Follow Up from Yesterday

- Cyberduck

```
scp ~/.ssh/id_rsa.pub kkeith@10.1.105.13:.ssh/authorized_keys
```

- GitHub
  - I put the repository in the wrong folder
  - Walk through it again slowly

Processing RNA-seq Data

2020-07-23

Trim Bad Quality Sequences

What is trimming and why do it?

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information
- Why is that important?
  - Helps prevent incorrect base calls by removing poor quality information
  - Increases speed and accuracy of alignment by removing artificial sequences and low quality sequences

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information
- Why is that important?
  - Helps prevent incorrect base calls by removing poor quality information
  - Increases speed and accuracy of alignment by removing artificial sequences and low quality sequences

- Trimming does two complementary things:
  1. Removes any sequence information that comes from library preparation or sequencing
  2. Removes low quality bases / low quality reads

Trim Sequences

1. Go back up to the rnaseq directory
2. Make a folder to put the analysis results in, `analysis`
3. Make a folder inside the analysis folder to put the trimmed reads in, `analysis/01_trim`

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

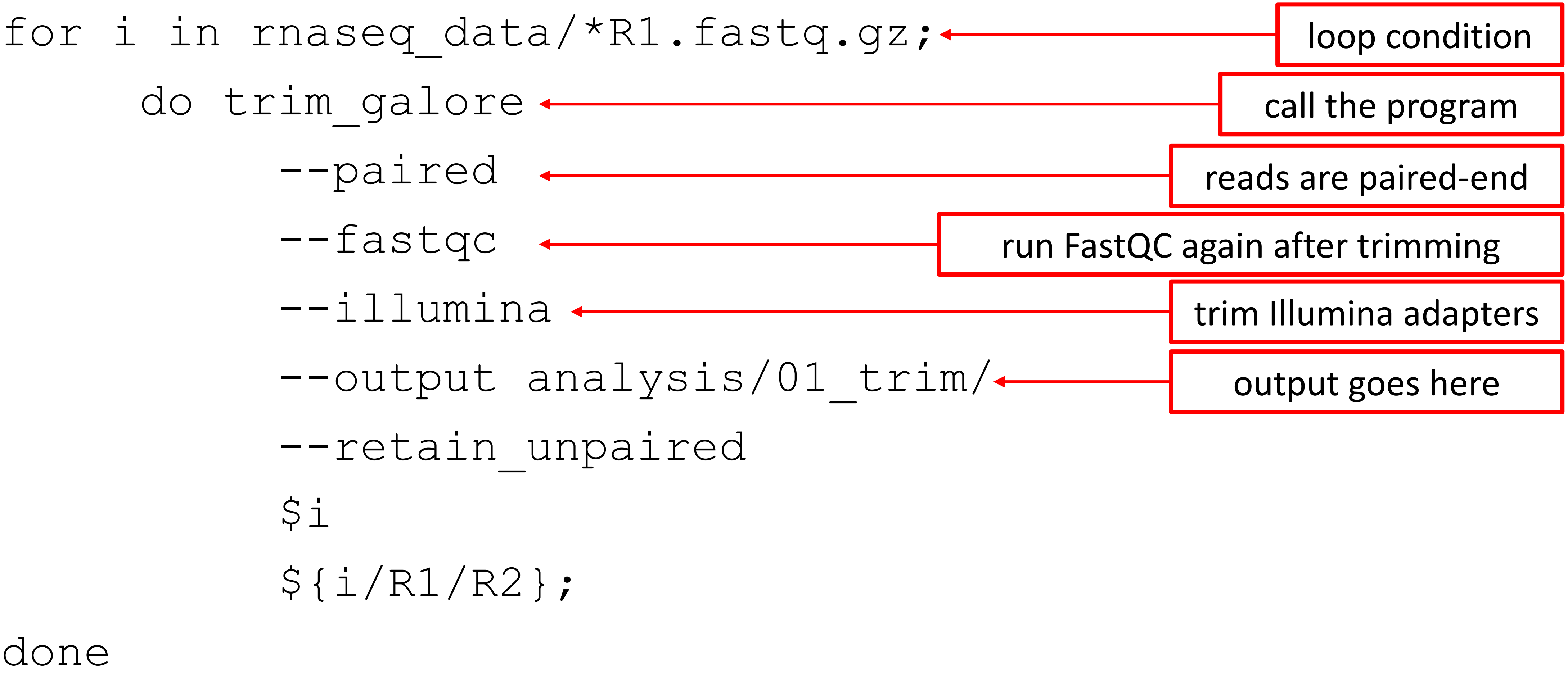
call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

Trim Sequences



```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

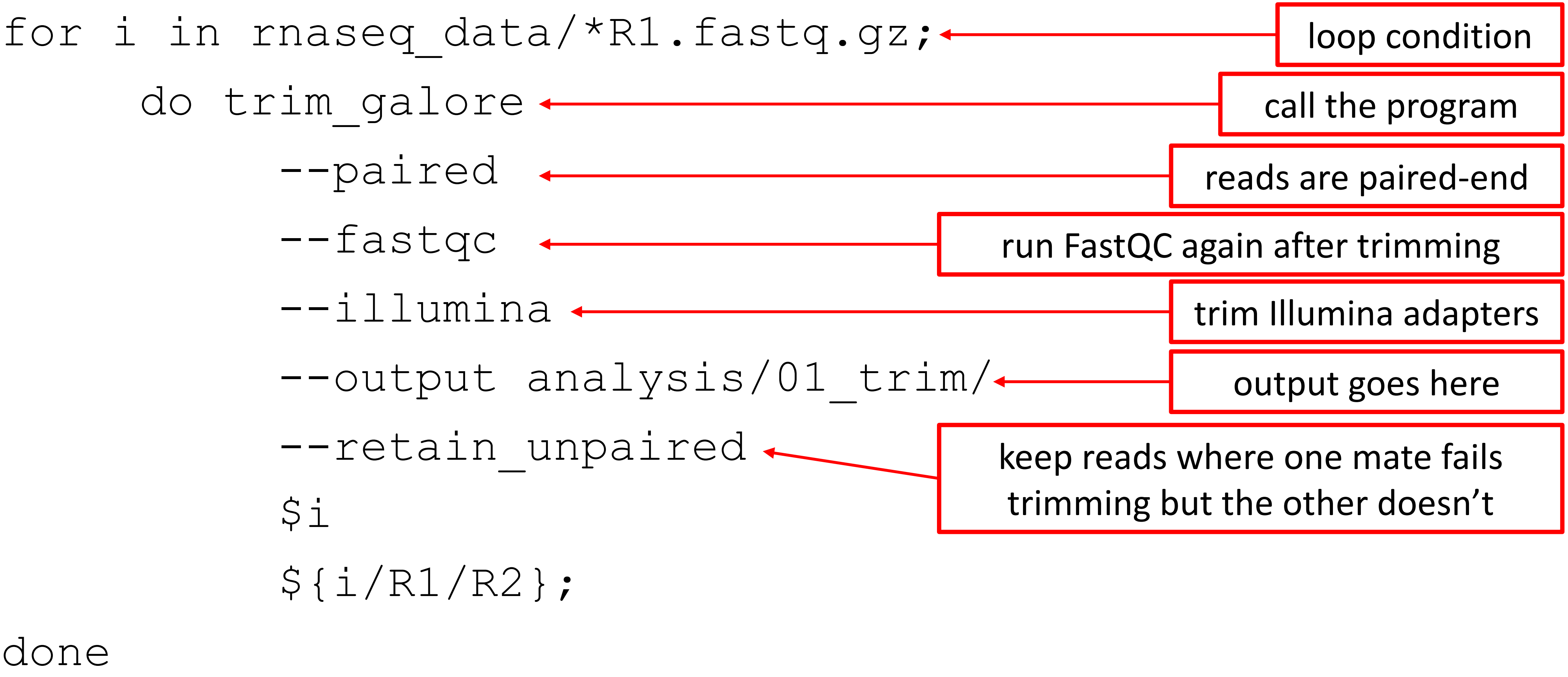
reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

Trim Sequences



```
for i in rnaseq_data/*R1.fastq.gz;
    do trim_galore
        --paired
        --fastqc
        --illumina
        --output analysis/01_trim/
        --retain_unpaired
        $i
        ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

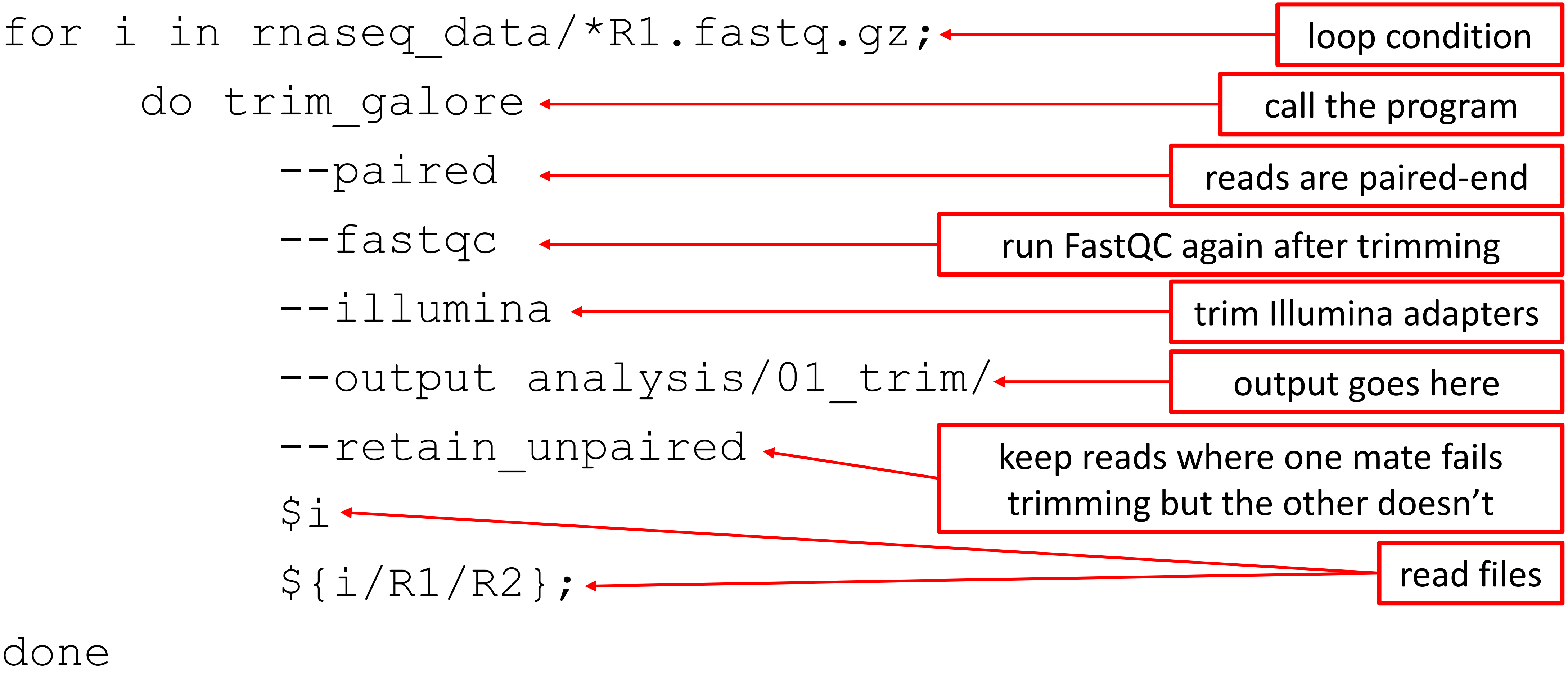
run FastQC again after trimming

trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn’t

Trim Sequences



```
for i in rnaseq_data/*R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output analysis/01_trim/
          --retain_unpaired
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

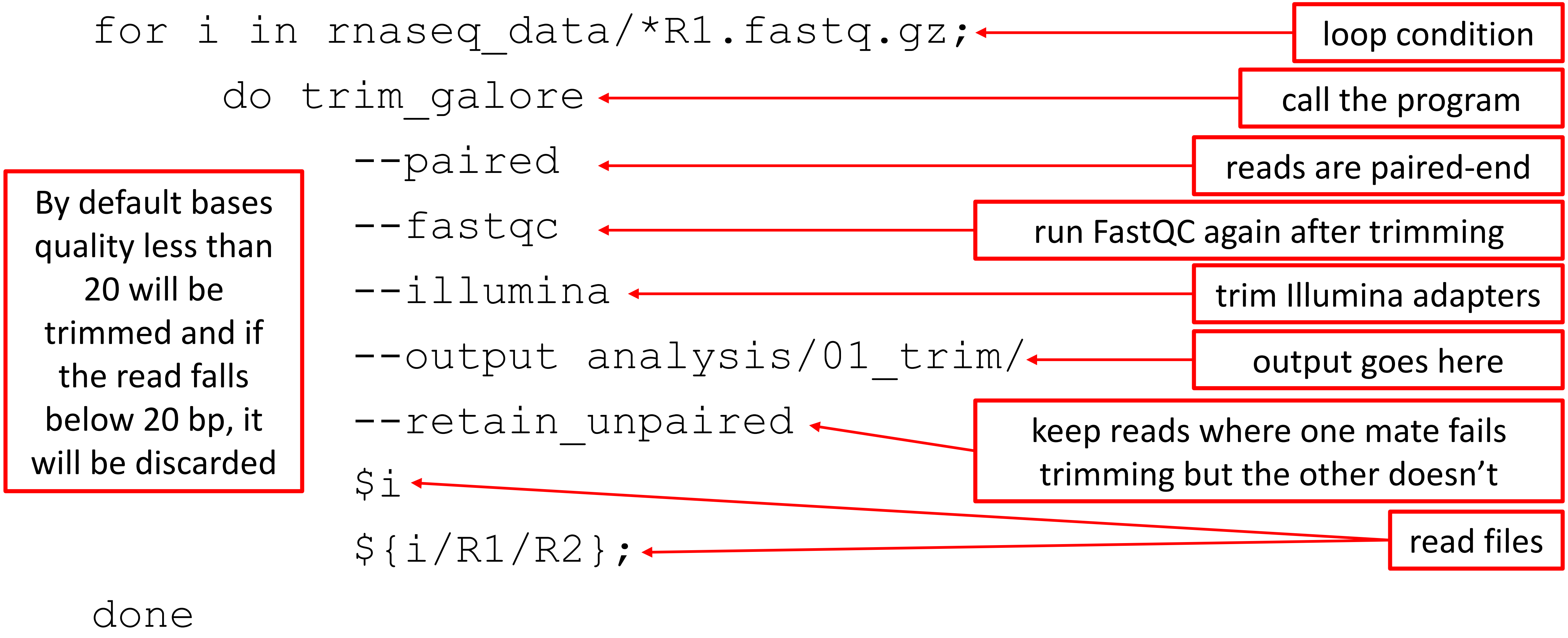
trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn't

read files

Trim Sequences



```
for i in rnaseq_data/*R1.fastq.gz;
        do trim_galore
                --paired
                --fastqc
                --illumina
                --output analysis/01_trim/
                --retain_unpaired
                $i
                ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn’t

read files

By default bases quality less than 20 will be trimmed and if the read falls below 20 bp, it will be discarded

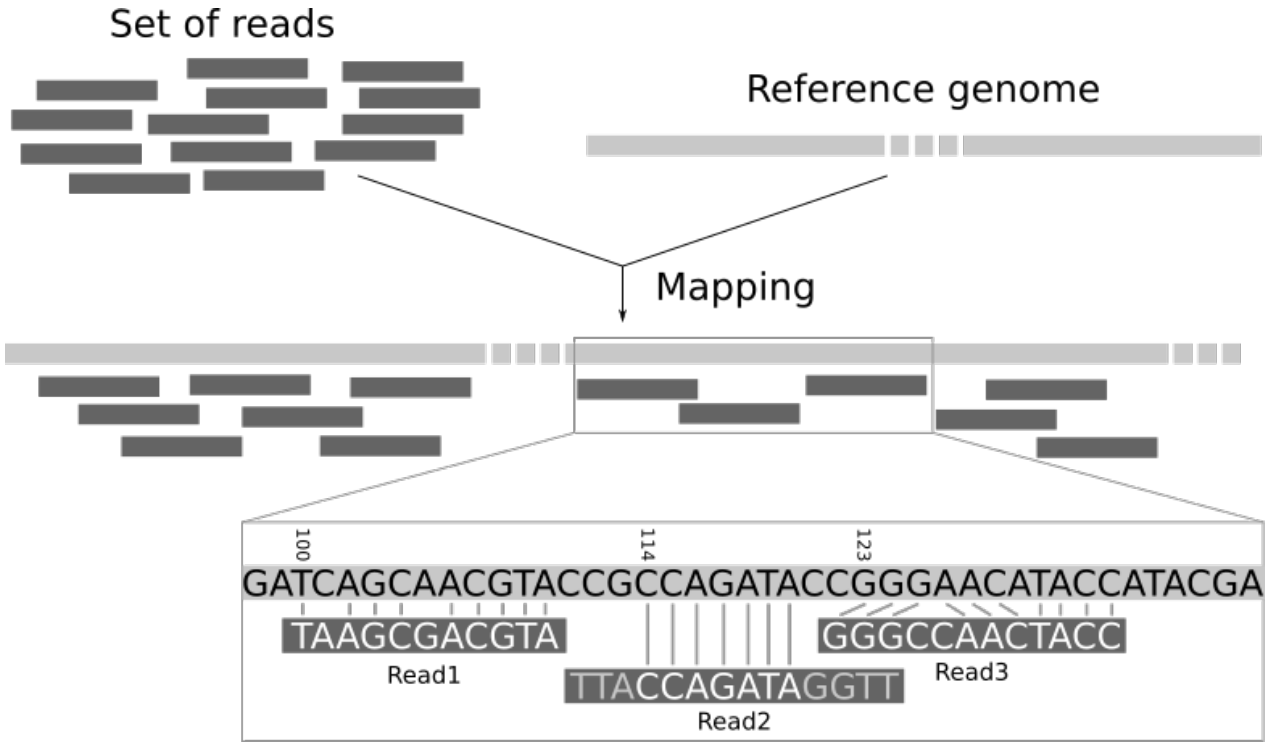
Trim Sequences

```
for i in rnaseq_data/*R1.fastq.gz; do trim_galore
--paired --fastqc --illumina --output analysis/01_trim/
--retain_unpaired $i ${i/R1/R2}; done
```

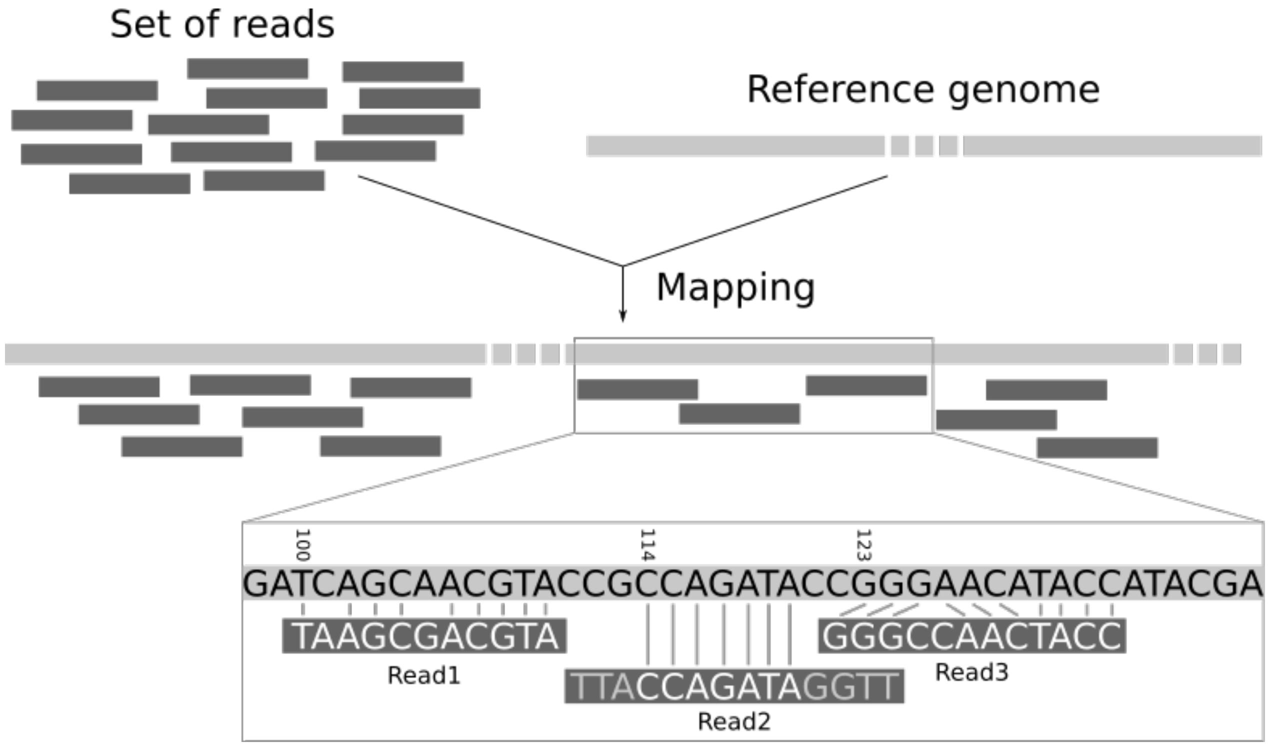
Align

How does aligning work?

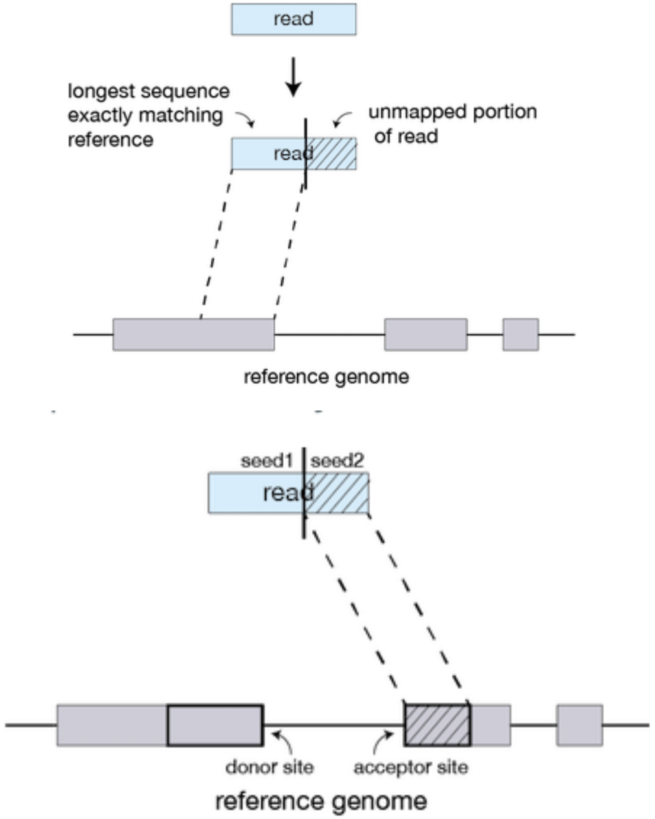
How does aligning work?



How does aligning work?



STAR (Spliced Transcripts Alignment to a Reference)



Align Sequences

1. Make a folder inside the analysis folder to put the aligned reads in, `analysis/02_align`
2. Change to the trimmed reads folder `analysis/01_trim`

Align Sequences

```
for i in *val_1.fq.gz;
     do STAR
         --genomeDir /mnt/data/gdata/human \
                      /hg38/chr21/STAR_index
         --readFilesIn $i ${i/R1_val_1/R2_val_2}
         --readFilesCommand zcat
         --outFileNamePrefix ../02_align/${i/R1*/}
         --outSAMtype BAM SortedByCoordinate;
done
```

Align Sequences

```
for i in *val_1.fq.gz;
     do STAR
          --genomeDir /mnt/data/gdata/human \
                      /hg38/chr21/STAR_index
          --readFilesIn $i ${i/R1_val_1/R2_val_2}
          --readFilesCommand zcat
          --outFileNamePrefix ../02_align/${i/R1*/}
          --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

Align Sequences

```
for i in *val_1.fq.gz;
     do STAR
          --genomeDir /mnt/data/gdata/human \
                      /hg38/chr21/STAR_index
          --readFilesIn $i ${i/R1_val_1/R2_val_2}
          --readFilesCommand zcat
          --outFileNamePrefix ../02_align/${i/R1*/}
          --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

Align Sequences

```
for i in *val_1.fq.gz;
        do STAR
                --genomeDir /mnt/data/gdata/human \
                                /hg38/chr21/STAR_index
                --readFilesIn $i ${i/R1_val_1/R2_val_2}
                --readFilesCommand zcat
                --outFileNamePrefix ../02_align/${i/R1*/}
                --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

path to reference genome

Align Sequences

```
for i in *val_1.fq.gz;
    do STAR
         --genomeDir /mnt/data/gdata/human \
                     /hg38/chr21/STAR_index
         --readFilesIn $i ${i/R1_val_1/R2_val_2}
         --readFilesCommand zcat
         --outFileNamePrefix ../02_align/${i/R1*/}
         --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

path to reference genome

trimmed read files

Align Sequences

```
for i in *val_1.fq.gz;
     do STAR
          --genomeDir /mnt/data/gdata/human \
                       /hg38/chr21/STAR_index
          --readFilesIn $i ${i/R1_val_1/R2_val_2}
          --readFilesCommand zcat
          --outFileNamePrefix ../02_align/${i/R1*/}
          --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

path to reference genome

trimmed read files

zipped files

Align Sequences

```
for i in *val_1.fq.gz;
   do STAR
      --genomeDir /mnt/data/gdata/human \
                  /hg38/chr21/STAR_index
      --readFilesIn $i ${i/R1_val_1/R2_val_2}
      --readFilesCommand zcat
      --outFileNamePrefix ../02_align/${i/R1*/}
      --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

path to reference genome

trimmed read files

zipped files

write the files here

Align Sequences

```
for i in *val_1.fq.gz;
     do STAR
          --genomeDir /mnt/data/gdata/human \
                      /hg38/chr21/STAR_index
          --readFilesIn $i ${i/R1_val_1/R2_val_2}
          --readFilesCommand zcat
          --outFileNamePrefix ../02_align/${i/R1*/}
          --outSAMtype BAM SortedByCoordinate;
done
```

loop condition

call aligner

path to reference genome

trimmed read files

zipped files

write the files here

write a sorted BAM

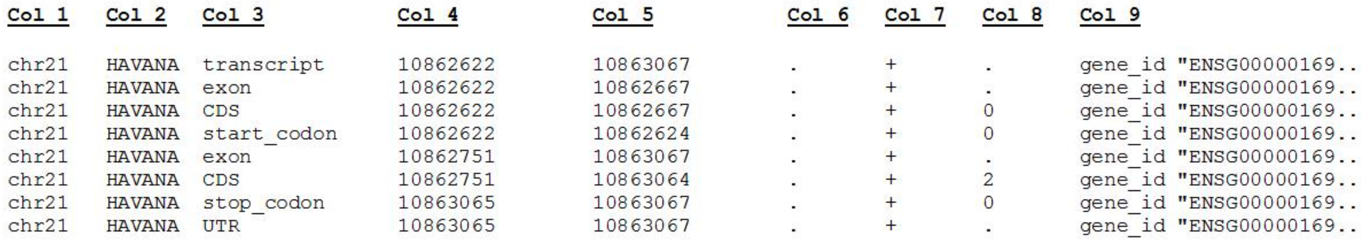
Align Sequences

```
for i in *val_1.fq.gz; do STAR --genomeDir
/mnt/data/gdata/human/hg38/chr21/STAR_index --
readFilesIn $i ${i/R1_val_1/R2_val_2} --
readFilesCommand zcat --outFileNamePrefix
../02_align/${i/R1*/} --outSAMtype BAM
SortedByCoordinate; done
```

Count Features

What do you mean by count features?

- We’re going to count genes, but you could also count:
    - transcripts
    - non-coding RNA
- Need an annotation file for whatever feature you want to count
- Going to use a gene transfer format (GTF) file for annotations



Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9
chr21	HAVANA	transcript	10862622	10863067	.	+	.	gene_id "ENSG00000169..
chr21	HAVANA	exon	10862622	10862667	.	+	.	gene_id "ENSG00000169..
chr21	HAVANA	CDS	10862622	10862667	.	+	0	gene_id "ENSG00000169..
chr21	HAVANA	start_codon	10862622	10862624	.	+	0	gene_id "ENSG00000169..
chr21	HAVANA	exon	10862751	10863067	.	+	.	gene_id "ENSG00000169..
chr21	HAVANA	CDS	10862751	10863064	.	+	2	gene_id "ENSG00000169..
chr21	HAVANA	stop_codon	10863065	10863067	.	+	0	gene_id "ENSG00000169..
chr21	HAVANA	UTR	10863065	10863067	.	+	.	gene_id "ENSG00000169..

Count Features

1. Make a folder inside the analysis folder to put the aligned reads in, `../03_count`
2. Change to the trimmed reads folder `../02_align/`

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

loop condition

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

loop condition

call program

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

loop condition

call program

path to genome annotation file

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

loop condition

call program

path to genome annotation file

where to write the output

Count Features

```
for i in *.bam;
        do featureCounts
                -a /mnt/data/gdata/human/hg38/chr21/ \
                   homo_sapiens_hg38_chr21.gtf
                -o ../03_count/${i/ \
                   Aligned.sortedByCoord.out.bam/ \
                   counts.txt}
                -R BAM
                $i;
done
```

loop condition

call program

path to genome annotation file

where to write the output

input files are BAM

Count Features

```
for i in *.bam;
     do featureCounts
          -a /mnt/data/gdata/human/hg38/chr21/ \
             homo_sapiens_hg38_chr21.gtf
          -o ../03_count/${i/ \
             Aligned.sortedByCoord.out.bam/ \
             counts.txt}
          -R BAM
          $i;
done
```

loop condition

call program

path to genome annotation file

where to write the output

input files are BAM

input file

Count Features

```
for i in *.bam; do featureCounts -a /mnt/data/gdata/human/hg38/chr21/homo_sapiens_hg38_chr21.gtf -o ../03_count/${i/Aligned.sortedByCoord.out.bam/counts.txt} -R BAM $i; done
```

General Steps

1. Check quality
2. Trim
3. Align
4. Count features
5. Statistics