

Bioinformatics Lessons Schedule

Date	Subject
01-07	Process RNA-seq
01-14	Process RNA-seq, continued
01-21	Process RNA-seq, continued
01-28	Analyze RNA-seq
02-04	Analyze RNA-seq, continued
02-11	Analyze RNA-seq, continued
02-18	skipped
02-25	Process RRBS
03-03	Process RRBS, continued
03-10	Process RRBS, continued
03-17	Analyze RRBS
03-24	Analyze RRBS, continued
03-31	Presentations / Practice Starts

Process RRBS, continued

2020-03-03

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
    --rrbs
    --fastqc
    --paired
    -q 30
    --illumina
    --output ../01_trim/
    $i
    ${i/1.fq.gz/2.fq.gz};
done
```

Step 2: Trimming

```
for i in *1.fq.gz;  
do trim_galore  
  --rrbs  
  --fastqc  
  --paired  
  -q 30  
  --illumina  
  --output ../01_trim/  
  $i  
  ${i/1.fq.gz/2.fq.gz};  
  
done
```

loop condition



Step 2: Trimming

```
for i in *1.fq.gz;  
do trim_galore  
  --rrbs  
  --fastqc  
  --paired  
  -q 30  
  --illumina  
  --output ../01_trim/  
  $i  
  ${i/1.fq.gz/2.fq.gz};  
  
done
```

loop condition

call program

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

loop condition

call program

trimming RRBS

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets
trim_galore
know to trim
bases that were
filled in by end
repair

loop condition

call program

trimming RRBS

run FastQC again after trimming

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

loop condition

call program

trimming RRBS

run FastQC again after trimming

reads are paired-end

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

loop condition

call program

trimming RRBS

run FastQC again after trimming

reads are paired-end

Minimum quality = 30

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

loop condition

call program

trimming RRBS

run FastQC again after trimming

reads are paired-end

Minimum quality = 30

trim Illumina adapters

Step 2: Trimming

```
for i in *1.fq.gz;
do trim_galore
--rrbs
--fastqc
--paired
-q 30
--illumina
--output ../01_trim/
$i
${i/1.fq.gz/2.fq.gz};
done
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

loop condition

call program

trimming RRBS

run FastQC again after trimming

reads are paired-end

Minimum quality = 30

trim Illumina adapters

output goes here

Step 2: Trimming

```
for i in *1.fq.gz;
```

loop condition

```
do trim_galore
```

call program

```
--rrbs
```

trimming RRBS

```
--fastqc
```

run FastQC again after trimming

```
--paired
```

reads are paired-end

```
-q 30
```

Minimum quality = 30

```
--illumina
```

trim Illumina adapters

```
--output ../01_trim/
```

output goes here

```
$i
```

reads

```
${i/1.fq.gz/2.fq.gz};
```

RRBS flag lets trim_galore know to trim bases that were filled in by end repair

```
done
```

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
    bismark
        --bowtie2
        /mnt/data/data_jj/jj4/rrbs/tools \
        /genomes/hg19lambda
        --output ../02_align
        -1 $i
        -2 ${i/1_val_1.fq.gz/ \
            2_val_2.fq.gz};
done
```

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
    bismark
        --bowtie2
        /mnt/data/data_jj/jj4/rrbs/tools \
        /genomes/hg19lambda
        --output ../02_align
        -1 $i
        -2 ${i/1_val_1.fq.gz/ \
            2_val_2.fq.gz};
done
```



loop condition

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
```

loop condition

```
  bismark
```

call program

Aligns to a
normal genome
and a C-to-T
converted
genome

```
  --bowtie2
```

```
  /mnt/data/data_jj/jj4/rrbs/tools \
```

```
  /genomes/hg19lambda
```

```
  --output ../02_align
```

```
  -1 $i
```

```
  -2 ${i/1_val_1.fq.gz/ \
```

```
      2_val_2.fq.gz};
```

```
done
```

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
    bismark
        --bowtie2
        /mnt/data/data_jj/jj4/rrbs/tools \
        /genomes/hg19lambda
        --output ../02_align
        -1 $i
        -2 ${i/1_val_1.fq.gz/ \
            2_val_2.fq.gz};
done
```

Aligns to a normal genome and a C-to-T converted genome

loop condition

call program

aligner

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
    bismark
        --bowtie2
        /mnt/data/data_jj/jj4/rrbs/tools \
        /genomes/hg19lambda
        --output ../02_align
        -1 $i
        -2 ${i/1_val_1.fq.gz/ \
            2_val_2.fq.gz};
done
```

loop condition

call program

aligner

path to genome

Aligns to a normal genome and a C-to-T converted genome

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
    bismark
        --bowtie2
        /mnt/data/data_jj/jj4/rrbs/tools \
        /genomes/hg19lambda
        --output ../02_align
        -1 $i
        -2 ${i/1_val_1.fq.gz/ \
            2_val_2.fq.gz};
done
```

loop condition

call program

aligner

path to genome

output here

Aligns to a
normal genome
and a C-to-T
converted
genome

Step 3: Aligning

```
for i in *1_val_1.fq.gz; do
```

```
    bismark
```

```
        --bowtie2
```

```
        /mnt/data/data_jj/jj4/rrbs/tools \
```

```
        /genomes/hg19lambda
```

```
        --output ../02_align
```

```
        -1 $i
```

```
        -2 ${i/1_val_1.fq.gz/ \
```

```
            2_val_2.fq.gz};
```

```
done
```

loop condition

call program

aligner

path to genome

output here

Read 1

Read 2

Aligns to a normal genome and a C-to-T converted genome

Step 3: Call Methylation

```
for i in *.bam; do
    bismark_methylation_extractor
        --paired-end
        --include_overlap
        --bedGraph
        --output ../03_extract_meth
        $i;
done
```

Step 3: Call Methylation

```
for i in *.bam; do
    bismark_methylation_extractor
        --paired-end
        --include_overlap
        --bedGraph
        --output ../03_extract_meth
        $i;
done
```



loop condition

Step 3: Call Methylation

```
for i in *.bam; do
    bismark_methylation_extractor
        --paired-end
        --include_overlap
        --bedGraph
        --output ../03_extract_meth
        $i;
done
```

loop condition

call program

Counts number
of Cs and Ts at
each position

Step 3: Call Methylation

```
for i in *.bam; do
    bismark_methylation_extractor
        --paired-end
        --include_overlap
        --bedGraph
        --output ../03_extract_meth
        $i;
done
```

Counts number
of Cs and Ts at
each position

loop condition

call program

reads paired

Step 3: Call Methylation

```
for i in *.bam; do
```

loop condition

```
    bismark_methylation_extractor
```

call program

```
        --paired-end
```

reads paired

Counts number
of Cs and Ts at
each position

```
        --include_overlap
```

Include CpGs where read 1 and read 2 overlap

```
        --bedGraph
```

```
        --output ../03_extract_meth
```

```
    $i;
```

```
done
```


Step 3: Call Methylation

```
for i in *.bam; do
```

loop condition

```
    bismark_methylation_extractor
```

call program

```
    --paired-end
```

reads paired

Counts number
of Cs and Ts at
each position

```
    --include_overlap
```

Include CpGs where read 1 and read 2 overlap

```
    --bedGraph
```

output results for each CpG

```
    --output ../03_extract_meth
```

```
    $i;
```

```
done
```

Step 3: Call Methylation

```
for i in *.bam; do
```

loop condition

```
    bismark_methylation_extractor
```

call program

```
    --paired-end
```

reads paired

Counts number
of Cs and Ts at
each position

```
    --include_overlap
```

Include CpGs where read 1 and read 2 overlap

```
    --bedGraph
```

output results for each CpG

```
    --output ../03_extract_meth
```

Output here

```
    $i;
```

```
done
```

Step 3: Call Methylation

```
for i in *.bam; do
    bismark_methylation_extractor
        --paired-end
        --include_overlap
        --bedGraph
        --output ../03_extract_meth
    $i;
done
```

Annotations for the script:

- loop condition (points to `for i in *.bam; do`)
- call program (points to `bismark_methylation_extractor`)
- reads paired (points to `--paired-end`)
- Include CpGs where read 1 and read 2 overlap (points to `--include_overlap`)
- output results for each CpG (points to `--bedGraph`)
- Output here (points to `--output ../03_extract_meth`)
- Bam file (points to `$i;`)

Counts number of Cs and Ts at each position (points to the entire script block)

(Very) General Bioinformatics Workflow

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