR=/home/dcuser/dc_workshop
```

```
# DATA:
```

```
DATA_DIR=${PROJECT_DIR}/data
```

```
DATA_RAW=${DATA_DIR}/raw
```

```
DATA_PROCESSED=${DATA_DIR}/processed
```

```
# RESULTS:
```

```
RESULTS_DIR=${PROJECT_DIR}/results
```

```
# DOCS:
```

```
DOCS_DIR=${PROJECT_DIR}/docs
```

```
# SCRIPTS:
```

```
SCRIPTS_DIR=${PROJECT_DIR}/scripts
```

```
mkdir -p ${PROJECT_DIR} \
${DATA_DIR} \
${DATA_RAW} \
${DATA_PROCESSED} \
${RESULTS_DIR} \
${DOCS_DIR} \
${SCRIPTS_DIR} \
```

denotes a comment
and is not run as part of
the script

```
# Bioinformatics project template
# Instructions:
# CHANGE the PROJECT_DIR directory path in the script below:
```

Define the VARIABLE that
corresponds to each
directory and subdirectory
you want to create:

```
PROJECT_DIR=/home/dcuser/dc_workshop
```

```
# DATA:
```

```
DATA_DIR=${PROJECT_DIR}/data
```

```
DATA_RAW=${DATA_DIR}/raw
```

```
DATA_PROCESSED=${DATA_DIR}/processed
```

```
# RESULTS:
```

```
RESULTS_DIR=${PROJECT_DIR}/results
```

```
# DOCS:
```

```
DOCS_DIR=${PROJECT_DIR}/docs
```

```
# SCRIPTS:
```

```
SCRIPTS_DIR=${PROJECT_DIR}/scripts
```

```
mkdir -p ${PROJECT_DIR} \
${DATA_DIR} \
${DATA_RAW} \
${DATA_PROCESSED} \
${RESULTS_DIR} \
${DOCS_DIR} \
${SCRIPTS_DIR} \
```

denotes a comment
and is not run as part of
the script

```
# Bioinformatics project template  
  
# Instructions:  
# CHANGE the PROJECT_DIR directory path in the script below:
```

Define the VARIABLE that
corresponds to each
directory and subdirectory
you want to create:

```
PROJECT_DIR=/home/dcuser/dc_workshop  
  
# DATA:  
DATA_DIR=${PROJECT_DIR}/data  
DATA_RAW=${DATA_DIR}/raw  
DATA_PROCESSED=${DATA_DIR}/processed  
  
# RESULTS:  
RESULTS_DIR=${PROJECT_DIR}/results  
  
# DOCS:  
DOCS_DIR=${PROJECT_DIR}/docs  
  
# SCRIPTS:  
SCRIPTS_DIR=${PROJECT_DIR}/scripts
```

```
mkdir -p ${PROJECT_DIR} \  
${DATA_DIR} \  
${DATA_RAW} \  
${DATA_PROCESSED} \  
${RESULTS_DIR} \  
${DOCS_DIR} \  
${SCRIPTS_DIR} \  
 
```

denotes a comment
and is not run as part of
the script

```
# Bioinformatics project template  
  
# Instructions:  
# CHANGE the PROJECT_DIR directory path in the script below:
```

Define the VARIABLE that
corresponds to each
directory and subdirectory
you want to create:

```
PROJECT_DIR=/home/dcuser/dc_workshop  
  
# DATA:  
DATA_DIR=${PROJECT_DIR}/data  
DATA_RAW=${DATA_DIR}/raw  
DATA_PROCESSED=${DATA_DIR}/processed  
  
# RESULTS:  
RESULTS_DIR=${PROJECT_DIR}/results  
  
# DOCS:  
DOCS_DIR=${PROJECT_DIR}/docs  
  
# SCRIPTS:  
SCRIPTS_DIR=${PROJECT_DIR}/scripts
```

Create each directory and
subdirectory using the
VARIABLE names

```
mkdir -p ${PROJECT_DIR} \  
${DATA_DIR} \  
${DATA_RAW} \  
${DATA_PROCESSED} \  
${RESULTS_DIR} \  
${DOCS_DIR} \  
${SCRIPTS_DIR} \  
\\
```

denotes a comment
and is not run as part of
the script

```
# Bioinformatics project template  
  
# Instructions:  
# CHANGE the PROJECT_DIR directory path in the script below:
```

Define the VARIABLE that
corresponds to each
directory and subdirectory
you want to create:

```
PROJECT_DIR=/home/dcuser/dc_workshop  
  
# DATA:  
DATA_DIR=${PROJECT_DIR}/data  
DATA_RAW=${DATA_DIR}/raw  
DATA_PROCESSED=${DATA_DIR}/processed  
  
# RESULTS:  
RESULTS_DIR=${PROJECT_DIR}/results  
  
# DOCS:  
DOCS_DIR=${PROJECT_DIR}/docs  
  
# SCRIPTS:  
SCRIPTS_DIR=${PROJECT_DIR}/scripts
```

Create each directory and
subdirectory using the
VARIABLE names

```
mkdir -p ${PROJECT_DIR} \  
${DATA_DIR} \  
${DATA_RAW} \  
${DATA_PROCESSED} \  
${RESULTS_DIR} \  
${DOCS_DIR} \  
${SCRIPTS_DIR} \  
\\
```

Customize these sections to fit
your project organization needs
and preferences by adding
additional directories