Script

Most of the following scripts were modified from the examples given by TA.

Modifications included changing the files used in the scripts and fixing bugs we met.

We also wrote some new scripts to meet our needs.

These scripts are marked out by【new】in the contents.

ps: all the annotations for "sbatch" were deleted to make this report shorter and more brief (unless for special use)

tip: click the title in the contents to jump directly to the corresponding script

- Prepare Data Matrix
  - Shapemapper
    - SHAPEMAPPER for AT1G09530.3【new】
    - Reactivity Calculation for AT1G09530【new】
  - Data Analysis
    - Structure Change Analysis
      - 3 'UTR and 5' UTR sequence information of the TE changing region was extracted Calculate Hit Level
      - Draw the Trend Chart of the Number of Transcripts Changing with HIT Level
      - Calculate the Normalization Factor
      - Preprocessing and Normalization of Reactivity
      - Calculate Gini Index
      - Merge Structural Change Regions

- Data Integration
  - Transcript Abundance & TE
  - Structure Change & TE
    - Hypothesis Testing
    - Draw Enrichment Degree
  - Motif Analysis
    - Extract All 3’UTR or 5’UTR FASTQ Files
    - Extract 3 'UTR and 5' UTR Sequence of the TE Changing Region
      - Generate Gene List of Up-regulated TE【new】
      - Generate Gene List of Down-regulated TE【new】
      - Motif Analysis with MEME

RNA-seq Analysis

Prepare Data Matrix

QC of Raw Data

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH
echo start fastqc `date`

sample=('CD1_1' 'CD1_2' 'CD1_3')
path=('/WT/control/nouvb/' '/WT/control/nouvb/' '/WT/control/nouvb/')

for i in $(seq 1 ${#sample[@]});do
echo finish ${sample[$i-1]} `date`

cd /data/user_03/RiboShape/Part1/RNAseq
mkdir -p 1.fastqc/control/${sample[$i-1]}

fastqc \
/data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/SHAPE-MaP/${path[$i-
1]}/${sample[$i-1]}.clean.1.fastq.gz \
/data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/SHAPE-MaP/${path[$i-
1]}/${sample[$i-1]}.clean.2.fastq.gz \
--outdir /data/user_03/RiboShape/Part1/RNAseq/1.fastqc/control/${sample[$i-1]} -
-noextract >> /data/user_03/RiboShape/Part1/RNAseq/1.fastqc/control/${sample[$i-
1]}/${sample[$i-1]}.log 2>&1

echo finish ${sample[$i-1]} `date`
done
echo finish fastqc `date`
```

Cut Adaptors and Trim Long Reads

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start trimmed
```

```
sample=('CD1_1' 'CD1_2' 'CD1_3')
path=('/WT/control/nouvb/' '/WT/control/nouvb/' '/WT/control/nouvb/')

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part1/RNAseq
mkdir -p 2.quality_control/control/${sample[$i-1]}

fastp -i /data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/SHAPE-
MaP/${path[$i-1]}/${sample[$i-1]}.clean.1.fastq.gz \
-I /data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/SHAPE-
MaP/${path[$i-1]}/${sample[$i-1]}.clean.2.fastq.gz \
-o /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.1.fastq.gz \
-O /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.2.fastq.gz \
--thread=4 -l 15 \
-j /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.json \
-h /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.html

echo finish ${sample[$i-1]} `date`
done

echo trimmed success
```

Remove rRNA Reads

Remove rRNA reads in CD1_1

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_1')

echo start remove_rRNA `date`

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part1/RNAseq/
mkdir -p 3.remove_rRNA/fastq/control/${sample[$i-1]}
mkdir -p 3.remove_rRNA/rRNA/control/${sample[$i-1]}

bowtie -n 0 -y -a --norc --best --strata -S -p 4 -l 15 \
--un
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA.fq \
/data/TA_QUIZ_RNA_regulation/data/ATH/index/bowtie1/rRNA/Arabidopsis_thaliana.TA
IR10.34.rRNA \
-1 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.1.fastq.gz \
-2 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.2.fastq.gz \
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.aligned_rRNA.txt
```

```
cd /data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/
gzip ${sample[$i-1]}.rm_rRNA_1.fq
gzip ${sample[$i-1]}.rm_rRNA_2.fq
rm ${sample[$i-1]}.alingned_rRNA.txt

echo finish ${sample[$i-1]} `date`
done

echo remove_rRNA success `date`
```

Remove rRNA reads in CD1_2

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_2')

echo start remove_rRNA `date`

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part1/RNAseq/
mkdir -p 3.remove_rRNA/fastq/control/${sample[$i-1]}
mkdir -p 3.remove_rRNA/rRNA/control/${sample[$i-1]}

bowtie -n 0 -y -a --norc --best --strata -S -p 4 -l 15 \
--un
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA.fq \
/data/TA_QUIZ_RNA_regulation/data/ATH/index/bowtie1/rRNA/Arabidopsis_thaliana.TA
IR10.34.rRNA \
-1 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.1.fastq.gz \
-2 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.2.fastq.gz \
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.aligned_rRNA.txt

cd /data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/
gzip ${sample[$i-1]}.rm_rRNA_1.fq
gzip ${sample[$i-1]}.rm_rRNA_2.fq
rm ${sample[$i-1]}.alingned_rRNA.txt

echo finish ${sample[$i-1]} `date`
done

echo remove_rRNA success `date`
```

Remove rRNA reads in CD1_3

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_3')

echo start remove_rRNA `date`

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part1/RNAseq/
mkdir -p 3.remove_rRNA/fastq/control/${sample[$i-1]}
mkdir -p 3.remove_rRNA/rRNA/control/${sample[$i-1]}

bowtie -n 0 -y -a --norc --best --strata -S -p 4 -l 15 \
--un
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA.fq \
/data/TA_QUIZ_RNA_regulation/data/ATH/index/bowtie1/rRNA/Arabidopsis_thaliana.TA
IR10.34.rRNA \
-1 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.1.fastq.gz \
-2 /data/user_03/RiboShape/Part1/RNAseq/2.quality_control/control/${sample[$i-
1]}/${sample[$i-1]}.clean.2.fastq.gz \
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.aligned_rRNA.txt

cd /data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/
gzip ${sample[$i-1]}.rm_rRNA_1.fq
gzip ${sample[$i-1]}.rm_rRNA_2.fq
rm ${sample[$i-1]}.alingned_rRNA.txt

echo finish ${sample[$i-1]} `date`
done

echo remove_rRNA success `date`
```

Mapping

Generate Genome Index

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH
mkdir -p /data/user_03/RiboShape/Part1/index/STAR/genome
STAR \
--runMode genomeGenerate \
--runThreadN 4 \
--genomeDir /data/user_03/RiboShape/Part1/index/STAR/genome \
--genomeFastaFiles
/data/TA_QUIZ_RNA_regulation/data/ATH/DNA/Arabidopsis_thaliana.TAIR10.dna.toplev
el.fa \
--sjdbGTFfile
/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/Arabidopsis_thaliana.TAIR10.34.gtf
```

Mapping & Samtools Sort and Index

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_1' 'CD1_2' 'CD1_3')

echo start mapping `date`
mkdir -p /data/user_03/RiboShape/Part1/RNAseq/differential_expression
for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part1/RNAseq/differential_expression
mkdir -p 4.mapping_expression/control/${sample[$i-1]}
cd 4.mapping_expression/control/${sample[$i-1]}

STAR \
--runThreadN 8 \
--limitBAMsortRAM 20000000000 \
--outFilterType BySJout \
--outFilterMismatchNmax 10  \
--genomeDir /data/user_03/RiboShape/Part1/index/STAR/genome \
--readFilesIn
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA_1.fq.gz \
/data/user_03/RiboShape/Part1/RNAseq/3.remove_rRNA/fastq/control/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA_2.fq.gz \
--readFilesCommand 'zcat' \
--outFileNamePrefix  ${sample[$i-1]} \
--outSAMtype BAM Unsorted \
--quantMode TranscriptomeSAM GeneCounts \
--outSAMattributes All  --outSAMstrandField intronMotif --outBAMcompression 6 --
outReadsUnmapped Fastx

samtools sort -T \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.out.sorted \
-o
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.sortedByCoord.out.bam \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.out.bam
```

```
samtools sort -T \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.toTranscriptome.out.sorted \
-o
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.toTranscriptome.out.sorted.bam
\
/data/user_03/A_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_expression/4.mapping_expression/control/${sample[$i-
1]}/${sample[$i-1]}Aligned.toTranscriptome.out.bam

samtools index \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.sortedByCoord.out.bam

samtools index \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.toTranscriptome.out.sorted.bam

echo finish ${sample[$i-1]} `date`
done

echo mapping success `date`
```

Get Read Counts

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_1' 'CD1_2' 'CD1_3')

echo start read_counts `date`

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`

cd /data/user_03/RiboShape/Part1/RNAseq/differential_expression
mkdir -p 5.read_counts/control/${sample[$i-1]}
cd 5.read_counts/control/${sample[$i-1]}

featureCounts \
-T 8 \
-s 0 \
-p -t CDS \
-g gene_id \
-a /data/TA_QUIZ_RNA_regulation/data/ATH/GTF/Arabidopsis_thaliana.TAIR10.34.gtf
\
-o
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/contr
ol/${sample[$i-1]}/${sample[$i-1]}.featurecounts.txt \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.sortedByCoord.out.bam

featureCounts \
-T 8 \
```

```
-s 0 \
-p -t exon \
-g gene_id \
-a /data/TA_QUIZ_RNA_regulation/data/ATH/GTF/Arabidopsis_thaliana.TAIR10.34.gtf
\
-o
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/contr
ol/${sample[$i-1]}/${sample[$i-1]}.featurecounts.all.txt \
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/4.mapping_expressio
n/control/${sample[$i-1]}/${sample[$i-1]}Aligned.sortedByCoord.out.bam

echo read_counts success
done
```

Merge Counts Matrix

Merge

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('CD1_1' 'CD1_2' 'CD1_3')

echo start read_counts `date`

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`

cd /data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts
mkdir -p result
cd 5.read_counts/result

echo -e "gene_id    ${sample[$i-1]}"
>//data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/res
ult/${sample[$i-1]}.txt
cat
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/contr
ol/${sample[$i-1]}/${sample[$i-1]}.featurecounts.txt| grep -v '#' | grep -v
'Geneid' | cut -f 1,7 >>
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/resul
t/${sample[$i-1]}.txt

echo -e "gene_id    ${sample[$i-1]}"
>//data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/res
ult/${sample[$i-1]}.all.txt
cat
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/contr
ol/${sample[$i-1]}/${sample[$i-1]}.featurecounts.all.txt| grep -v '#' | grep -v
'Geneid' | cut -f 1,7 >>
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/resul
t/${sample[$i-1]}.all.txt

echo finish ${sample[$i-1]} `date`
done
```

Get Count Matrix

Running Code:

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start 6.2.Count_matrix `date`

mkdir -p
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/exp_count_matrix
python /data/user_03/RiboShape/6.2.count_matrix.py

echo finish 6.2.Count_matrix `date`
```

Python Script:

```
import numpy as np
import pandas as pd
import re
import os

sample_WT=['CD1_1','CD1_2','CD1_3']
path =
'/data/user_03/RiboShape/Part1/RNAseq/differential_expression/5.read_counts/resu
lt/'
data_list=[]
for i in range(len(sample_WT)):
    print(sample_WT[i])
    temp=pd.read_csv(path+sample_WT[i]+'.all.txt',sep='\t',index_col=0)
    data_list.append(temp)
matrix_WT_C = data_list[0]
for i in range(len(data_list)-1):
    matrix_WT_C = matrix_WT_C.join(data_list[i+1],how="outer")
matrix_WT_C = matrix_WT_C.sort_index(axis=1)

matrix_WT_C.to_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expression
/exp_count_matrix/count_all.txt',sep='\t',index=True,header=True)
```

Data Analysis

Differential Expression

DEseq

Running code:

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start 7.DESeq `date`

mkdir -p /data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEseq/wt
mkdir -p
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEseq/uvr8

/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin/Rscript
/data/user_03/RiboShape/7.DEseq.R

echo finish 7.DESeq2 `date`
```

R script:

```
#!/data/zhaoyizi/software/anaconda3/envs/Riboshape/lib/R/bin/Rscript --vanilla
library(DESeq2)
library(tidyverse)

raw_count <- read.table("/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_expression/exp_count_matrix/count_all.txt",sep='\t',he
ader=T)

# 1) Input data
col0_raw_count <-
raw_count[c("gene_id","CD1_1","CD1_2","CD1_3","CD0_1","CD0_2","CD0_3")]
row.names(col0_raw_count) <- col0_raw_count[,1]
col0_raw_count <- col0_raw_count[,-1]
col0_raw_count <- col0_raw_count[rowSums(col0_raw_count)>100,]
countData_col0 <- col0_raw_count

# 2) Provide smaple information
condition_merge <-factor(c(rep("control",3),rep("KD",3)))
colData<-data.frame(row.names=colnames(countData_col0),condition_merge)

# 3）DGE analysis
dds <- DESeqDataSetFromMatrix(countData_col0,colData,design=~condition_merge)
dds2 <- DESeq(dds)
res <- results(dds2)

res <- res[order(res$padj),]
diff_gene_deseq_col0 <- subset(res,padj<0.05&(log2FoldChange>1|log2FoldChange<
-1))

col0_gene_names <-row.names(diff_gene_deseq_col0)
write.table(col0_gene_names,"/data/user_03/RiboShape/Part1/RNAseq/differential_e
xpression/7.DEseq/wt/wt_gene_list.txt",sep=',',col.names=F,row.names=F,quote=F)
resdata_col0 <-
merge(as.data.frame(res),as.data.frame(counts(dds)),by="row.names",sort=FALSE)
write.csv(resdata_col0,"/data/user_03/RiboShape/Part1/RNAseq/differential_expres
sion/7.DEseq/wt/wt_rawdata.csv",row.names=F,quote=F)
```

edgeR

Running code:

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start 7.edgeR `date`

mkdir -p /data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.edgeR/wt
mkdir -p
/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.edgeR/uvr8

/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin/Rscript
/data/user_03/RiboShape/7.EdgeR.R

echo finish 7.edgeR `date`
```

R script:

```
#!/data/zhaoyizi/software/anaconda3/envs/Riboshape/lib/R/bin/Rscript --vanilla
library(edgeR)
library(tidyverse)
library(limma)

raw_count <- read.table("/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_expression/exp_count_matrix/count_all.txt",sep='\t',he
ader=T)

col0_raw_count <-
raw_count[c("gene_id","CD1_1","CD1_2","CD1_3","CD0_1","CD0_2","CD0_3")]
row.names(col0_raw_count) <- col0_raw_count[,1]
col0_raw_count <- col0_raw_count[,-1]
col0_raw_count <- col0_raw_count[rowSums(col0_raw_count)>100,]

countData_col0 <- col0_raw_count

dgListGroups <- c(rep("Control",3),rep("Treat",3))
dgList <- DGEList(counts=countData_col0,
genes=rownames(countData_col0),group=factor(dgListGroups))

dgList <- calcNormFactors(dgList, method="TMM")
countsPerMillion <- cpm(dgList, normalized.lib.sizes=TRUE)

design.mat <- model.matrix(~0 + dgList$sample$group)
colnames(design.mat) <- levels(dgList$sample$group)

d2 <- estimateGLMCommonDisp(dgList, design=design.mat)
d2 <- estimateGLMTrendedDisp(d2, design=design.mat)
d2 <- estimateGLMTagwiseDisp(d2, design=design.mat)

fit <- glmFit(d2, design.mat)
lrt <- glmLRT(fit,contrast=c(-1,1))

edgeR_result <- topTags(lrt,n=nrow(dgList))
```

```
outfile <-
'/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.edgeR/wt/wt_rawd
ata.csv'
write.table(edgeR_result$table,file = outfile,sep = "\t",quote =
F,row.names=F,col.names=T)
```

Isolate up-regulated and down-regulated genes for KEGG pathway analysis【new】

Lists of up-regulated and down-regulated genes were generated respectively. Two scripts were created (with same function). KEGG is performed in the web. KEGG

```
#!/data/zhaoyizi/software/anaconda3/envs/Riboshape/lib/R/bin/Rscript --vanilla

data <-
read.csv("/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEseq/w
t/wt_rawdata.csv")

up_data <- data[which(data[,3]>1),]
down_data <- data[which(data[,3] < -1),]

write.csv(up_data,"/data/user_03/RiboShape/Final_Intergration/test/up_data.csv")
write.csv(down_data,"/data/user_03/RiboShape/Final_Intergration/test/down_data.c
sv")
```

```
#!/bin/sh

awk '{
while(getline)
{
    if ($3>1)
        print $1
}}' wt_rawdata.txt > up_try.txt
awk '{
while(getline)
{
    if ($3<(-1))
        print $1
}}' wt_rawdata.txt > down_try.txt
```

Differential Splicing

rMATS

```
#!/bin/sh

export
PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:/data/zhaoyizi/softwar
e/anaconda3/envs/Riboshape/rMATS:$PATH

echo start rMATS `date`
echo start wt.UVB-vs-noUVB `date`
mkdir /data/user_03/RiboShape/Part1/RNAseq/differential_splicing
cd /data/user_03/RiboShape/Part1/RNAseq/differential_splicing
mkdir -p wt.UVB-vs-noUVB

python /data/zhaoyizi/software/anaconda3/envs/Riboshape/rMATS/rmats.py \
```

```
--b1 /data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/wt.noUVB.txt \
--b2 /data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/wt.UVB.txt \
--gtf
/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/Arabidopsis_thaliana.TAIR10.34.gtf \
--od /data/user_03/RiboShape/Part1/differential_splicing/wt.UVB-vs-noUVB \
-t paired \
--readLength 151 \
--cstat 0.0001 \
--tmp /data/user_03/RiboShape/Part1/differential_splicing/wt.UVB-vs-noUVB/tmp \
--nthread 4 \
--variable-read-length

echo finish wt.UVB-vs-noUVB `date`

echo rMATS end  `date`
```

Processing Splicing Events

Running code:

```
#!/bin/bash

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH
cd /data/user_03/RiboShape/Part1/differential_splicing
mkdir -p wt.UVB-vs-noUVB_filtered_psi_0.1

python /data/user_03/RiboShape/9.splice_sig_psi.py PValue 0.05 0.1 wt.UVB-vs-
noUVB wt.UVB-vs-noUVB_filtered_psi_0.1

rename xls txt wt.UVB-vs-noUVB_filtered_psi_0.1/*.xls
```

Python script:

```
import os,re,sys
import pandas as pd
import numpy as np
filter_sig = sys.argv[1]
filter_sig_threshold = float(sys.argv[2])
psi_threshold = float(sys.argv[3])
input_dir = os.path.abspath(sys.argv[4])
output_dir = os.path.abspath(sys.argv[5])
stat_file = os.path.join(output_dir,"splicing.sig.stat.xls")
STAT = open(stat_file,"w")

for root,dirs,files in os.walk(input_dir):
    for input_file in files:
        if re.search("MATS.JC.txt",input_file) or
re.search("MATS.JCEC.txt",input_file):
            file_type = re.sub("\.MATS|\.txt","",input_file)
            input_file_path = os.path.join(root,input_file)
            os.system("sed -i 's/\"//g' "+input_file_path)
            df_splice =
pd.read_csv(input_file_path,sep="\t",header=0,index_col=False)
```

```
            df_splice_filtered = df_splice[df_splice[filter_sig] <=
filter_sig_threshold]
            psi_judge = np.abs(df_splice_filtered["IncLevelDifference"]) >=
psi_threshold
            df_splice_filtered = df_splice_filtered[psi_judge]
            output_file = os.path.join(output_dir,input_file)
            df_splice_filtered.to_csv(output_file,
sep="\t",header=True,index=False, na_rep="NA")
            splicing_sig_num = len(df_splice_filtered)
            STAT.write(file_type+"\t"+str(splicing_sig_num)+"\n")
```

Visualized Analysis With rmats2sashimiplot

```
#!/bin/bash

export PATH=/home/user_03/minconda3/bin:$PATH

inputDir="/data/user_03/RiboShape/Part1/differential_splicing/wt.UVB-vs-
noUVB_filtered_psi_0.1"
outDir="$inputDir/wt.UVB-vs-noUVB_sashimiplot_all"

echo  start rmats2sashimiplot `date`

for event in "SE" "A5SS" "A3SS" "MXE" "RI"
do

mkdir -p $outDir/$event
echo start processing $event

#python
/data/zhaoyizi/software/rmats2sashimiplot/src/rmats2sashimiplot/rmats2sashimiplo
t.py \
rmats2sashimiplot \
--b1 /data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD1_1/CD1_1.Aligne
d.sortedByCoord.out.bam,/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD1_2/CD1_2.Aligne
d.sortedByCoord.out.bam,/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD1_3/CD1_3.Aligne
d.sortedByCoord.out.bam \
--b2 /data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD0_1/CD0_1.Aligne
d.sortedByCoord.out.bam,/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD0_2/CD0_2.Aligne
d.sortedByCoord.out.bam,/data/TA_QUIZ_RNA_regulation/result/PartI.RNA-
seq_analysis/differential_splicing/3.mapping_splicing/control/CD0_3/CD0_3.Aligne
d.sortedByCoord.out.bam \
--l1 Control_wt_noUVB \
--l2 Treat_wt_UVB \
--exon_s 1 \
--intron_s 2 \
-t $event \
-e $inputDir/${event}.MATS.JCEC.txt \
-o $outDir/$event

done
```

```
rm -r $outDir/*/Sashimi_index*
echo finish rmats2sashimiplot `date`
```

Functional Analysis With KEGG

KEGG

Ribo-seq Analysis

Prepare Data Matrix

QC of Raw Data

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start fastqc `date`

sample=("CR1_1" "CR1_2" "CR1_3")

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 1.fastqc/${sample[$i-1]}
cd /data/user_03/RiboShape/Part2_Riboseq/1.fastqc/${sample[$i-1]}

fastqc -t 4 \
/data/user_03/RiboShape/Part2_Riboseq/prepare_data/${sample[$i-1]}.fq.gz \
--outdir /data/user_03/RiboShape/Part2_Riboseq/1.fastqc/${sample[$i-1]} --
noextract

echo finish ${sample[$i-1]} `date`
done

echo fastqc success `date`
```

Cut Adaptors and Trim Long Reads

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start trimmed

sample=("CR1_1" "CR1_2" "CR1_3")

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 2.quality_control/${sample[$i-1]}
```

```
cd /data/user_03/RiboShape/Part2_Riboseq/2.quality_control/${sample[$i-1]}

fastp --length_limit 50 \
--adapter_fasta /data/TA_QUIZ_RNA_regulation/data/ATH/Riboseq/fastp/adapters.fa
\
-i /data/user_03/RiboShape/Part2_Riboseq/prepare_data/${sample[$i-1]}.fq.gz \
-o /data/user_03/RiboShape/Part2_Riboseq/2.quality_control/${sample[$i-
1]}/${sample[$i-1]}.clean.fq.gz \
--thread=4 -l 15 -j ${sample[$i-1]}.json -h ${sample[$i-1]}.html

echo finish ${sample[$i-1]} `date`
done
echo trimmed success
```

Clean rRNA Reads

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start remove_rRNA `date`

sample=("CR1_1" "CR1_2" "CR1_3")

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 3.remove_rRNA/fastq/${sample[$i-1]}
mkdir -p 3.remove_rRNA/rRNA/${sample[$i-1]}

bowtie -y -a --norc --best --strata -S -p 4 -l 15 \
--un /data/user_03/RiboShape/Part2_Riboseq/3.remove_rRNA/fastq/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA.fq \
/data/TA_QUIZ_RNA_regulation/data/ATH/index/bowtie1/rRNA/Arabidopsis_thaliana.TA
IR10.34.rRNA \
-q /data/user_03/RiboShape/Part2_Riboseq/2.quality_control/${sample[$i-
1]}/${sample[$i-1]}.clean.fq.gz \
/data/user_03/RiboShape/Part2_Riboseq/3.remove_rRNA/fastq/${sample[$i-
1]}/${sample[$i-1]}.aligned_rRNA.txt

cd /data/user_03/RiboShape/Part2_Riboseq/3.remove_rRNA/fastq/${sample[$i-1]}/
gzip ${sample[$i-1]}.rm_rRNA.fq
rm /data/user_03/RiboShape/Part2_Riboseq/3.remove_rRNA/fastq/${sample[$i-
1]}/${sample[$i-1]}.aligned_rRNA.txt

echo finish ${sample[$i-1]} `date`
done
echo remove_rRNA success `date`
```

Mapping

Generating Genome Index

The index obtained from RNA-seq analysis is used here.

Mapping & Samtools Sort and Index

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH
echo start remove_rRNA `date`

sample=("CR1_1" "CR1_2" "CR1_3")

for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 4.mapping/${sample[$i-1]}
cd 4.mapping/${sample[$i-1]}

STAR \
--runThreadN 4 \
--outFilterType BySJout \
--outFilterMismatchNmax 2 \
--outFilterMultimapNmax 1 \
--genomeDir /data/TA_QUIZ_RNA_regulation/data/ATH/index/STAR/genome/ \
--readFilesIn
/data/user_03/RiboShape/Part2_Riboseq/3.remove_rRNA/fastq/${sample[$i-
1]}/${sample[$i-1]}.rm_rRNA.fq.gz \
--readFilesCommand 'zcat' \
--outFileNamePrefix ${sample[$i-1]}. \
--outSAMtype BAM SortedByCoordinate \
--quantMode TranscriptomeSAM GeneCounts \
--outSAMattributes All \
--outSAMattrRGline ID:1 LB:ribo_seq PL:ILLUMINA SM:${sample[$i-1]} \
--outBAMcompression 6 \
--outReadsUnmapped Fastx

samtools sort -T \
/data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-1]}/${sample[$i-
1]}.Aligned.toTranscriptome.out.sorted \
-o /data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-1]}/${sample[$i-
1]}.Aligned.toTranscriptome.out.sorted.bam \
/data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-1]}/${sample[$i-
1]}.Aligned.toTranscriptome.out.bam

samtools index /data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-
1]}/${sample[$i-1]}.Aligned.toTranscriptome.out.sorted.bam \

samtools index /data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-
1]}/${sample[$i-1]}.Aligned.sortedByCoord.out.bam

echo finish ${sample[$i-1]} `date`
done
```

Get Reads Counts

```
#!/bin/sh

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start read_counts `date`

sample=("CR1_1" "CR1_2" "CR1_3")
for i in $(seq 1 ${#sample[@]});do
echo start ${sample[$i-1]} `date`
cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 5.read_count_HTSeq/${sample[$i-1]}
cd 5.read_count_HTSeq/${sample[$i-1]}

htseq-count -f bam -s no -i gene_id -t CDS -m union \
/data/user_03/RiboShape/Part2_Riboseq/4.mapping/${sample[$i-1]}/${sample[$i-
1]}.Aligned.sortedByCoord.out.bam \
/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/Arabidopsis_thaliana.TAIR10.34.gtf
>/data/user_03/RiboShape/Part2_Riboseq/5.read_count_HTSeq/${sample[$i-
1]}/${sample[$i-1]}.read_count.HTSeq.txt

grep __ /data/user_03/RiboShape/Part2_Riboseq/5.read_count_HTSeq/${sample[$i-
1]}/${sample[$i-1]}.read_count.HTSeq.txt
>/data/user_03/RiboShape/Part2_Riboseq/5.read_count_HTSeq/${sample[$i-
1]}/${sample[$i-1]}.read_count.HTSeq.txt.summary

sed -i '/^__/d'
/data/user_03/RiboShape/Part2_Riboseq/5.read_count_HTSeq/${sample[$i-
1]}/${sample[$i-1]}.read_count.HTSeq.txt

echo finish ${sample[$i-1]} `date`
done

echo read success  `date`
```

Data Analysis

Periodicity and ORF Analysis

```
#!/bin/sh

cd /data/user_03/RiboShape/Part2_Riboseq
mkdir -p 6.RiboCode/metaplot
dataPath='/data/TA_QUIZ_RNA_regulation/result/PartII.Ribo-
seq_analysis/3.mapping'
outDir='/data/user_03/RiboShape/Part2_Riboseq/6.RiboCode/metaplot'
sample_list='/data/user_03/RiboShape/Part2_Riboseq/sample_list.txt'
RiboCode_annot='/data/TA_QUIZ_RNA_regulation/data/Ribocode/RiboCode_annot'

script="$outDir/script"
if [ ! -d $script ]
then
mkdir -p $script
fi
```

```
for i in `cat $sample_list`
do

echo -e "#!/bin/bash

echo start \`date\`

if [ ! -d $outDir/${i} ]
then
mkdir -p $outDir/${i}
fi
export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:\$PATH
metaplots -a $RiboCode_annot -r
$dataPath/${i}/${i}.Aligned.toTranscriptome.out.sorted.bam -o $outDir/$i. -m 26
-M 34 -s no -pv1 0.05 -pv2 0.05

echo end \`date\`
" > "$script/${i}.metaplots.sh"
done
```

Differential translation efficiency

Running code:

```
#!/bin/sh
name=$(hostname)
if [ ${name} == "tmgt" ]; then
    mkdir -p /data/user_03/RiboShape/Part2_Riboseq/7.TE
    /BioII/lulab_b/liuxiaofan/software/conda2/bin/Rscript
/data/user_03/RiboShape/Part2_Riboseq/9.xtail.R
else
    echo Now we will go to TMGT
    ssh tmgt "mkdir -p
/data/user_03/RiboShape/Part2_Riboseq/7.TE;/BioII/lulab_b/liuxiaofan/software/co
nda2/bin/Rscript /data/user_03/RiboShape/Part2_Riboseq/9.xtail.R"
fi
```

R Script:

```
library(xtail)

ribo <- read.table('/data/TA_QUIZ_RNA_regulation/result/PartII.Ribo-
seq_analysis/6.Differential_translation_efficiency/Ribo-
seq/WT_count.txt',header=T, quote='',check.names=F, sep='\t',row.names=1)
mrna <- read.table('/data/TA_QUIZ_RNA_regulation/result/PartII.Ribo-
seq_analysis/6.Differential_translation_efficiency/RNA-seq-
CDS/count_CDS.txt',header=T, quote='',check.names=F, sep='\t',row.names=1)

ribo <- ribo[,c("CR1_1","CR1_2","CR1_3","CR0_1","CR0_2","CR0_3")]
mrna <- mrna[c("CD1_1","CD1_2","CD1_3","CD0_1","CD0_2","CD0_3")]

condition <- c("control","control","control","treat","treat","treat")
results <- xtail(mrna,ribo,condition,minMeanCount=1,bins=10000)
results_tab <- resultsTable(results,sort.by="pvalue.adjust",log2FCs=TRUE,
log2Rs=TRUE)
write.table(results_tab,"/data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-
1.TE_new.xls",quote=F,sep="\t")
```

SHAPE Data Analysis

Prepare Data Matrix

Shapemapper

```
#!/bin/bash

export
PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:/data/liuxiaofan/softw
are/shapemap/:$PATH
mkdir -p /data/user_03/RiboShape/Part3_SHAPE/test.shapemapper
cd /data/user_03/RiboShape/Part3_SHAPE/test.shapemapper

/data/liuxiaofan/software/shapemap/shapemapper \
--target
/data/user_03/RiboShape/Part3_SHAPE/prepare_data/Arabidopsis_thaliana.TAIR10.34.
transcripts_new_2.fa \
--name "C1_1" \
--min-depth 100 \
--min-qual-to-count 20 \
--overwrite \
--modified --folder /data/user_03/RiboShape/Part3_SHAPE/prepare_data/modified \
--untreated --folder /data/user_03/RiboShape/Part3_SHAPE/prepare_data/control \
--star-aligner \
--nproc 8 \
--verbose
```

SHAPEMAPPER for AT1G09530.3【new】

```
#!/bin/bash

mkdir -p /data/user_03/RiboShape/Part3_SHAPE/shapemapper_AT1G09530.3
cd /data/user_03/RiboShape/Part3_SHAPE/shapemapper_AT1G09530.3
```

```
FA_FILE=/data/TA_QUIZ_RNA_regulation/data/ATH/transcript/Arabidopsis_thaliana.TAIR10.34.transcripts_new.fa
cat $FA_FILE|
awk '{
    do{
        if($0==">AT1G09530.3"){
            print $0;
            while(getline){
                if(index($0,/>*/)!=0)
                    break;
                print $0;
            }
        }
    }while(getline)
}'>AT1G09530.3.fa

echo .fa obtained

export
PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:/data/liuxiaofan/software/shapemap/:$PATH

/data/liuxiaofan/software/shapemap/shapemapper \
--target
/data/user_03/RiboShape/Part3_SHAPE/shapemapper_AT1G09530.3/AT1G09530.3.fa \
--name "C1_1" \
--min-depth 100 \
--min-qual-to-count 20 \
--overwrite \
--modified --folder /data/user_03/RiboShape/Part3_SHAPE/prepare_data/modified \
--untreated --folder /data/user_03/RiboShape/Part3_SHAPE/prepare_data/control \
--star-aligner \
--nproc 8 \
--verbose

echo finish
```

Reactivity Calculation for AT1G09530【new】

```
library(dplyr)
data=read.table("/data/user_03/RiboShape/Part3_SHAPE/shapemapper_AT1G09530.3/shapemapper_out/C1_1_AT1G09530.3_profile.txt",header=T)
data=mutate(data,Reactvity=Modified_mutations/Modified_effective_depth-
Untreated_mutations/Untreated_effective_depth)
data=filter(data,is.nan(Reactvity)!=T)
cat("The mean value of Reactivity is",mean(data$Reactivity),"\n")
```

Data Analysis

Structure Change Analysis

3 'UTR and 5' UTR sequence information of the TE changing region was extracted Calculate Hit Level

Running code:

```
#!/bin/bash

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

sample=('col_nouv' 'col_uv' )

for ((i=0;i<=3;i++));do
echo start ${sample[$i]} `date`
path0=/data/user_03/RiboShape/Part3_SHAPE
path1=/data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/final.modified_u
modified/${sample[$i]}
path2=/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/${sample[$i]}
mkdir -p $path2

cp $path1/final.modified_unmodified $path2/final.modified_unmodified

python $path0/33.hit_level.py \
--data_path $path1/final.modified_unmodified \
--savepath_hit $path2/final.modified_unmodified.hit

echo -e
"cutoff\ttranscript_id\tmodified.median\tunmodified.median\tmodified.sum\tunmodi
fied.sum\thit"       >       $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=25 && $2>0{print "0\t"$0}'
$path2/final.modified_unmodified.hit  >>        $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=50 && $2 > 25{print "25\t"$0}'
$path2/final.modified_unmodified.hit  >>        $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=100 && $2 > 50{print "50\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=200 && $2 > 100{print "100\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=300 && $2 > 200{print "200\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=500 && $2 >300{print "300\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=750 && $2 >500{print "500\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=1000 && $2 >750{print "750\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=2000 && $2 >1000{print "1000\t"$0}'
$path2/final.modified_unmodified.hit >>     $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2<=5000 && $2 >2000{print "2000\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;
awk -F '\t' '$3>0 && $2 > 5000{print "5000\t"$0}'
$path2/final.modified_unmodified.hit >>      $path2/cutoff.hit.group;

done
echo finish
```

Python Script:

```
import pandas as pd
import numpy as np
from numba import jit
import argparse

@jit(nopython=False)
def get_R(X,method='median'):
    name = X[0]

    modified = np.array(X[2].split(',')).astype('float').astype('int')
    modified_depth = np.array(X[3].split(',')).astype('float').astype('int')
    unmodified = np.array(X[4].split(',')).astype('float').astype('int')
    unmodified_depth = np.array(X[5].split(',')).astype('float').astype('int')

    modified_depth_median = np.median(modified_depth)
    unmodified_depth_median = np.median(unmodified_depth)
    modified_depth_sum = np.sum(modified_depth)
    unmodified_depth_sum = np.sum(unmodified_depth)
    hit = np.nan
    if len(modified) == 0:
        hit = np.nan
        out = np.nan
    else:
        if method == 'median':

            if np.median(unmodified_depth) == 0 :
                hit = np.nan
            else:
                hit = np.sum(modified) - (np.median(modified_depth) *
np.sum(unmodified))/np.median(unmodified_depth)
        elif method == 'mean':
            if np.mean(unmodified_depth) == 0:
                hit = np.nan
            else:
                hit = np.sum(modified) - (np.mean(modified_depth) *
np.sum(unmodified))/np.mean(unmodified_depth)
        out = hit/len(modified)

    return
pd.Series([name,modified_depth_median,unmodified_depth_median,modified_depth_sum
,unmodified_depth_sum,out])

@jit(nopython=False)
def get_data(data,method='median'):
    data_ = pd.DataFrame(np.zeros([len(data), 6]))
    for i in range(len(data)):
        if i %1000 == 0:
            print(i)
        A = np.array(get_R(data.iloc[i,:],method=method))
        data_.iloc[i,:]=A
    return data_

def hit_level(args):

    data = pd.read_csv(args.data_path, sep='\t')
```

```
    data.columns = ['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                    'Untreated_mutations', 'Untreated_effective_depth']
    data =data[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                    'Untreated_mutations', 'Untreated_effective_depth']]
    data_hit =  get_data(data,method=args.method)

    data_hit.to_csv(args.savepath_hit,sep='\t',index=False,header=None)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Feature selection module')

    parser.add_argument('--data_path', type=str, required=True,
                        help= 'data_path', dest='data_path')
    parser.add_argument('--savepath_hit', type=str, required=True,
                        help='savepath_hit', dest='savepath_hit')
    parser.add_argument('--method', type=str, default="median",
                        help='method', dest='method')

    args = parser.parse_args()
    hit_level(args)
```

Draw the Trend Chart of the Number of Transcripts Changing with HIT Level

Running code:

```
#!/bin/bash

export
PATH=/home/user_03/miniconda3/bin:/data/zhaoyizi/software/anaconda3/envs/Ribosha
pe/bin:$PATH

echo start calculate_hit_level `date`
mkdir -p /data/user_03/RiboShape/Part3_SHAPE/hit_level_plot/
path0=/data/user_03/RiboShape/Part3_SHAPE
python $path0/34.hit_level2.py

echo finish calculate_hit_level `date`
```

Python script:

```
import numpy as np
import pandas as pd
import re
import seaborn as sns
import matplotlib.pyplot as plt
from numpy import median
from numba import jit

exon_gtf_path='/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/shape_map/ath_exons.gtf
'
col_uv_f_path='/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_
nouv/'
col_uv_z_path='/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_
uv/'
```

```
gtf_data=pd.read_csv(exon_gtf_path,sep='\t',header=None)
gtf_data_new=pd.DataFrame(columns=
{'transcript_id','gene_type','gene_name','chr'})
gtf_data_new['transcript_id']=gtf_data.iloc[:,8].apply(lambda x:x.split(';')
[1].split('"')[1])
gtf_data_new['gene_type'] = gtf_data.iloc[:,8].apply(lambda
x:re.findall('gene_biotype ".*?"',x)[0].split('"')[1])
gtf_data_new['gene_name'] = gtf_data.iloc[:,8].apply(lambda
x:re.findall('gene_name ".*?"',x)[0].split('"')[1] if 'gene_name' in x else
np.nan)
gtf_data_new['chr'] = gtf_data.iloc[:,0].apply(lambda x: 6 if x=='Mt' else 7 if
x=='Pt' else x ).astype('int')
gtf_data_new = gtf_data_new.drop_duplicates()
gtf_data_new.index = range(len(gtf_data_new))

hit_level_col_uv_f =
pd.read_csv(col_uv_f_path+'/final.modified_unmodified.hit',sep='\t',header=None)
hit_level_col_uv_f.columns =
['transcript_id','modified.median','unmodified.median','modified.sum','unmodifie
d.sum','hit']
hit_level_col_uv_f =
pd.merge(hit_level_col_uv_f,gtf_data_new,on='transcript_id',how='left')
hit_level_col_uv_f['Type'] = 'WT_UV-'

hit_level_col_uv_z =
pd.read_csv(col_uv_z_path+'/final.modified_unmodified.hit',sep='\t',header=None)
hit_level_col_uv_z.columns =
['transcript_id','modified.median','unmodified.median','modified.sum','unmodifie
d.sum','hit']
hit_level_col_uv_z =
pd.merge(hit_level_col_uv_z,gtf_data_new,on='transcript_id',how='left')
hit_level_col_uv_z['Type'] = 'WT_UV+'

data1=pd.DataFrame(columns={'Type','Number of transcripts','hit'})
for num in [-1000,0,1,2,5,10,15]:
    WT_UV_f = len(hit_level_col_uv_f.loc[(hit_level_col_uv_f['hit'] > num) &
(hit_level_col_uv_f['modified.median'] > 100), :])
    WT_UV_z = len(hit_level_col_uv_z.loc[(hit_level_col_uv_z['hit'] > num) &
(hit_level_col_uv_z['modified.median'] > 100)])

    data2 = pd.DataFrame(columns={'Type', 'Number of transcripts', 'hit'})
    data2['Type'] = ['WT_UV-', 'WT_UV+']
    data2['Number of transcripts'] = [WT_UV_f, WT_UV_z]

    if num==-1000:
        data2['hit'] ='ALL transcripts'
    else:
        data2['hit']='Hit level>'+str(num)
    data1 = pd.concat([data1,data2])

plt.switch_backend('agg')
plt.figure(figsize=(8, 6))
sns.set(style="ticks", context="talk")
sns.axes_style({'font.family': ['sans-serif'],'font.sans-serif': ['Arial']})
g = sns.barplot(y='Number of transcripts',x='hit',hue='Type',data=data1,palette=
["#3498DB", "#1F618D"])

sns.despine()
```

```
font1 = {'family' : 'Arial','weight' : 'roman','size': 22}
plt.xticks(rotation=60)

plt.legend(fontsize='small')
plt.tight_layout()
#因为生成的PNG文件不能在windows下打开，所以我们改为生成svg文件，这样就可以用浏览器打开这个文件
plt.savefig('/data/user_03/RiboShape/Part3_SHAPE/hit_level_plot/2a.transcript_num_hit.svg')
plt.close()
```

Calculate the Normalization Factor

Running code:

```
#!/bin/bash

export
PATH=/home/user_03/miniconda3/bin:/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start normalization_factor `date`

path0=/data/user_03/RiboShape/Part3_SHAPE
python $path0/35.normalization_factor.py

echo finish normalization_factor `date`
```

Python script:

```
import numpy as np
import pandas as pd
import re
import matplotlib
matplotlib.use('Agg')
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns

exon_gtf_path='/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/shape_map/ath_exons.gtf'
col_uv_f_path='/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_nouv/'
col_uv_z_path='/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_uv/'

gtf_data=pd.read_csv(exon_gtf_path,sep='\t',header=None)
gtf_data_new=pd.DataFrame(columns=
{'transcript_id','gene_type','gene_name','chr'})
gtf_data_new['transcript_id']=gtf_data.iloc[:,8].apply(lambda x:x.split(';')
[1].split('"')[1])
gtf_data_new['gene_type'] = gtf_data.iloc[:,8].apply(lambda
x:re.findall('gene_biotype ".*?"',x)[0].split('"')[1])
gtf_data_new['gene_name'] = gtf_data.iloc[:,8].apply(lambda
x:re.findall('gene_name ".*?"',x)[0].split('"')[1] if 'gene_name' in x else
np.nan)
gtf_data_new['chr'] = gtf_data.iloc[:,0].apply(lambda x: 6 if x=='Mt' else 7 if
x=='Pt' else x ).astype('int')
```

```
gtf_data_new = gtf_data_new.drop_duplicates()
gtf_data_new.index = range(len(gtf_data_new))

hit_level_col_uv_f = pd.read_csv(col_uv_f_path+'/cutoff.hit.group',sep='\t')
hit_level_col_uv_f.columns =
['group','transcript_id','modified.median','unmodified.median','modified.sum','u
nmodified.sum','hit']
hit_level_col_uv_f =
pd.merge(hit_level_col_uv_f,gtf_data_new,on='transcript_id',how='left')
hit_level_col_uv_f['spe'] = 'WT_UV-'

hit_level_col_uv_z = pd.read_csv(col_uv_z_path+'/cutoff.hit.group',sep='\t')
hit_level_col_uv_z.columns =
['group','transcript_id','modified.median','unmodified.median','modified.sum','u
nmodified.sum','hit']
hit_level_col_uv_z =
pd.merge(hit_level_col_uv_z,gtf_data_new,on='transcript_id',how='left')
hit_level_col_uv_z['spe'] = 'WT_UV+'

col_uv_f_id =
hit_level_col_uv_f.loc[(hit_level_col_uv_f.gene_type.isin(['lncRNA','rRNA','tRNA
']))&(hit_level_col_uv_f['modified.median']>5000)&
(hit_level_col_uv_f['hit']>10),'transcript_id']
col_uv_z_id =
hit_level_col_uv_z.loc[(hit_level_col_uv_z.gene_type.isin(['lncRNA','rRNA','tRNA
']))&(hit_level_col_uv_z['modified.median']>5000)&
(hit_level_col_uv_z['hit']>10),'transcript_id']

print (col_uv_f_id)
print (col_uv_z_id)
print (set(col_uv_f_id)&set(col_uv_z_id))
```

Preprocessing and Normalization of Reactivity

Running code:

```
#!/bin/bash

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start data_normalization `date`

path0=/data/user_03/RiboShape/Part3_SHAPE
python $path0/36.data_normalization.py

echo finish data_normalization `date`
```

Python script:

```
import numpy as np
import pandas as pd
import re
from numba import jit
import math

@jit(nopython=False)
def calc_quartile(x, q, qtype=7):
```

```
    y = np.copy(x)
    n = len(y)
    abcd = [(0, 0, 1, 0),  # inverse empirical distrib.function., R type 1
            (0.5, 0, 1, 0),  # similar to type 1, averaged, R type 2
            (0.5, 0, 0, 0),  # nearest order statistic,(SAS) R type 3
            (0, 0, 0, 1),  # California linear interpolation, R type 4
            (0.5, 0, 0, 1),  # hydrologists method, R type 5
            (0, 1, 0, 1),  # mean-based estimate(Weibull method),
(SPSS,Minitab), type 6
            (1, -1, 0, 1),  # mode-based method,(S, S-Plus), R type 7
            (1.0 / 3, 1.0 / 3, 0, 1),  # median-unbiased ,  R type 8
            (3 / 8.0, 0.25, 0, 1)  # normal-unbiased, R type 9.
            ]
    a, b, c, d = abcd[qtype - 1]
    g, j = math.modf(a + (n + b) * q - 1)

    if j < 0:
        return x[0]
    elif j >= n:
        return x[n - 1]
    j = int(math.floor(j))
    if g == 0:
        return x[j]
    else:
        return y[j] + (y[j + 1] - y[j]) * (c + d * g)

def find_boxplot_factor(array):
    x, o, a = [], [], 0
    x = array[np.where(np.isfinite(array))]
    x = x[np.nonzero(x)]

    if x.shape[0] < 10:
        norm_factor = np.nan
    else:
        x.sort()
        ten_pct = len(x) // 10
        five_pct = len(x) // 20

        q_limit = 1.5 * abs(calc_quartile(x, 0.25) - calc_quartile(x, 0.75))
        ten_limit = x[x.shape[0] - 1 - ten_pct]
        five_limit = x[x.shape[0] - 1 - five_pct]

        limit = max(q_limit, ten_limit)
        if len(x) < 100:
            limit = max(q_limit, five_limit)

        for i in range(len(x)):
            if x[i] < limit:
                o.append(x[i])

        try:
            for i in range(-ten_pct, 0):
                a = o[i] + a
            norm_factor = a / ten_pct
        except IndexError:
            norm_factor = np.nan
    return norm_factor
```

```
@jit(nopython=False)
def find_boxplot_factor_new(array):

    x = array[np.where(np.isfinite(array))]
    x = x[np.nonzero(x)]
    if x.shape[0] < 10:
        norm_factor = np.nan
    else:
        o = x[(x>=np.quantile(x,0.92))&(x<=np.quantile(x,0.98))]
        norm_factor = np.mean(o)
    return norm_factor


@jit(nopython=False)
def
filter(X,Mutru_cut_off=0.02,read_depth_cutoff=100,R_window=10,window_mutru_cutof
f=0.03,window_mutru_num=3,window_mutrs_cutoff=0.1,window_mutrs_num=3):

    name = X[0]
    nucleotide = X[1]
    modified = np.array(X[2].split(',')).astype('float').astype('int')
    modified_depth = np.array(X[3].split(',')).astype('float').astype('int')
    unmodified = np.array(X[4].split(',')).astype('float').astype('int')
    unmodified_depth = np.array(X[5].split(',')).astype('float').astype('int')

    Mutrs = modified/modified_depth
    Mutru = unmodified/unmodified_depth
    R = Mutrs - Mutru
    n = len(modified)

    R[Mutru>Mutru_cut_off] = np.nan
    R[(modified_depth <= read_depth_cutoff)|(unmodified_depth)
<=read_depth_cutoff] = np.nan
    R[R<0]= np.nan
    for i in range(len(R)-R_window+1):
        data_Mutru = Mutru[i:i+R_window-1]
        data_Mutrs = Mutrs[i:i+R_window-1]
        if (len(data_Mutru[data_Mutru>window_mutru_cutoff])>=window_mutru_num)|
(len(data_Mutrs[data_Mutrs>window_mutrs_cutoff])>=window_mutrs_num):
            R[i:i + R_window-1] = np.nan
    R_new = ",".join(list(R.astype('str'))).replace('nan','-999')
    X = X.append(pd.Series(R_new))
    return X

@jit(nopython=False)
def get_fatcor(data):
    R_all = np.array(data.iloc[0,6].split(',')).astype('float')
    for i in range(1,len(data)):
        R_ = np.array(data.iloc[i,6].split(',')).astype('float')
        R_all = np.hstack((R_all,R_))
    R_all[R_all==-999]=np.nan
    factor = find_boxplot_factor_new(R_all)
    return factor

@jit(nopython=False)
def normalization_all(X,factor):

    R = np.array(X[6].split(',')).astype('float')
```

```
    R_nan = len(R[R == -999])/len(R)
    R_0 = len(R[R == 0])/len(R)
    R[R == -999] = np.nan
    R_new = R/factor
    R_new = ",".join(list(R_new.astype('str'))).replace('nan','-999')
    X[6] =R_new
    return X,R_nan,R_0
def main(data,transcript_list=[]):

    data_filter = pd.DataFrame(np.zeros([len(data),7]))
    data_filter.columns = ['transcript_id','Nucleotide' ,'Modified_mutations',
'Modified_effective_depth',
                           'Untreated_mutations', 'Untreated_effective_depth',
'reactivity']
    data_new = pd.DataFrame(np.zeros([len(data), 7]))
    data_new.columns = ['transcript_id','Nucleotide' ,'Modified_mutations',
'Modified_effective_depth',
                        'Untreated_mutations', 'Untreated_effective_depth',
'reactivity']
    print(len(data))
    for i in range(len(data)):
        if i %1000 == 0:
            print(i)
        X=filter(data.iloc[i,:])
        data_filter.iloc[i,:] = list(X)
    if len(transcript_list)> 0:
        factor =
get_fatcor(data_filter.loc[data_filter.transcript_id.isin(transcript_list),:])
    else:
        factor = get_fatcor(data_filter)
    for i in range(len(data_filter)):
        if i %1000 == 0:
            print(i)

        data_new.iloc[i,:],R_nan,R_0 =
normalization_all(data_filter.iloc[i,:],factor)

    data_new = data_new[['transcript_id','Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                        'Untreated_mutations', 'Untreated_effective_depth',
'reactivity']]
    return data_new

def loctaion(X):
    X_1 = X.split('(')[1].split(')')[0].split(',')
    loctaion_list = []
    for i in range(len(X_1)):
        loctaion_list_ = [i for i in range(int(X_1[i].split(':')
[0]),int(X_1[i].split(':')[1])+1)]
        loctaion_list.extend(loctaion_list_)
    loctaion_list =np.array(loctaion_list)
    loctaion_list = ",".join(list(loctaion_list.astype('str')))

    return loctaion_list



def mapping(exon_data):
```

```
    data_location = pd.DataFrame(np.zeros([len(exon_data),4]))
    data_location.columns = ['transcript_id','chr','strand','location']

    for i in range(len(exon_data)):
        if i%1000 ==0:
            print(i)
        data_location.loc[i, 'transcript_id'] = exon_data.loc[i,
'transcript_id']
        data_location.loc[i,'chr'] = exon_data.loc[i,'chr'].split('.')[0]
        data_location.loc[i, 'strand'] = exon_data.loc[i, 'chr'].split('.')[1]
        data_location.loc[i, 'location'] = loctaion(exon_data.loc[i,
'start_end'])
    return data_location

if __name__ == '__main__':
    col_uv_f =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_nouv/'
    col_uv_z =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_uv/'
    path=[col_uv_f,col_uv_z]
    for i in range(len(path)):
        transcript_list = [ 'AT3G41768.1', 'AT3G06355.1', 'ATMG01390.1' ]
        data_uv_z = pd.read_csv(path[i] + '/final.modified_unmodified',
sep='\t')
        data_uv_z.columns = ['transcript_id', 'Nucleotide',
'Modified_mutations', 'Modified_effective_depth','Untreated_mutations',
'Untreated_effective_depth']
        data_uv_z_new=main(data_uv_z,transcript_list)
        print(data_uv_z_new)

 data_uv_z_new.to_csv(path[i]+'/final.modified_unmodified_new',sep='\t',header=T
rue,index=False)
```

Calculate Gini Index

Running code:

```
#!/bin/bash

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start calculate_gini_index `date`
mkdir -p /data/user_03/RiboShape/Part3_SHAPE/gini_index
path0=/data/user_03/RiboShape/Part3_SHAPE
python $path0/37.calculate_gini_index.py

echo finish calculate_gini_index `date`
```

Python script:

```
########################计算gini_index##############################
import numpy as np
import pandas as pd
import re
from numba import jit
from scipy.stats import ks_2samp
import statsmodels.stats.multitest as multi
```

```
jit(nopython=False)
def get_gini(R_, nucleotide_, cut_AC=False, ratio_nan=0.9, ratio_nan_AC=0.8):
    ########
    if len(R_)==0:
        gini=np.nan
    else:
        R_nonull = R_[np.isnan(R_) == False]
        # R_nonull = R_[R_>0]
        ratio = len(R_nonull) / len(R_)
        if ratio <= ratio_nan:
            gini = np.nan
        else:
            if cut_AC:
                R_AC = R_[(nucleotide_ == b'A') | (nucleotide_ == b'C')]
                R_AC_nonull = R_AC[np.isnan(R_AC) == False]
                ratio_AC = len(R_AC_nonull) / len(R_AC)
                if (ratio_AC <= ratio_nan_AC) | len(R_AC) <= 1 |
len(R_AC_nonull) <= 1:
                    gini = np.nan
                else:
                    sorted = np.sort(R_AC_nonull)
                    height, area = 0, 0
                    for i in range(0, len(sorted)):
                        height += sorted[i]
                        area += height - sorted[i] / 2.
                    fair_area = height * len(sorted) / 2.
                    if fair_area == 0:
                        gini = np.nan
                    else:
                        gini = (fair_area - area) / fair_area
            else:
                sorted = np.sort(R_nonull)
                height, area = 0, 0
                for i in range(0, len(sorted)):
                    height += sorted[i]
                    area += height - sorted[i] / 2.
                fair_area = height * len(sorted) / 2.
                if fair_area == 0:
                    gini = np.nan
                else:
                    gini = (fair_area - area) / fair_area
    return gini

@jit(nopython=False)
def get_p(R_1,R_2,ratio_nan=0.9):
    R_1_nonull = R_1[np.isnan(R_1) == False]
    ratio_1 = len(R_1_nonull) / len(R_1)
    R_2_nonull = R_2[np.isnan(R_2) == False]
    ratio_2 = len(R_2_nonull) / len(R_2)
    if (ratio_1 <= ratio_nan)|(ratio_2 <= ratio_nan):
        p = np.nan
    else:
        s,p=ks_2samp(R_1, R_2)
    return p

@jit(nopython=False)
```

```
def get_window(X, window=50, step=1, cut_AC=False, ratio_nan=0.9,
ratio_nan_AC=0.8):

    name = X[0]
    nucleotide = np.array(list(X[1]))
    modified = np.array(X[2].split(',')).astype('float').astype('int')
    modified_depth = np.array(X[3].split(',')).astype('float').astype('int')
    unmodified = np.array(X[4].split(',')).astype('float').astype('int')

    unmodified_depth = np.array(X[5].split(',')).astype('float').astype('int')
    Mutrs = modified / modified_depth
    Mutru = unmodified / unmodified_depth
    # R = Mutrs - Mutru
    R = np.array(X[6].split(',')).astype('float')
    R[R == -999] = np.nan
    n = len(nucleotide)

    nucleotide_ = np.zeros([len(range(0, n - (window), step)), window],
dtype=np.string_)
    R_ = np.zeros([len(range(0, n - (window), step)), window])
    gini = np.zeros([len(range(0, n - (window), step)), ])
    j = 0
    for i in range(0, n - (window), step):
        nucleotide_[j, :] = nucleotide[i:i + window]
        R_[j, :] = R[i:i + window]
        gini[j] = get_gini(R[i:i + window], nucleotide[i:i + window],
cut_AC=cut_AC, ratio_nan=ratio_nan,
                           ratio_nan_AC=ratio_nan_AC)
        j = j + 1

    return nucleotide_, R_, gini

@jit(nopython=False)
def get_window_p(X1,X2, window=50, step=1,ratio_nan=0.9):
    name = X1[0]
    nucleotide = np.array(list(X1[1]))
    R_z = np.array(X1[6].split(',')).astype('float')
    R_z[R_z == -999] = np.nan
    R_f = np.array(X2[6].split(',')).astype('float')
    R_f[R_f == -999] = np.nan
    n = len(nucleotide)

    nucleotide_ = np.zeros([len(range(0, n - (window), step)), window],
dtype=np.string_)
    R_z_ = np.zeros([len(range(0, n - (window), step)), window])
    R_f_ = np.zeros([len(range(0, n - (window), step)), window])
    p = np.zeros([len(range(0, n - (window), step)), ])
    j = 0
    for i in range(0, n - (window), step):
        nucleotide_[j, :] = nucleotide[i:i + window]
        R_z_[j, :] = R_z[i:i + window]
        R_f_[j, :] = R_f[i:i + window]
        p[j] = get_p(R_z[i:i + window], R_f[i:i + window],ratio_nan=ratio_nan,)
        j = j + 1
    p_=p.copy()
    if len(p[~np.isnan(p)])>0:
        a,p_bh,b,c =multi.multipletests(p[~np.isnan(p)],method='fdr_bh')
        p_[~np.isnan(p_)]=p_bh
```

```
    return p_,p

@jit(nopython=False)
def calculate_delta_gini(R_1, R_2, gini_1, gini_2, ratio_nan=0.9):
    '''Calculates Standard RMSD on two vectors of numbers of the same length'''
    if len(R_1) != len(R_2):
        return np.nan
    else:
        R = R_1 - R_2
        if len(R[np.isnan(R) == False]) / len(R) <= ratio_nan:
            return np.nan
        else:
            delta_gini = gini_1 - gini_2
    return delta_gini

@jit(nopython=False)
def get_all(X,cut_AC=False, ratio_nan=0, ratio_nan_AC=0.8):
    #########读取数据##############
    name = X[0]
    nucleotide = np.array(list(X[1]))
    R = np.array(X[6].split(',')).astype('float')
    R[R == -999] = np.nan
    n = len(nucleotide)
    gini= get_gini(R, nucleotide, cut_AC=cut_AC,
ratio_nan=ratio_nan,ratio_nan_AC=ratio_nan_AC)
    return gini

@jit(nopython=False)
def get_se(X,location ,start,end,strand,cut_AC=False, ratio_nan=0,
ratio_nan_AC=0.8):
    #########读取数据##############
    name = X[0]
    nucleotide = np.array(list(X[1]))
    R = np.array(X[6].split(',')).astype('float')
    R[R == -999] = np.nan
    n = len(nucleotide)
    if strand=='+':
        R_=R[location<start]
        nucleotide_=nucleotide[location<start]
        gini_5= get_gini(R_, nucleotide_, cut_AC=cut_AC,
ratio_nan=ratio_nan,ratio_nan_AC=ratio_nan_AC)
        R_ = R[location > end]
        nucleotide_ = nucleotide[location > end]
        gini_3 = get_gini(R_, nucleotide_, cut_AC=cut_AC, ratio_nan=ratio_nan,
ratio_nan_AC=ratio_nan_AC)
    else:
        R_ = R[location < start]
        nucleotide_ = nucleotide[location < start]
        gini_3 = get_gini(R_, nucleotide_, cut_AC=cut_AC, ratio_nan=ratio_nan,
ratio_nan_AC=ratio_nan_AC)
        R_ = R[location > end]
        nucleotide_ = nucleotide[location > end]
        gini_5 = get_gini(R_, nucleotide_, cut_AC=cut_AC, ratio_nan=ratio_nan,
ratio_nan_AC=ratio_nan_AC)
    R_ = R[(location<= end)&(location>=start)]
    nucleotide_ = nucleotide[(location<= end)&(location>=start)]
    gini_cds = get_gini(R_, nucleotide_, cut_AC=cut_AC, ratio_nan=ratio_nan,
ratio_nan_AC=ratio_nan_AC)
```

```
    return gini_3,gini_cds,gini_5


@jit(nopython=False)
def get_statistics(data_z, data_f,data_gff_,location_list,strand):
    data_z = np.array(data_z).reshape([7, ])
    data_f = np.array(data_f).reshape([7, ])
    nucleotide_z, R_z, gini_z = get_window(data_z)
    nucleotide_f, R_f, gini_f = get_window(data_f)
    ###uv+###
    gini_z_=gini_z[pd.notnull(gini_z)]
    if len(gini_z_)==0:
        gini_max_z=np.nan
    else:
        gini_max_z=gini_z[pd.notnull(gini_z)].max()
    gini_all_z=get_all(data_z)
    if len(data_gff_.loc[data_gff_['location']=='CDS','strat'])==0:
        gini_3_z=np.nan
        gini_5_z=np.nan
        gini_cds_z=np.nan
    else:
        CDS_strat=list(data_gff_.loc[data_gff_['location']=='CDS','strat'])[0]
        CDS_end = list(data_gff_.loc[data_gff_['location']=='CDS','end'])[0]
        gini_3_z, gini_cds_z, gini_5_z =
get_se(data_z,location_list,CDS_strat,CDS_end,strand)
    ###uv-###
    gini_f_ = gini_f[pd.notnull(gini_f)]
    if len(gini_f_) == 0:
        gini_max_f = np.nan
    else:
        gini_max_f = gini_f[pd.notnull(gini_f)].max()
    gini_all_f = get_all(data_f)
    if len(data_gff_.loc[data_gff_['location'] == 'CDS', 'strat']) == 0:
        gini_3_f = np.nan
        gini_5_f = np.nan
        gini_cds_f = np.nan
    else:
        CDS_strat = list(data_gff_.loc[data_gff_['location'] == 'CDS', 'strat'])
[0]
        CDS_end = list(data_gff_.loc[data_gff_['location'] == 'CDS', 'end'])[0]
        gini_3_f, gini_cds_f, gini_5_f = get_se(data_f, location_list,
CDS_strat, CDS_end, strand)

    if len(R_z) != len(R_f):
        print("error")
    else:
        delta_gini = np.zeros([len(R_z), ])
        for i in range(len(R_z)):
            delta_gini[i] = calculate_delta_gini(R_z[i, :], R_f[i, :],
gini_z[i], gini_f[i])
    return delta_gini,gini_max_z,gini_all_z,gini_3_z, gini_cds_z,
gini_5_z,gini_max_f,gini_all_f,gini_3_f, gini_cds_f, gini_5_f


def main_sum_gini(data_uv_z, data_uv_f, data_location, data_gff,
transcript_id_list):
    statistics_sum = pd.DataFrame(columns={'transcript_id', 'num',
'num_0.1','delta_max', 'delta_min'})
```

```
    statistics_z = pd.DataFrame(columns={'transcript_id', 'gini_all',
'gini_max', 'gini_3UTR', 'gini_5UTR', 'gini_CDS'})
    statistics_f = pd.DataFrame(columns={'transcript_id', 'gini_all',
'gini_max', 'gini_3UTR', 'gini_5UTR', 'gini_CDS'})
    i = 0
    for transcript in transcript_id_list:
        data_z = data_uv_z.loc[data_uv_z['transcript_id'] == transcript, :]
        data_f = data_uv_f.loc[data_uv_f['transcript_id'] == transcript, :]
        data_location_ = data_location.loc[data_location['transcript_id'] ==
transcript,:]
        data_gff_ = data_gff.loc[data_gff['transcript_id'] == transcript, :]
        location_list=np.array(list(data_location_['location'])
[0].split(',')).astype('int')
        chr = list(data_location_['chr'])[0]
        strand=list(data_location_['strand'])[0]
        delta_gini, gini_max_z, gini_all_z, gini_3_z, gini_cds_z, gini_5_z,
gini_max_f, gini_all_f, gini_3_f, gini_cds_f, gini_5_f=
get_statistics(data_z,data_f, data_gff_, location_list, strand)

        statistics_sum.loc[i, 'transcript_id'] = transcript
        num = sum(np.abs(delta_gini) >=0.2)
        statistics_sum.loc[i, 'num'] = num
        num_1 = sum(np.abs(delta_gini) >=0.1)
        statistics_sum.loc[i, 'num_0.1'] = num_1
        statistics_sum.loc[i, 'delta_gini_list'] =
','.join(list(delta_gini.astype('str')))

        if len(delta_gini[np.isnan(delta_gini) == False]) > 0:
            statistics_sum.loc[i, 'delta_max'] =
np.max(delta_gini[np.isnan(delta_gini) == False])
            statistics_sum.loc[i, 'delta_min'] =
np.min(delta_gini[np.isnan(delta_gini) == False])
        else:
            statistics_sum.loc[i, 'delta_max'] = np.nan
            statistics_sum.loc[i, 'delta_min'] = np.nan

        statistics_z.loc[i,'transcript_id']=transcript
        statistics_z.loc[i,'gini_all']=gini_all_z
        statistics_z.loc[i, 'gini_max'] = gini_max_z
        statistics_z.loc[i, 'gini_3UTR'] = gini_3_z
        statistics_z.loc[i, 'gini_5UTR'] = gini_5_z
        statistics_z.loc[i, 'gini_CDS'] = gini_cds_z

        statistics_f.loc[i,'transcript_id']=transcript
        statistics_f.loc[i,'gini_all']=gini_all_f
        statistics_f.loc[i, 'gini_max'] = gini_max_f
        statistics_f.loc[i, 'gini_3UTR'] = gini_3_f
        statistics_f.loc[i, 'gini_5UTR'] = gini_5_f
        statistics_f.loc[i, 'gini_CDS'] = gini_cds_f
        i = i + 1
        if i % 100 == 0:
            print(i)
    return statistics_sum,statistics_z,statistics_f

if __name__ == '__main__':
    col_uv_f =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_nouv/'
```

```
    col_uv_z =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_uv/'

###col###
    result = '/data/user_03/RiboShape/Part3_SHAPE/gini_index'
    uv_z = col_uv_z
    uv_f = col_uv_f

    data_uv_z = pd.read_csv(uv_z + '/final.modified_unmodified_new', sep='\t')
    data_uv_z = data_uv_z[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                           'Untreated_mutations', 'Untreated_effective_depth',
'reactivity']]
    data_uv_z = data_uv_z.drop_duplicates()

    data_uv_f = pd.read_csv(uv_f + '/final.modified_unmodified_new', sep='\t')
    data_uv_f = data_uv_f[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                           'Untreated_mutations', 'Untreated_effective_depth',
'reactivity']]
    data_uv_f = data_uv_f.drop_duplicates()
    transcript_uv_z = pd.read_csv(uv_z + '/cutoff.hit.group', sep='\t')
    transcript_uv_z.columns = ['cut_off', 'transcript_id',
'modified_depth_median',
                               'unmodified_depth_median', 'modified_depth_sum',
'unmodified_depth_sum', 'hit_level']

    transcript_uv_f = pd.read_csv(uv_f + '/cutoff.hit.group', sep='\t')
    transcript_uv_f.columns = ['cut_off', 'transcript_id',
'modified_depth_median',
                               'unmodified_depth_median', 'modified_depth_sum',
'unmodified_depth_sum', 'hit_level']

    uv_z_transcript =
list(transcript_uv_z.loc[(transcript_uv_z['modified_depth_median'] > 100) & (
            transcript_uv_z['hit_level'] > 0), 'transcript_id'])

    uv_f_transcript =
list(transcript_uv_f.loc[(transcript_uv_f['modified_depth_median'] > 100) & (
            transcript_uv_f['hit_level'] > 0), 'transcript_id'])
    transcript_all = list(set(uv_z_transcript) & set(uv_f_transcript))
    print(len(transcript_all))
    gff_path = '/data/TA_QUIZ_RNA_regulation/data/ATH/GFF/Ath_genes.gff'
    data_gff = pd.read_csv(gff_path, sep='\t')
    data_location = pd.read_csv(

 '/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/shape_map/result/transcript_exon_loc
ation.csv',
        sep='\t')
    statistics_sum, statistics_z, statistics_f = main_sum_gini(data_uv_z,
data_uv_f, data_location, data_gff, transcript_all)
    pd.DataFrame(statistics_sum).to_csv(result + '/gini_summary_50_1.csv',
sep='\t', index=False, header=True)
    pd.DataFrame(statistics_z).to_csv(result + '/gini_summary_UV+_50_1.csv',
sep='\t', index=False, header=True)
    pd.DataFrame(statistics_f).to_csv(result + '/gini_summary_UV-_50_1.csv',
sep='\t', index=False, header=True)
```

Merge Structural Change Regions

Running code:

```
#!/bin/bash

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

echo start gini_transcript_merge_windows_new `date`
mkdir -p /data/user_03/RiboShape/Part3_SHAPE/gini
path0=/data/user_03/RiboShape/Part3_SHAPE
python $path0/38.gini.py

echo finish gini_transcript_merge_windows_new `date`

mv $path0/gini_index/summary_result_merge_50_1.csv $path0/gini
mv $path0/gini_index/summary_result_merge_50_1_new.csv $path0/gini
```

Python Script:

```
import numpy as np
import pandas as pd
import re
from numba import jit
from scipy.stats import ks_2samp
import statsmodels.stats.multitest as multi

@jit(nopython=False)
def get_gini(R_, nucleotide_, cut_AC=False, ratio_nan=0.9, ratio_nan_AC=0.8):
    R_nonull = R_[np.isnan(R_)==False]
    ratio = len(R_nonull)/len(R_)
    if ratio <= ratio_nan:
        gini = np.nan
    else:
        if cut_AC:
            R_AC = R_[(nucleotide_==b'A')|(nucleotide_==b'C')]
            R_AC_nonull =  R_AC[np.isnan(R_AC)==False]
            ratio_AC = len(R_AC_nonull) / len(R_AC)
            if (ratio_AC <= ratio_nan_AC)|len(R_AC)<=1|len(R_AC_nonull)<=1:
                gini = np.nan
            else:
                sorted = np.sort(R_AC_nonull)
                height, area = 0, 0
                for i in range(0,len(sorted)):
                    height += sorted[i]
                    area += height - sorted[i] / 2.
                fair_area = height * len(sorted) / 2.
                if fair_area ==0:
                    gini=np.nan
                else:
                    gini =  (fair_area - area) / fair_area
        else:
            sorted = np.sort(R_nonull)
            height, area = 0, 0
            for i in range(0, len(sorted)):
                height += sorted[i]
                area += height - sorted[i] / 2.
```

```
            fair_area = height * len(sorted) / 2.
            if fair_area == 0:
                gini = np.nan
            else:
                gini = (fair_area - area) / fair_area
    return gini

@jit(nopython=False)
def get_p(R_1,R_2,ratio_nan=0.9):
    R_1_nonull = R_1[np.isnan(R_1) == False]
    ratio_1 = len(R_1_nonull) / len(R_1)
    R_2_nonull = R_2[np.isnan(R_2) == False]
    ratio_2 = len(R_2_nonull) / len(R_2)
    if (ratio_1 <= ratio_nan)|(ratio_2 <= ratio_nan):
        p = np.nan
    else:
        s,p=ks_2samp(R_1, R_2)
    return p

@jit(nopython=False)
def
get_window(X,location,window=50,step=1,cut_AC=False,ratio_nan=0.9,ratio_nan_AC=0
.8):

    name = X[0]
    nucleotide = np.array(list(X[1]))
    modified = np.array(X[2].split(',')).astype('int')
    modified_depth = np.array(X[3].split(',')).astype('int')
    unmodified = np.array(X[4].split(',')).astype('int')
    unmodified_depth = np.array(X[5].split(',')).astype('int')
    Mutrs = modified/modified_depth
    Mutru = unmodified/unmodified_depth
    # R = Mutrs - Mutru
    R = np.array(X[6].split(',')).astype('float')
    R[R == -999] = np.nan
    n = len(nucleotide)
    index = np.array(range(0, n))
    location_index = np.array(np.array(location['location'])
[0].split(',')).astype('int')

    nucleotide_ = np.zeros([len(range(0, n-(window), step)),
window],dtype=np.string_)
    R_ = np.zeros([len(range(0, n-(window), step)), window])
    gini = np.zeros([len(range(0, n-(window), step)),])
    index_ = np.zeros([len(range(0, n-(window), step)), window])
    location_index_ = np.zeros([len(range(0, n - (window), step)), window])
    j =0
    for i in range(0, n-(window), step):
        index_[j,:]=index[i:i+window]
        location_index_[j,:] = location_index[i:i+window]
        nucleotide_[j,:] = nucleotide[i:i+window]
        R_[j,:] = R[i:i+window]
        gini[j] =
get_gini(R[i:i+window],nucleotide[i:i+window],cut_AC=cut_AC,ratio_nan=ratio_nan,
ratio_nan_AC=ratio_nan_AC)
        j=j+1

    return index_,location_index_,nucleotide_,R_,gini
```

```
@jit(nopython=False)
def get_window_p(X1,X2, window=50, step=1,ratio_nan=0.9):

    name = X1[0]
    nucleotide = np.array(list(X1[1]))
    R_z = np.array(X1[6].split(',')).astype('float')
    R_z[R_z == -999] = np.nan
    R_f = np.array(X2[6].split(',')).astype('float')
    R_f[R_f == -999] = np.nan
    n = len(nucleotide)

    nucleotide_ = np.zeros([len(range(0, n - (window), step)), window],
dtype=np.string_)
    R_z_ = np.zeros([len(range(0, n - (window), step)), window])
    R_f_ = np.zeros([len(range(0, n - (window), step)), window])
    p = np.zeros([len(range(0, n - (window), step)), ])
    j = 0
    for i in range(0, n - (window), step):
        nucleotide_[j, :] = nucleotide[i:i + window]
        R_z_[j, :] = R_z[i:i + window]
        R_f_[j, :] = R_f[i:i + window]
        p[j] = get_p(R_z[i:i + window], R_f[i:i + window],ratio_nan=ratio_nan,)
        j = j + 1
    p_=p.copy()
    if len(p[~np.isnan(p)])>0:
        a,p_bh,b,c =multi.multipletests(p[~np.isnan(p)],method='fdr_bh')
        p_[~np.isnan(p_)]=p_bh
    return p_,p


@jit(nopython=False)
def calculate_delta_gini(R_1, R_2, gini_1, gini_2, ratio_nan=0.9):
    '''Calculates Standard RMSD on two vectors of numbers of the same length'''
    # Check to see the vectors are of equal length.
    if len(R_1) != len(R_2):
        return np.nan
    else:
        R = R_1 - R_2
        if len(R[np.isnan(R) == False]) / len(R) <= ratio_nan:
            return np.nan
        else:
            delta_gini = gini_1 - gini_2
    return delta_gini

@jit(nopython=False)
def get_gff(X,chr,strand,data_gff_):
    gff=[]
    if len(data_gff_.loc[data_gff_['location'] == 'CDS', 'strat']) == 0:
        for i in range(len(X)):
            gff_='erro'
            gff.append(gff_)
    else:
        CDS_strat = list(data_gff_.loc[data_gff_['location'] == 'CDS', 'strat'])
[0]
        CDS_end = list(data_gff_.loc[data_gff_['location'] == 'CDS', 'end'])[0]
        for i in range(len(X)):
```

```
            if strand == '+':
                if X[i]<CDS_strat:
                    gff_='five_prime_UTR'
                elif X[i]>CDS_end:
                    gff_ = 'three_prime_UTR'
                else:
                    gff_='CDS'
            else:
                if X[i]<CDS_strat:
                    gff_='three_prime_UTR'
                elif X[i]>CDS_end:
                    gff_ = 'five_prime_UTR'
                else:
                    gff_='CDS'
            gff.append(gff_)
    return gff


@jit(nopython=False)
def merge_gini(X1,X2,strat,end):

    name = X1[0]
    nucleotide = np.array(list(X1[1]))
    R_z = np.array(X1[6].split(',')).astype('float')
    R_z[R_z == -999] = np.nan
    R_f = np.array(X2[6].split(',')).astype('float')
    R_f[R_f == -999] = np.nan
    n = len(nucleotide)
    R_z_=R_z[strat:end+1]
    R_f_=R_f[strat:end+1]
    gini_z=get_gini(R_z_,nucleotide_=[],ratio_nan=0)
    gini_f = get_gini(R_f_, nucleotide_=[],ratio_nan=0)
    delta=gini_z-gini_f
    p=get_p(R_z_,R_f_,ratio_nan=0)
    R_f_mean = R_f_[R_f_>=0].mean()
    R_z_mean = R_z_[R_z_ >= 0].mean()
    return delta,p,R_f_mean,R_z_mean,gini_z,gini_f



@jit(nopython=False)
def get_statistics(data_z,data_f,data_location_,data_gff_,strand):
    data_z = np.array(data_z).reshape([7,])
    data_f = np.array(data_f).reshape([7,])
    chr=list(data_location_['chr'])[0]
    index_z,location_index_z,nucleotide_z, R_z, gini_z =
get_window(data_z,data_location_)
    index_f,location_index_f,nucleotide_f, R_f, gini_f =
get_window(data_f,data_location_)
    # print('ok')
    if len(R_z)!=len(R_f):
        print("error")
    else:
        location_exon = get_gff(location_index_z[:,0],chr,strand,data_gff_)
        # print('ok')
        location_exon = np.array(location_exon)
        p_bh, p = get_window_p(data_z, data_f)
        delta_gini = np.zeros([len(R_z), ])
```

```
        index_list = []
        nucleotide_list = []
        location_index_list = []
        for i in range(len(R_z)):
            index_list.append(
','.join(list(index_z[i,:].astype('int').astype('str'))))
            nucleotide_list.append(','.join(list(nucleotide_z[i,
:].astype('str'))))
            location_index_list.append(
','.join(list(location_index_z[i,:].astype('int').astype('str'))))
            delta_gini[i] = calculate_delta_gini(R_z[i, :], R_f[i, :],
gini_z[i], gini_f[i])
        index_list = np.array(index_list)
        location_index_list = np.array(location_index_list)
        nucleotide_list = np.array(nucleotide_list)
    return delta_gini,p_bh,p, gini_z,
gini_f,index_list,location_index_list,nucleotide_list,location_exon


# @jit(nopython=False,error_model="numpy")
def
merge_data(delta_gini,index_list,data_z,data_f,location_list,chr,strand,data_gff
_,windows=50):
    data_z = np.array(data_z).reshape([7,])
    data_f = np.array(data_f).reshape([7,])
    nucleotide = np.array(list(data_z[1]))
    index = np.array([x.split(',')[0] for x in index_list]).astype('int')
    delta_gini_ =delta_gini.copy()
    delta_gini_[np.isnan(delta_gini_)]=0
    i=0
    windows_num=[]
    n=index[abs(delta_gini_)>0.1][0]+windows-1
    for j in range(len(index)-1):
        if abs(delta_gini_[j])<0.1:
            windows_num.append(np.nan)
        else:
            if index[j]>n+1:
                i=i+1
            n=index[j]+windows-1
            windows_num.append(i)
    if abs(delta_gini_[len(index)-1])<0.1:
        windows_num.append(np.nan)
    else:
        if index[j] > n + 1:
            i = i + 1
        windows_num.append(i)

    data_all= pd.DataFrame(np.zeros([len(set(windows_num) - set([np.nan])),9]))
    for i in list(set(windows_num) - set([np.nan])):
        windows_num_=np.array(windows_num)
        strat = index[windows_num_==i].astype('int')[0]
        end = index[windows_num_==i].astype('int')[-1]+windows-1
        strat_chr=location_list[strat]
        end_chr = location_list[end]
        delta,p=merge_gini(data_z,data_f,strat,end)
        location = get_gff(np.array([strat_chr,end_chr]),chr,strand,data_gff_)
        nucleotide_ =nucleotide[strat:end+1]
        nucleotide_str = ','.join(list(nucleotide_.astype('str')))
```

```
        data_all.iloc[i,:]=
[strat,end,strat_chr,end_chr,location[0],location[1],delta,p,nucleotide_str]
    data_all.columns=
['start','end','start_chr','end_chr','location_start','location_end','delta','p'
,'nucleotide']
    return data_all


def
merge_data_2(delta_gini,index_list,data_z,data_f,location_list,chr,strand,data_g
ff_,windows=50):
    data_z = np.array(data_z).reshape([7,])
    data_f = np.array(data_f).reshape([7,])
    nucleotide = np.array(list(data_z[1]))
    index = np.array([x.split(',')[0] for x in index_list]).astype('int')
    delta_gini_ =delta_gini.copy()
    delta_gini_[np.isnan(delta_gini_)]=0
    i=0
    windows_num=[]
    if abs(delta_gini_[0]) < 0.1:
        windows_num.append(np.nan)
    else:
        i = i + 1
        windows_num.append(i)

    for j in range(1,len(index)):
        if abs(delta_gini_[j])<0.1:
            windows_num.append(np.nan)
        else:
            if abs(delta_gini_[j-1])>0.1:
                windows_num.append(i)
            else:
                i=i+1
                windows_num.append(i)

    data_all= pd.DataFrame(np.zeros([len(set(windows_num) - set([np.nan])),13]))
    for i in list(set(windows_num) - set([np.nan])):
        windows_num_=np.array(windows_num)
        strat = index[windows_num_==i].astype('int')[0]
        end = index[windows_num_==i].astype('int')[-1]+windows-1
        delta, p, R_f_mean,
R_z_mean,gini_z,gini_f=merge_gini(data_z,data_f,strat,end)
        if abs(delta)<=0.1:
            num = len(index[windows_num_==i].astype('int'))
            for n in range(1,num):
                end_=end-n
                delta, p, R_f_mean, R_z_mean,gini_z,gini_f = merge_gini(data_z,
data_f, strat, end_)
                if abs(delta)>0.1:
                    end = end_
                    break
        strat_chr=location_list[strat]
        end_chr = location_list[end]
        location = get_gff(np.array([strat_chr,end_chr]),chr,strand,data_gff_)
        nucleotide_ =nucleotide[strat:end+1]
        nucleotide_str = ','.join(list(nucleotide_.astype('str')))
```

```
        data_all.iloc[i-1,:]=
[strat,end,strat_chr,end_chr,location[0],location[1],delta,p,R_f_mean,
R_z_mean,gini_z,gini_f,nucleotide_str]
    data_all.columns=
['start','end','start_chr','end_chr','location_start','location_end','delta','p'
,'R_f_mean','R_z_mean','gini_z','gini_f','nucleotide']
    return data_all



def
main_sum(data_uv_z,data_uv_f,transcript_id_list,data_location,data_gff,save_path
):
    i=0
    data_all = pd.DataFrame(columns=
['transcript_id','start','end','start_chr','end_chr','location_start','location_
end','delta','p','R_f_mean','R_z_mean','gini_z','gini_f','nucleotide'])
    for transcript in transcript_id_list:
        data_z = data_uv_z.loc[data_uv_z['transcript_id']==transcript,:]
        data_f = data_uv_f.loc[data_uv_f['transcript_id'] == transcript,:]
        data_location_ = data_location.loc[data_location['transcript_id'] ==
transcript,:]
        data_gff_ = data_gff.loc[data_gff['transcript_id']== transcript,:]
        location_list=np.array(list(data_location_['location'])
[0].split(',')).astype('int')
        chr = list(data_location_['chr'])[0]
        strand = list(data_location_['strand'])[0]
        delta_gini, p_bh,p,gini_z, gini_f, index_list, location_index_list,
nucleotide_list,location_exon =
get_statistics(data_z,data_f,data_location_,data_gff_,strand)
        # print(delta_gini)

        data_new =
pd.DataFrame(np.vstack([location_exon,index_list,location_index_list,nucleotide_
list,gini_z, gini_f,delta_gini,p_bh,p])).T
        data_new.columns =
['location_exon','index','location_index','nucleotide','gini_z',
'gini_f','delta_gini','p_bh','p']
        data_new['transcript']=transcript
        #
pd.DataFrame(data_new).to_csv(save_path+'/'+transcript+'_gini.csv',sep='\t',head
er=True,index=False)
        data_all_=merge_data_2(delta_gini, index_list, data_z, data_f,
location_list,chr,strand,data_gff_)
        data_all_['transcript_id']=transcript
        data_all = pd.concat([data_all,data_all_])
        i = i+1
        print(i)
    return data_all

####################主函数部分####################
if __name__ == '__main__':
    col_uv_f =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_nouv/'
    col_uv_z =
'/data/user_03/RiboShape/Part3_SHAPE/final.modified_umodified/col_uv/'

    result = '/data/user_03/RiboShape/Part3_SHAPE/gini_index'
```

```
    uv_z = col_uv_z
    uv_f = col_uv_f

    data_uv_z = pd.read_csv(uv_z + '/final.modified_unmodified_new', sep='\t')
    data_uv_z.columns = ['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                         'Untreated_mutations', 'Untreated_effective_depth',
'R1']
    data_uv_z = data_uv_z[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                         'Untreated_mutations', 'Untreated_effective_depth',
'R1']]
    data_uv_z = data_uv_z.drop_duplicates()

    data_uv_f = pd.read_csv(uv_f + '/final.modified_unmodified_new', sep='\t',
header=None)
    data_uv_f.columns = ['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                         'Untreated_mutations', 'Untreated_effective_depth',
'R1']
    data_uv_f = data_uv_f[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                         'Untreated_mutations', 'Untreated_effective_depth',
'R1']]
    data_uv_f = data_uv_f.drop_duplicates()
    statistics_sum=pd.read_csv(result + '/gini_summary_50_1.csv', sep='\t')
    # statistics_sum_new = pd.read_csv(result +
'/summary_result_merge_50_1_0.csv',sep='\t')

    transcript_all =
list(statistics_sum.loc[statistics_sum['num_0.1']>0,'transcript_id'])
    # transcript_all =transcript_all[:1000]
    # transcript_all=['AT5G26000.1','AT5G26000.2']
    # transcript_all = ['AT5G45260.2']
    print(len(transcript_all))
    gff_path = '/data/TA_QUIZ_RNA_regulation/data/ATH/GFF/Ath_genes.gff'
    data_gff = pd.read_csv(gff_path, sep='\t')
    # data_gff.columns = ['exosome','name','location','strat','end','.','+/-
','num','id']
    data_location =
pd.read_csv('/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/shape_map/result/transcri
pt_exon_location.csv', sep='\t')
    data_all =
main_sum(data_uv_z,data_uv_f,transcript_all,data_location,data_gff,result+'/tran
script_gini_merge')

 data_all.to_csv(result+'/summary_result_merge_50_1.csv',sep='\t',index=False)
    hit_level_coil_uv_f = pd.read_csv(uv_f+'/cutoff.hit.group',sep='\t')
    #hit_level_coil_uv_f = hit_level_coil_uv_f.reset_index()
    hit_level_coil_uv_f.columns =
['group','transcript_id','modified.median','unmodified.median','modified.sum','u
nmodified.sum','hit_f']
    hit_level_coil_uv_z = pd.read_csv(uv_z+'/cutoff.hit.group',sep='\t')
    #hit_level_coil_uv_z = hit_level_coil_uv_z.reset_index()
    hit_level_coil_uv_z.columns =
['group','transcript_id','modified.median','unmodified.median','modified.sum','u
nmodified.sum','hit_z']
```

```
    shape_data_coil
=pd.merge(pd.merge(data_all,hit_level_coil_uv_f[['transcript_id','hit_f']],on='t
ranscript_id',how='left'),hit_level_coil_uv_z[['transcript_id','hit_z']],on='tra
nscript_id',how='left')
    shape_data_coil_2 =shape_data_coil.loc[(shape_data_coil['hit_f']>2)&
(shape_data_coil['hit_z']>2),:]

 shape_data_coil.to_csv(result+'/summary_result_merge_50_1_new.csv',sep='\t',ind
ex=False)
```

Data Integration

Transcript Abundance & TE

```
import pandas as pd
import re
#import seaborn as sns
import matplotlib.pyplot as plt
plt.switch_backend('agg')
from numpy import median
from numba import jit

RF_data=pd.read_csv('/data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-
1.TE_new.csv',sep='\t')
RF_data=RF_data.reset_index()
RF_data=RF_data.rename(columns={'index':'gene'})

RS_data=pd.read_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expressio
n/7.DEseq/wt/wt_rawdata.csv',sep=',')
RS_data=RS_data.rename(columns={'Row.names':'gene'})

data=pd.merge(RS_data[['gene','pvalue','padj','log2FoldChange']],RF_data[['gene'
,'pvalue_final','pvalue.adjust','log2FC_TE_final']],on='gene',how='right')
data.columns=['gene','pvalue(RNA-seq)','padj(RNA-seq)','log2FoldChange(RNA-
seq)','pvalue(TE)','padj(TE)','log2FoldChange(TE)']


data['group'] = 'darkgray'
result_1=data.loc[(data['pvalue(TE)']<0.05)&(data['log2FoldChange(TE)']>0)&
(data['padj(RNA-seq)']>0.05),:]
result_2=data.loc[(data['pvalue(TE)']<0.05)&(data['log2FoldChange(TE)']<0)&
(data['padj(RNA-seq)']>0.05),:]

data_= data[["log2FoldChange(RNA-seq)","log2FoldChange(TE)"]]
data_corr = data_.corr().iloc[0,1]

xmin=-3
xmax=6
ymin=-8
ymax=8

fig = plt.figure(figsize=plt.figaspect(5/6)) #确定fig比例（h/w）
ax = fig.add_subplot()
ax.set(xlim=(xmin, xmax), ylim=(ymin, ymax), title='')
```

```
ax.scatter(data['log2FoldChange(RNA-seq)'], data['log2FoldChange(TE)'], s=15,
c=data['group'])

ax.scatter(result_1['log2FoldChange(RNA-seq)'], result_1['log2FoldChange(TE)'],
s=20, marker='.',c='#cc0000',label = str(len(result_1))+' TE up mRNAs')
ax.scatter(result_2['log2FoldChange(RNA-seq)'], result_2['log2FoldChange(TE)'],
s=20, marker='.',c='steelblue',label = str(len(result_2))+' TE down mRNAs')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_linewidth(2)
ax.spines['left'].set_linewidth(2)

plt.tick_params(labelsize=15)
ax.set_xticks(range(xmin,xmax,1))
ax.set_yticks(range(ymin,ymax,2))
# font2 = {'family': 'Times New Roman','weight': 'normal','size': 15}
font2 = {'weight': 'normal','size': 18}
plt.xlabel('log2FoldChange(RNA-seq)',font2)
plt.ylabel('log2FoldChange(TE)',font2)
plt.legend(fontsize=10)
font3 = {'weight': 'normal','size': 16}
plt.text(3,4, 'r = '+str(round(data_corr,2)),font3)
plt.tight_layout()
# plt.show()
plt.savefig('/data/user_03/RiboShape/Final_Intergration/result/1/logFC_TE_corr.p
ng')
plt.close()
```

Structure Change & TE

Hypothesis Testing

```
import pandas as pd # Data analysis
import numpy as np # Scientific computing
import matplotlib.pyplot as plt # Plotting
import matplotlib.colors as colors # Coloring
from scipy.stats import chi2_contingency
from scipy.stats import fisher_exact

exp_data_wt =
pd.read_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEse
q/wt/wt_rawdata.csv',sep=',')
exp_data_wt =exp_data_wt.rename(columns=
{'Row.names':'gene','log2FoldChange':'log2FoldChange(EXP)','pvalue':'pvalue(EXP)
','padj':'FDR(EXP)'})
exp_data_wt =
exp_data_wt[['gene','log2FoldChange(EXP)','pvalue(EXP)','FDR(EXP)']]
ribo_wt = pd.read_csv('/data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-
1.TE_new.csv',sep='\t')
ribo_wt = ribo_wt.reset_index()
ribo_wt =ribo_wt.rename(columns=
{'index':'gene','log2FC_TE_final':'log2FoldChange(TE)','pvalue_final':'pvalue(TE
)','pvalue.adjust':'FDR(TE)'})
ribo_wt = ribo_wt[['gene','log2FoldChange(TE)','pvalue(TE)','FDR(TE)']]
result = pd.merge(ribo_wt,exp_data_wt,on='gene',how='left')
```

```
gene_all_=set(result.loc[((result['log2FoldChange(TE)']>0.5)|
(result['log2FoldChange(TE)']<-0.5))&(result['pvalue(TE)']<0.05)&
(result['FDR(EXP)']>0.05),'gene'])
gene_TE_up_2=set(result.loc[(result['log2FoldChange(TE)']>0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])
gene_TE_no_up_2=gene_all_-gene_TE_up_2
gene_TE_down_2=set(result.loc[(result['log2FoldChange(TE)']<-0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])
gene_TE_no_down_2=gene_all_-gene_TE_down_2

#Have structure changed region(|delta gini index|>0.1)
shape_data_wt = pd.read_csv('/data/TA_QUIZ_RNA_regulation/result/PartIII.SHAPE-
seq_analysis/merge/merge_data_WT.csv',sep='\t')
shape_data_wt = shape_data_wt.loc[(shape_data_wt['hit_z']>1)|
(shape_data_wt['hit_f']>1),:]
shape_data_wt['up_down']= shape_data_wt['delta'].map(lambda x: 'up' if x>0 else
'down')
shape_data_wt['gene']=shape_data_wt['transcript_id'].map(lambda x:x.split('.')
[0])
shape_up_1=set(shape_data_wt.loc[(shape_data_wt['up_down']=='up'),'gene'])&gene_
all_
shape_no_up_1=gene_all_-shape_up_1
shape_down_1=set(shape_data_wt.loc[(shape_data_wt['up_down']=='down'),'gene'])&g
ene_all_
shape_no_down_1=gene_all_-shape_down_1

########################
dtest1=np.array([[len(gene_TE_down_2&shape_up_1),len(gene_TE_down_2&shape_no_up_
1)],

[len(gene_TE_no_down_2&shape_up_1),len(gene_TE_no_down_2&shape_no_up_1)]])
# k,p,f,expctd =chi2_contingency(dtest)
o,p=fisher_exact(dtest1,alternative='greater')
print('shape up,TE down')
print(p)

dtest3=np.array([[len(gene_TE_up_2&shape_down_1),len(gene_TE_up_2&shape_no_down_
1)],

[len(gene_TE_no_up_2&shape_down_1),len(gene_TE_no_up_2&shape_no_down_1)]])
o,p=fisher_exact(dtest3,alternative='greater')
print('shape down,TE up')
print(p)
```

Draw Enrichment Degree

```
import pandas as pd # Data analysis
import numpy as np # Scientific computing
import matplotlib.pyplot as plt # Plotting
import matplotlib.colors as colors # Coloring


exp_data_wt =
pd.read_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEse
q/wt/wt_rawdata.csv',sep=',')
```

```
exp_data_wt =exp_data_wt.rename(columns=
{'Row.names':'gene','log2FoldChange':'log2FoldChange(EXP)','pvalue':'pvalue(EXP)
','padj':'FDR(EXP)'})
exp_data_wt =
exp_data_wt[['gene','log2FoldChange(EXP)','pvalue(EXP)','FDR(EXP)']]
exp_wt_list=list(exp_data_wt.loc[exp_data_wt['FDR(EXP)']>0.05,'gene'])
ribo_wt = pd.read_csv('//data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-
1.TE_new.csv',sep='\t')
ribo_wt = ribo_wt.reset_index()
ribo_wt =ribo_wt.rename(columns=
{'index':'gene','log2FC_TE_final':'log2FoldChange(TE)','pvalue_final':'pvalue(TE
)','pvalue.adjust':'FDR(TE)'})
ribo_wt = ribo_wt[['gene','log2FoldChange(TE)','pvalue(TE)','FDR(TE)']]

shape_data_wt = pd.read_csv('/data/TA_QUIZ_RNA_regulation/result/PartIII.SHAPE-
seq_analysis/merge/merge_data_WT.csv',sep='\t')
shape_data_wt = shape_data_wt.loc[(shape_data_wt['hit_z']>1)|
(shape_data_wt['hit_f']>1),:]
shape_data_wt['up_down']= shape_data_wt['delta'].map(lambda x: 'up' if x>0 else
'down')
shape_data_wt['gene']=shape_data_wt['transcript_id'].map(lambda x:x.split('.')
[0])
shape_up_list=list(shape_data_wt.loc[(shape_data_wt['up_down']=='up'),'gene'])
shape_down_list=list(shape_data_wt.loc[(shape_data_wt['up_down']=='down'),'gene'
])


result = pd.merge(ribo_wt,exp_data_wt,on='gene',how='left')
result=result.loc[result['FDR(EXP)']>0.05,:]
result['shape_up']=result['gene'].map(lambda x:1 if x in shape_up_list else 0)
result['shape_down']=result['gene'].map(lambda x:1 if x in shape_down_list else
0)
result['x'] = result['log2FoldChange(TE)']
result['y'] = -np.log10(result['pvalue(TE)'])

x_threshold=0.5
y_threshold=-np.log10(0.05)

result_2=result.loc[((result['x']<-0.5)|(result['x'] >0.5))&(result['y']
>y_threshold)&(result['shape_up']==1),:]
result_3=result.loc[((result['x']<-0.5)|(result['x'] >0.5))&(result['y']
>y_threshold)&(result['shape_down']==1),:]

result['group'] = 'dimgrey'
# result.loc[(result.x > x_threshold)&(result.y > y_threshold),'group'] =
'tab:red' #x=-+x_threshold直接截断
# result.loc[(result.x < -x_threshold)&(result.y > y_threshold),'group'] =
'tab:blue' #x=-+x_threshold直接截断
# result.loc[result.y < y_threshold,'group'] = 'dimgrey' #阈值以下点为灰色

xmin=-8
xmax=8
ymin=-1
ymax=8

fig = plt.figure(figsize=plt.figaspect(5/7)) #确定fig比例（h/w）
ax = fig.add_subplot()
ax.set(xlim=(xmin, xmax), ylim=(ymin, ymax), title='')
```

```
ax.scatter(result['x'], result['y'], s=2, c=result['group'])

# ax.scatter(result_2['x'], result_2['y'], s=10,
marker='o',c='',edgecolors='tab:purple',label = 'More Structure in UV+')
ax.scatter(result_2['x'], result_2['y'], s=8,
marker='o',c='',edgecolors='fuchsia',label = 'More Structure in UV+')
ax.scatter(result_3['x'], result_3['y'], s=5,
marker='o',c='',edgecolors='tab:orange',label = 'Less Structure in UV+')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

ax.vlines(-x_threshold, ymin, ymax, color='dimgrey',linestyle='dashed',
linewidth=1) #画竖直线
ax.vlines(x_threshold, ymin, ymax, color='dimgrey',linestyle='dashed',
linewidth=1) #画竖直线
ax.hlines(y_threshold, xmin, xmax, color='dimgrey',linestyle='dashed',
linewidth=1) #画竖水平线

plt.tick_params(labelsize=13)
ax.set_xticks(range(xmin,xmax,2))
ax.set_yticks(range(ymin,ymax,2))
# font2 = {'family': 'Times New Roman','weight': 'normal','size': 15}
font2 = {'weight': 'normal','size': 18}
plt.xlabel('Log2FoldChange(TE)',font2)
plt.ylabel('Log10Pvalue(TE)',font2)
plt.legend(fontsize=13)
plt.tight_layout()
plt.savefig('/data/user_03/RiboShape/Final_Intergration/result/2/volcano.png')
plt.close()
```

Motif Analysis

Extract All 3'UTR or 5'UTR FASTQ Files

```
import numpy as np
import pandas as pd

col_uv_f =
'/data/TA_QUIZ_RNA_regulation/data/riboshape_liulab_batch4/final.modified_umodif
ied/col_nouv/'

gff_path = '/data/TA_QUIZ_RNA_regulation/data/ATH/GFF/Ath_genes.gff'
data_gff = pd.read_csv(gff_path, sep='\t')
data_location = pd.read_csv(
'/data/TA_QUIZ_RNA_regulation/data/ATH/GTF/shape_map/result/transcript_exon_loca
tion.csv',sep='\t')
data_gff=pd.merge(data_gff,data_location[['transcript_id','strand']],on='transcr
ipt_id',how='left')
gene=pd.read_csv('/data/TA_QUIZ_RNA_regulation/data/gene_list/wt/ribo_wt_gene_li
st.txt',sep='\t',header=None)

gene_list=list(gene[0])

data_1 = pd.read_csv(col_uv_f + '/final.modified_unmodified_new', sep='\t')
data_1.columns = ['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
```

```
                     'Untreated_mutations', 'Untreated_effective_depth', 'R1']
data_1 = data_1[['transcript_id', 'Nucleotide', 'Modified_mutations',
'Modified_effective_depth',
                      'Untreated_mutations', 'Untreated_effective_depth',
'R1']]
data_1 = data_1.drop_duplicates()
data_1['gene']=data_1['transcript_id'].map(lambda x:x.split('.')[0])


data=data_1
data_nuc=pd.DataFrame(columns={'gene','transcript_id','5UTR','CDS','3UTR'})
j=0

for i in gene_list:
    data_ = data.loc[data['gene'] == i, :]

    transcript_id = list(data_['transcript_id'])[0]

    data_location_ = data_location.loc[data_location['transcript_id'] ==
transcript_id, :]
    data_gff_ = data_gff.loc[data_gff['transcript_id'] == transcript_id, :]
    if len(data_gff_) == 0:
        continue
    else:
        if list(data_gff_['strand'])[0] == '+':
            five_UTR_CDS = list(data_gff_.loc[data_gff_['location'] == 'CDS',
'strat'])[0]
            three_UTR_CDS = list(data_gff_.loc[data_gff_['location'] == 'CDS',
'end'])[0]
        else:
            five_UTR_CDS = list(data_gff_.loc[data_gff_['location'] == 'CDS',
'end'])[0]
            three_UTR_CDS = list(data_gff_.loc[data_gff_['location'] == 'CDS',
'strat'])[0]
        Nuc = list(data_['Nucleotide'])[0]
        location = np.array(list(data_location_['location'])
[0].split(',')).astype('int')
        five_UTR_CDS_ = np.where(location == five_UTR_CDS)[0][0]
        three_UTR_CDS_ = np.where(location == three_UTR_CDS)[0][0]

        data_5UTR_ = Nuc[:five_UTR_CDS_]
        data_CDS_ = Nuc[five_UTR_CDS_:three_UTR_CDS_ + 1]
        data_3UTR_ = Nuc[three_UTR_CDS_ + 1:]
    data_nuc.loc[j, 'gene'] = i
    data_nuc.loc[j, 'transcript_id'] = transcript_id
    data_nuc.loc[j, '5UTR'] = data_5UTR_
    data_nuc.loc[j, '5UTR_len'] = len(data_5UTR_)
    data_nuc.loc[j, 'CDS'] = data_CDS_
    data_nuc.loc[j, 'CDS_len'] = len(data_CDS_)
    data_nuc.loc[j, '3UTR'] = data_3UTR_
    data_nuc.loc[j, '3UTR_len'] = len(data_3UTR_)
    j = j + 1
    print(j)

data_nuc=data_nuc[['gene','transcript_id','5UTR_len','CDS_len','3UTR_len','5UTR'
,'CDS','3UTR']]
print('ALL',len(data_nuc))
```

```
n=15
data_nuc_=data_nuc.loc[data_nuc['5UTR_len']>n]
data_nuc_.index=range(len(data_nuc_))
print('5UTR',len(data_nuc_))

f = open('/data/user_03/RiboShape/Final_Intergration/result/3/merge_5UTR.fasta',
'w')
for i in range(len(data_nuc_)):
    transcript = data_nuc_.loc[i, 'transcript_id']
    nucleotide= data_nuc_.loc[i, '5UTR']
    f.write('>' + transcript+'\n')
    f.write(nucleotide + '\n')

f.close()

data_nuc_=data_nuc.loc[data_nuc['3UTR_len']>n]
data_nuc_.index=range(len(data_nuc_))
print('3UTR',len(data_nuc_))
f = open('/data/user_03/RiboShape/Final_Intergration/result/3/merge_3UTR.fasta',
'w')
for i in range(len(data_nuc_)):
    transcript = data_nuc_.loc[i, 'transcript_id']
    nucleotide= data_nuc_.loc[i, '3UTR']
    f.write('>' + transcript+'\n')
    f.write(nucleotide + '\n')

f.close()
```

Extract 3 'UTR and 5' UTR Sequence of the TE Changing Region

Generate Gene List of Up-regulated TE【new】

Running code:

```
#!/bin/bash
#SBATCH -J PartIII.SHAPE-seq_analysis
#SBATCH -p CN_BIOT
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --output=genelist_up.txt
#SBATCH --error=genelist_up.err

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

python /data/user_03/RiboShape/Final_Intergration/genelist_up.py
```

Python script:

```
import pandas as pd # Data analysis
import numpy as np # Scientific computing
import matplotlib.pyplot as plt # Plotting
import matplotlib.colors as colors # Coloring
from scipy.stats import chi2_contingency
from scipy.stats import fisher_exact
```

```
exp_data_wt =
pd.read_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEseq/wt/wt_rawdata.csv',sep=',')
exp_data_wt =exp_data_wt.rename(columns=
{'Row.names':'gene','log2FoldChange':'log2FoldChange(EXP)','pvalue':'pvalue(EXP)','padj':'FDR(EXP)'})
exp_data_wt =
exp_data_wt[['gene','log2FoldChange(EXP)','pvalue(EXP)','FDR(EXP)']]
ribo_wt = pd.read_csv('/data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-1.TE_new.csv',sep='\t')
ribo_wt = ribo_wt.reset_index()
ribo_wt =ribo_wt.rename(columns=
{'index':'gene','log2FC_TE_final':'log2FoldChange(TE)','pvalue_final':'pvalue(TE)','pvalue.adjust':'FDR(TE)'})
ribo_wt = ribo_wt[['gene','log2FoldChange(TE)','pvalue(TE)','FDR(TE)']]
result = pd.merge(ribo_wt,exp_data_wt,on='gene',how='left')

gene_all_=set(result.loc[((result['log2FoldChange(TE)']>0.5)|
(result['log2FoldChange(TE)']<-0.5))&(result['pvalue(TE)']<0.05)&
(result['FDR(EXP)']>0.05),'gene'])
gene_TE_up_2=set(result.loc[(result['log2FoldChange(TE)']>0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])
gene_TE_down_2=set(result.loc[(result['log2FoldChange(TE)']<-0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])

for v in gene_TE_up_2:
    print(v)
```

Generate Gene List of Down-regulated TE【new】

Running code:

```
#!/bin/bash
#SBATCH -J PartIII.SHAPE-seq_analysis
#SBATCH -p CN_BIOT
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --output=genelist_down.txt
#SBATCH --error=genelist_down.err

export PATH=/data/zhaoyizi/software/anaconda3/envs/Riboshape/bin:$PATH

python /data/user_03/RiboShape/Final_Intergration/genelist_down.py
```

Python script:

```
import pandas as pd # Data analysis
import numpy as np # Scientific computing
import matplotlib.pyplot as plt # Plotting
import matplotlib.colors as colors # Coloring
from scipy.stats import chi2_contingency
from scipy.stats import fisher_exact

exp_data_wt =
pd.read_csv('/data/user_03/RiboShape/Part1/RNAseq/differential_expression/7.DEseq/wt/wt_rawdata.csv',sep=',')
```

```
exp_data_wt =exp_data_wt.rename(columns=
{'Row.names':'gene','log2FoldChange':'log2FoldChange(EXP)','pvalue':'pvalue(EXP)
','padj':'FDR(EXP)'})
exp_data_wt =
exp_data_wt[['gene','log2FoldChange(EXP)','pvalue(EXP)','FDR(EXP)']]
ribo_wt = pd.read_csv('/data/user_03/RiboShape/Part2_Riboseq/7.TE/wt.0-vs-
1.TE_new.csv',sep='\t')
ribo_wt = ribo_wt.reset_index()
ribo_wt =ribo_wt.rename(columns=
{'index':'gene','log2FC_TE_final':'log2FoldChange(TE)','pvalue_final':'pvalue(TE
)','pvalue.adjust':'FDR(TE)'})
ribo_wt = ribo_wt[['gene','log2FoldChange(TE)','pvalue(TE)','FDR(TE)']]
result = pd.merge(ribo_wt,exp_data_wt,on='gene',how='left')

gene_all_=set(result.loc[((result['log2FoldChange(TE)']>0.5)|
(result['log2FoldChange(TE)']<-0.5))&(result['pvalue(TE)']<0.05)&
(result['FDR(EXP)']>0.05),'gene'])
gene_TE_up_2=set(result.loc[(result['log2FoldChange(TE)']>0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])
gene_TE_down_2=set(result.loc[(result['log2FoldChange(TE)']<-0.5)&
(result['pvalue(TE)']<0.05)&(result['FDR(EXP)']>0.05),'gene'])

for w in gene_TE_down_2:
    print (w)
```

Motif Analysis with MEME

MEME