```
##########################################################################
```
##SCRIPT 1
This script opens the fast raw data datasets located in a known directory and extracts the DNA sequence downstream 23As or upstream 23Ts. This script then aligns the reads using bowtie and further prepare the data using Samtools suite. Please cite the manuscript attached to this code if you use it entirely or part of it. Code written by Marco Mangone
```
##########################################################################

#!/usr/bin/perl

use File::stat;
use URI::Escape;
use Bio::SeqIO;
use strict;

my $file = "outfile";
open OUTFILE, ">$file.fasta" or die $!;
my $directory = '/home/mangone/fastq';
opendir( DIR, $directory ) or die $!;
while ( my $file = readdir(DIR) ) {
    if ( $file =~ m/cumulative_plus/ ) {
        print "processing -> $file\n";
        open DATABASE, $file or die "cannot find the file";
        my $i = 1;
        my $dataset;
        my $trueStart;
        my @seq;
        my $length;
        while (my $line = <DATABASE> ) {
            chop $line;
            if (($line =~ m/\@/ ) and ( $line =~ m/length/ ) ) {
                $line =~ m/\@(.+?) /;
                $dataset = $1;
                #print ">$dataset|$i\n";
                #$i++;
            }
            if ($line =~ m/^[ATCG]+$/ ) {
                my @seq = split( /AAAAAAAAAAAAAAAAAAAAAAA/, $line );
                #my @seq = split( /TTTTTTTTTTTTTTTTTTTTTTT/, $line );
                my $length = length( $seq[0] );
                if ( $length > 20 ) {
                    print OUTFILE ">$dataset|$i\n";
                    print OUTFILE "$seq[0]\n";
                    $i++;
                }
                else {
                }
            }
        }
    }
}

close OUTFILE;

print "convert fasta file in a fake fastq file\n";
system("perl fasta_to_fastq.pl outfile.fasta > outfile.fastq");
print "running bowtie2 for $file\n";
system(
```

```
"bowtie2  -x /home/mangone/Genomes/elegans/WS250/WS250  -p 16 outfile.fastq
-S outfile.sam"
);
system("sleep(5)");
print "running samtools for $file\n";
system(
"samtools view -bt /home/mangone/Genomes/elegans/WS250/WS250.fa.fai
outfile2.sam > outfile.bam"
);
system("sleep(5)");
print "sorting $file\n";
system("samtools sort -@ 16 outfile.bam outfile.bam.sorted");
system("sleep(5)");
system("samtools index outfile.bam.sorted.bam");
print "$file completed!\n";



##############################################################################
```

##SCRIPT 2

This script opens the samtools .sam files, extracts from the CIGAR the reads that map 100% to the reference genome WS250, prepares the bedgraph file, and the clusters used to map 3'UTRs. Please cite the manuscript attached to this code if you use it entirely or part of it. Code written by Marco Mangone

```
##############################################################################

#!/usr/bin/perl

use File::stat;
use URI::Escape;
use Bio::SeqIO;
use Bio::DB::SeqFeature::Store;
use DBI;
use strict;
use Bio::Perl;

my $db2 = Bio::DB::SeqFeature::Store->new(
    -adaptor => 'DBI::mysql',
    -dsn     => 'dbi:mysql:WS250',
    -user    => '',
    -password => '',
);

my $mySQL = getMySQLHandle();

##name of the aligned bam file
my $name = "outfile";

open ENDS, ">end.fa" or die "cannot find the file";

##############################################################################
#########REVERSE STRAND########################################################
##############################################################################

##extract reverse strands
print "extracting sam reads for NEGATIVE strand from $name\n";
system(
"samtools view -f 16 $name.bam.sorted.bam | grep XM:i:0 > $name.negative.sam "
);
print "done!\n";

##map PAS end in the positive strand
##############################################################################
print "map PAS end in the negative strand from $name.negative.sam\n";
open NEGATIVE_SAM, "$name.negative.sam"       or die "cannot find the file";
open NEGATIVE_BED, ">$name.negative.bedgraph" or die "cannot find the file";
while ( my $line = <NEGATIVE_SAM> ) {
```

```
    #print "<<$line>>\n";
    chop $line;
    my @string    = split( "\t", $line );
    my $chr       = $string[2];
    my $map_start = $string[3];
    my $seq       = $string[9];
    my $cigar     = $string[5];
    $cigar =~ s/M//;
    my $seq_length = length($seq);

    #print "<<$seq_length $cigar>>\n";
    #exit;
    if ( $cigar ne $seq_length ) {

        #print "error!!!\n";
    }
    else {
        #make final cluster of 5nt long
        my $start = $map_start;
        my $end   = $start + 1;

        print NEGATIVE_BED "$chr\t$start\t$end\n";
    }
}
close NEGATIVE_SAM;
print "sorting the file\n";
system("bedtools sort -i $name.negative.bedgraph > tmp.bed");
print "done!!!\n";
print "merging the file\n";
system("bedtools merge -c 1 -o count -i tmp.bed > $name.negative.cluster");
print "done!!!\n";

#check numbers of As in the genome
print "check numbers of As in the genome\n";
open DATABASE, "$name.negative.cluster" or die "cannot find the file";
open CLUSTER_MINUS, ">$name.negative.cluster.final"
  or die "cannot find the file";
while ( my $line = <DATABASE> ) {
    chop $line;
    my @string           = split( /\t/, $line );
    my $chr              = $string[0];
    my $start            = $string[1];
    my $stop             = $string[2];
    my $cluster_coverage = $string[3];
##use only cluster with at least 5 reads
    if ( $cluster_coverage >= 5 ) {
        my $seq = $db2->seq( $chr, $start - 19 => $start );

        #print "chr$chr:$start-$stop $seq\n";
        my $totalAs = $seq =~ tr/t//;

        #print "total Ts = $totalAs\n";
        #remove if more than 13 As
        if ( $totalAs >= 13 ) {
        }
        else {
            my $left = $start - 10;

            #map closest downstream gene within 2k in the positive orientation
            my $sentence =
"SELECT WBGene, name, CDS_stop, description FROM tissues_stop_WS250 where chr='$chr' and
strand='-' and $start between $left and CDS_stop ORDER BY `tissues_stop_WS250`.`CDS_stop` ASC";
            my $query = $mySQL->prepare($sentence)
              or die "Couldn't syn prepare statement: " . DBI->errstr;
            $query->execute();
            my ( $wb, $name, $cds_stop, $description ) =
              $query->fetchrow_array();

      #print CLUSTER_MINUS"chr$chr:$start-$stop $cluster_coverage $seq $name\n";
            my $length = $cds_stop - $start;

            my $UTR_seq = $db2->seq( $chr, $start     => $cds_stop + 3 );
            my $UTR_end = $db2->seq( $chr, $start - 2 => $start + 30 );
            my $rev         = reverse_complement_as_string($UTR_seq);
```

```
            my $UTR_end_rev = reverse_complement_as_string($UTR_end);
            if ( $length > 2000 ) {
            }
            else {

                print CLUSTER_MINUS
"chr$chr:$start-$stop length=$length cluster_coverage=$cluster_coverage downstream_seq=$seq
UTR_seq=$rev $name\n";
                if ( $cluster_coverage > 200 ) {

                    #print ENDS "$UTR_end_rev\n";
                    print ENDS ">$name\_$cluster_coverage\n$UTR_end_rev\n";
                }
                else {
                }
            }
        }
    }
}
print "completed negative strand\n";

################################################################################
#########FORWARD STRAND##########################################################
################################################################################

## extract forward strands
print "extracting sam reads for POSITIVE strand from $name\n";
system(
"samtools view -F 20 $name.bam.sorted.bam | grep XM:i:0 > $name.positive.sam "
);
print "done!\n";

##map PAS end in the positive strand
################################################################################
print "map PAS end in the positive strand from $name.positive.sam\n";
open POSITIVE_SAM, "$name.positive.sam"       or die "cannot find the file";
open POSITIVE_BED, ">$name.positive.bedgraph" or die "cannot find the file";
while ( my $line = <POSITIVE_SAM> ) {

    #print "<<$line>>\n";
    chop $line;
    my @string    = split( "\t", $line );
    my $chr       = $string[2];
    my $map_start = $string[3];
    my $seq       = $string[9];
    my $cigar     = $string[5];
    $cigar =~ s/M//;
    my $seq_length = length($seq);

    if ( $cigar ne $seq_length ) {

        #print "error!!!\n";
    }
    else {
        #make final cluster of 5nt long
        my $end   = $map_start + $seq_length - 1;
        my $start = $end - 1;
        print POSITIVE_BED "$chr\t$start\t$end\n";
    }
}
close POSITIVE_SAM;
print "sorting the file\n";
system("bedtools sort -i $name.positive.bedgraph > tmp.bed");
print "done!!!\n";
print "merging the file\n";
system("bedtools merge -c 1 -o count -i tmp.bed > $name.positive.cluster");
print "done!!!\n";

#check numbers of As in the genome
print "check numbers of As in the genome\n";
open DATABASE, "$name.positive.cluster" or die "cannot find the file";
open CLUSTER_PLUS, ">$name.positive.cluster.final"
  or die "cannot find the file";
while ( my $line = <DATABASE> ) {
    chop $line;
```

```
    my @string           = split( /\t/, $line );
    my $chr              = $string[0];
    my $start            = $string[1];
    my $stop             = $string[2];
    my $cluster_coverage = $string[3];
##use only cluster with at least 5 reads
    if ( $cluster_coverage >= 5 ) {
        my $seq = $db2->seq( $chr, $stop => $stop + 19 );

        #print "chr$chr:$start-$stop $seq\n";
        my $totalAs = $seq =~ tr/a//;

        #print "total As = $totalAs\n";
        #remove if more than 13 As
        if ( $totalAs >= 13 ) {
        }
        else {
            my $right = $stop + 10;

            #map closest downstream gene within 2k in the positive orientation
            my $sentence =
"SELECT WBGene, name, CDS_stop, description FROM tissues_stop_WS250 where chr='$chr' and
strand='+' and $stop between CDS_stop and $right ORDER BY `tissues_stop_WS250`.`CDS_stop` DESC";

            #print "$sentence\n";
            my $query = $mySQL->prepare($sentence)
              or die "Couldn't syn prepare statement: " . DBI->errstr;
            $query->execute();
            my ( $wb, $name, $cds_stop, $description ) =
              $query->fetchrow_array();
            my $length = $start - $cds_stop;

            my $UTR_seq = $db2->seq( $chr, $cds_stop - 3 => $stop );
            my $UTR_end = $db2->seq( $chr, $stop - 30    => $stop + 2 );
            if ( $length > 2000 ) {
            }
            else {

                print CLUSTER_PLUS
"chr$chr:$start-$stop length=$length cluster_coverage=$cluster_coverage downstream_seq=$seq
UTR_seq=$UTR_seq $name\n";
                if ( $cluster_coverage > 200 ) {
                    print ENDS
">$name\_$cluster_coverage\_reads\_$length(nt)\n$UTR_end\n";

                    #print ENDS "$UTR_end\n";
                }
                else {
                }
            }
        }
    }
}

sub getMySQLHandle {
    my $db              = "";
    my $host            = "";
    my $port            = "";
    my $userid          = "";
    my $passw           = "";
    my $connectionInfo = "DBI:mysql:database=$db;$host:$port";
    my $dbh = DBI->connect( $connectionInfo, $userid, $passw ) or die "Couldn't
#connect to database ($db): " . DBI->errstr;
    return $dbh;
}
```