In [58]:
```
#### same pipeline on DSV's data
import pandas as pd
from jupyterquiz import display_quiz
from IPython.display import IFrame
from IPython.display import display
from jupytercards import display_flashcards
from IPython.display import Image
import pandas as pd
```

In [5]:
```
numthreads=!lscpu | grep '^CPU(s)'| awk '{print $2-1}'
numthreadsint = int(numthreads[0])
```

In [6]:
```
numthreadsint
```

Out[6]:
```
7
```

In [2]:
```
# !mkdir -p dsv/QC
# !mkdir -p dsv/Trimmed
# !mkdir -p dsv/Mapped
```

In [7]:
```
!ls dsv/Input
```

```
10C-100K-2_1.fq.gz  10C-100K-2_2.fq.gz
```

In [8]:
```
#This command runs fastqc on each fastq.gz file inside our InputFiles directory and stores the ouput reports in
!fastqc -t $numthreadsint -q -o dsv/QC dsv/Input/*.fq.gz

#We then use multiqc to summarize the report.
!multiqc -o dsv/QC -f dsv/QC 2> dsv/QC/multiqc_log.txt

#We'll load this into a pandas table to work in this context, but fastqc also produces an html report that you
dframe = pd.read_csv("dsv/QC/multiqc_data/multiqc_fastqc.txt", sep='\t')
```

```
application/gzip
application/gzip
```

In [22]:
```
display(dframe[['Sample','Sequences flagged as poor quality', 'Sequence length',
       '%GC', 'total_deduplicated_percentage', 'avg_sequence_length',
       'median_sequence_length', 'basic_statistics',
       'per_base_sequence_quality', 'per_tile_sequence_quality',
       'per_sequence_quality_scores', 'per_base_sequence_content',
       'per_sequence_gc_content', 'per_base_n_content',
       'sequence_length_distribution', 'sequence_duplication_levels',
       'overrepresented_sequences', 'adapter_content']])
```

	Sample	Sequences flagged as poor quality	Sequence length	%GC	total_deduplicated_percentage	avg_sequence_length	median_sequence_length	basic_statistics	per
0	10C-100K-2_1	0.0	150.0	48.0	51.207315	150.0	150	pass	
1	10C-100K-2_2	0.0	150.0	48.0	53.592826	150.0	150	pass	

In [19]:
```
# #We can display the resulting fastqc results.
# IFrame(src='dsv/QC/10C-100K-2_1_fastqc.html', width=1500, height=600)
```

In [20]:
```
# #We can display the resulting fastqc results.
# IFrame(src='dsv/QC/10C-100K-2_2_fastqc.html', width=1500, height=600)
```

In [24]:
```
######### TRIMMING

#This will trim off N's as well as nextera adapters present in ATAC-seq library preparation. placing the trimme
!trimmomatic PE -threads $numthreadsint dsv/Input/10C-100K-2_1.fq.gz dsv/Input/10C-100K-2_2.fq.gz dsv/Trimmed/1
```

```
TrimmomaticPE: Started with arguments:
 -threads 7 dsv/Input/10C-100K-2_1.fq.gz dsv/Input/10C-100K-2_2.fq.gz dsv/Trimmed/10C-100K-2_1_trimmed_R1.fq.gz
dsv/Trimmed/10C-100K-2_1_unpaired_R1.fq.gz dsv/Trimmed/10C-100K-2_2_trimmed_R2.fq.gz dsv/Trimmed/10C-100K-2_2_u
npaired_R2.fq.gz ILLUMINACLIP:dsv/RefGenome/NexteraPE.fa:2:30:10 LEADING:3 TRAILING:3
Using PrefixPair: 'AGATGTGTATAAGAGACAG' and 'AGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'TCGTCGGCAGCGTCAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTCCGAGCCCACGAGAC'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTGACGCTGCCGACGA'
ILLUMINACLIP: Using 1 prefix pairs, 4 forward/reverse sequences, 0 forward only sequences, 0 reverse only seque
nces
Quality encoding detected as phred33
Input Read Pairs: 73087456 Both Surviving: 47674892 (65.23%) Forward Only Surviving: 25382213 (34.73%) Reverse
Only Surviving: 1968 (0.00%) Dropped: 28383 (0.04%)
TrimmomaticPE: Completed successfully
```

In [25]:
```
#This command runs fastqc on each fastq.gz file inside our InputFiles directory and stores the ouput reports in
!fastqc -t $numthreadsint -q -o dsv/Trimmed dsv/Trimmed/*.fq.gz

#We then use multiqc to summarize the report.
!multiqc -o dsv/QC -f dsv/Trimmed 2> dsv/QC/multiqc_log.txt

#We'll load this into a pandas table to work in this context, but fastqc also produces an html report that you
dframe = pd.read_csv("dsv/QC/multiqc_data/multiqc_fastqc.txt", sep='\t')
```

```
application/gzip
application/gzip
application/gzip
application/gzip
```

In [26]:
```
dframe
```

Out[26]:

	Sample	Filename	File type	Encoding	Total Sequences	Total Bases	Sequences flagged as poor quality	Sequence length	%GC	total_deduplicated_perce
0	10C-100K-2_1_trimmed_R1	10C-100K-2_1_trimmed_R1.fq.gz	Conventional base calls	Sanger / Illumina 1.9	47674892.0	7.1 Gbp	0.0	2-150	48.0	52.6
1	10C-100K-2_1_unpaired_R1	10C-100K-2_1_unpaired_R1.fq.gz	Conventional base calls	Sanger / Illumina 1.9	25382213.0	2 Gbp	0.0	3-150	50.0	52.4
2	10C-100K-2_2_trimmed_R2	10C-100K-2_2_trimmed_R2.fq.gz	Conventional base calls	Sanger / Illumina 1.9	47674892.0	7.1 Gbp	0.0	2-150	48.0	58.0
3	10C-100K-2_2_unpaired_R2	10C-100K-2_2_unpaired_R2.fq.gz	Conventional base calls	Sanger / Illumina 1.9	1968.0	291.5 kbp	0.0	2-150	48.0	85.6

4 rows × 23 columns

In [29]:
```
#We can display the resulting fastqc results.
# IFrame(src='dsv/Trimmed/10C-100K-2_2_trimmed_R2_fastqc.html', width=1080, height=800)
```

In [30]:
```
#### Step3 Mapping
```

In [31]:
```
!ls dsv/RefGenome/*bt2
```

```
dsv/RefGenome/hg38_noalt_as.1.bt2  dsv/RefGenome/hg38_noalt_as.4.bt2
dsv/RefGenome/hg38_noalt_as.2.bt2  dsv/RefGenome/hg38_noalt_as.rev.1.bt2
dsv/RefGenome/hg38_noalt_as.3.bt2  dsv/RefGenome/hg38_noalt_as.rev.2.bt2
```

In [32]:
```
!ls dsv/RefGenome/*fa
```

```
dsv/RefGenome/NexteraPE.fa  dsv/RefGenome/hg38.fa
```

In [33]:
```
#####BOWTIE2 using the index we build
```

In [ ]:
```
#Notes: The -x option specifies the prefix of the index. -1 specifies our left-end trimmed reads file. -2 speci
!bowtie2 -p $numthreadsint -x dsv/RefGenome/hg_self_buildindex/hg38_selfbuild -1 dsv/Trimmed/10C-100K-2_1_trimm
```

In [35]:
```
###check run
```

In [36]:
```
#sam to bam
```

In [37]:
```
#This will convert to bam by using samtools view with the -b option. The h and S option tells samtools that the
!samtools view -q 10 -bhS dsv/Mapped/dsv_data.sam | samtools sort -o dsv/Mapped/dsv_data.bam -
print("done")
```

```
[bam_sort_core] merging from 7 files and 1 in-memory blocks...
done
```

In [38]:
```
### remove duplicates using picard
```

In [39]:
```
#this will take the sorted bam file and remove duplicates, saving a new bam file and a summary in a text file.
!picard MarkDuplicates --REMOVE_DUPLICATES TRUE -I dsv/Mapped/dsv_data.bam -O dsv/Mapped/dsv_data_dedup.bam --M
print("done")
```

```
done
```

In [40]:
```
#We can use multiqc to summarize the metrics
!multiqc -o dsv/QC -f dsv/Mapped 2> dsv/Mapped/multiqc_log.txt
dframe = pd.read_csv("dsv/QC/multiqc_data/multiqc_general_stats.txt", sep='\t')
display(dframe)
```

	Sample	Picard_mqc-generalstats-picard-PERCENT_DUPLICATION
0	dsv_data	0.395758

In [41]:
```
#### view sam
```

In [46]:
```
!mkdir -p dsv/BigWigFiles
!mkdir -p dsv/Peaks
!mkdir -p dsv/Plots
```

In [42]:
```
!samtools view dsv/Mapped/dsv_data_dedup.bam | head -3
#Note that there will be an error message because we are breaking a pipe by printing only the first 3 lines. Pl
```

```
E00572:542:HJHFCCCX2:7:2220:18111:58743 73      chr1    10016   12      131M1D19M       =       10016   0
CCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCAACCCTAACCCTAACCCT
AACCCTAACCCTAACCCTAACCCTAACCCTAACCCTCAC AAFFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFFFFFJJJFFFJJFFFJJFJJFJJ--<-A7---7---<---<F))A MD:Z:131^C16A2  PG:Z:MarkDuplic
ates    XG:i:1  NM:i:2  XM:i:1  XN:i:0  XO:i:1  AS:i:-10        YT:Z:UP
E00572:542:HJHFCCCX2:7:2119:24413:44169 73      chr1    10148   11      30M1I53M1D66M   =       10148   0
CCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAACCCTAAAC
CCTAACCCTAACCCTAACCCTAACCCTAACCCCAACCCT AAFFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJJJJFAJFJJFFJJJFJJFJFF<7
FJJFF<FJAA7AJJJAJ<FFJFJ-<AFFAAJFJJ---7-A-7<-7AF7-<<A7FA-7<FFFJ<FJF--7-A<)7)<7J7 MD:Z:83^C65C0   PG:Z:MarkDuplic
ates    XG:i:2  NM:i:3  XM:i:1  XN:i:0  XO:i:2  AS:i:-20        YT:Z:UP
E00572:542:HJHFCCCX2:7:2115:14549:16639 99      chr1    15821   31      150M    =       15830   159     CTTCTCC
AGCTTTCGCTCCTTCATGCTGCGCAGCTTGGCCTTGCCGATGCCCCCAGCTTGGCGGATGGACTCTAGCAGAGTGGCCAGCCACCGGAGGGGTCAACCACTTCCCTGGGAG
CTCCCTGGACTGGAGCCGGGAGGTGGGGAACA        AAFFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJJ<FJJJAJFJ MD:Z:150        PG:Z:MarkDuplic
ates    XG:i:0  NM:i:0  XM:i:0  XN:i:0  XO:i:0  AS:i:0  XS:i:-20        YS:i:0  YT:Z:CP
samtools view: writing to standard output failed: Broken pipe
samtools view: error closing standard output: -1
```

In [43]:
```
# better visualization
```

In [44]:
```
# First we need to create an index of our bam file.
!samtools index dsv/Mapped/dsv_data_dedup.bam
```

In [47]:
```
# Then we can create a bigwig file of the sample.
!bamCoverage -b dsv/Mapped/dsv_data_dedup.bam -o dsv/BigWigFiles/dsv_data.bw -bs 1 -p $numthreadsint --normaliz
```

In [49]:
```
import sys
# # sys.path.insert(0, "/home/jupyter/.local/lib/python3.7/site-packages")
sys.path.insert(0,  "/home/jupyter/.local/lib/python3.10/site-packages")
```

In [50]:
```
import igv_notebook
```

In [52]:
```
igv_notebook.init()
myigv = igv_notebook.Browser(
    {
        "genome": "hg38",
        "locus": "chr21:15,400,000-26,400,000"
    }
)
myigv.load_track(
{
        "name": "DSV",
        "url": "dsv/BigWigFiles/dsv_data.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
```

In [53]:
```
#### I tried to find whole genome bed but I couldn't so will just use the one provided in tutorial for chr4
```

In [54]:
```
#For example, ATAC-seq signal should be enriched near TSSs
```

In [55]:
```
#-S option specifies the bigwig signal file, where we can specify multiple separated by spaces. -R option speci
!computeMatrix reference-point --referencePoint TSS -S dsv/BigWigFiles/dsv_data.bw -R dsv/GenomeAnnotations/hg3
```

In [56]:
```
!plotProfile -m dsv/Plots/TSSprofileMatrix -o dsv/Plots/TSSprofile.png
```

Let's view the output:

In [59]:
```
Image(url= "dsv/Plots/TSSprofile.png", width=400, height=400)
```

Out[59]:

Nucleosomes consist of 145 bp of DNA wrapped around histones. Because Tn5 randomly inserts near protected sites, in paired-end ATAC-seq this results in a slightly larger range of protected fragments (i.e. insertion sizes). Based on this information, look at the graph and think about the size range that would be most consistent with TF binding vs mono-nucleosomes.

We can use Deeptools to summarize our insert sizes.

In [61]:
```
!bamPEFragmentSize -b dsv/Mapped/dsv_data_dedup.bam -o dsv/Plots/Insertsizes_histogram.png -p $numthreadsint --
print("done")
```

```
done
```

In [62]:
```
Image(url= "dsv/Plots/Insertsizes_histogram.png", width=400, height=400)
```

Out[62]:

I do not understand the above plot but may you be able to understand

With paired-end ATAC-seq data we can separate by fragment size to obtain Transposase HyperSensitive Sites (THSS) and Nucleosomal Fragments. Alternatively, some choose to keep the data together as a more general measure of "accessible" sites.

dsv_data_dedup is known as DDD from now on

In [66]:
```
#Filter by insert size:

### Nucleosomal
!samtools view -h dsv/Mapped/dsv_data_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 150 && $9<=250) || ($9<=-15


### THSS
!samtools view -h dsv/Mapped/dsv_data_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 10 && $9<=125) || ($9<=-10
```

STEP3: Peak Detection

ATAC shift reads

Tn5 insertion of adapters leaves a 9 bp gap. In the end, this probably won't impact the results much. However, to be safe we can shift the reads to account for this insertion offset.

In [67]:
```
!alignmentSieve -p $numthreadsint --ATACshift -b dsv/Mapped/dsv_data_dedup.bam -o dsv/Mapped/DDD_shift.bam
```

from now on DDD_shift is the file to work with

Let's identify Peaks genome-wide using macs2.

In [70]:
```
!macs2 callpeak -f BAMPE -g hs --keep-dup all --cutoff-analysis -n CTL -t dsv/Mapped/DDD_shift.bam --outdir dsv
```

peakcalling done

now visualize

In [72]:
```
igv_notebook.init()
```

In [74]:
```
igv_notebook.init()
myigv = igv_notebook.Browser(
    {
        "genome": "hg38",
        "locus": "chr4:55,570,000-55,670,000"
    }
)
myigv.load_track(
{
        "name": "DSV",
        "url": "dsv/BigWigFiles/dsv_data.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
myigv.load_track(
{
        "name": "DSV_peaks",
        "url": "dsv/Peaks/CTL_peaks.narrowPeak",
        "format": "bed",
        "type": "annotation"
    }

)
```

In [ ]: