1 - Lariat Read Aligner Scripts:

README.txt

Make sure to have bowtie and bedtools in path.

usage:
perl find_lariats.pl -f file.fastq -i bowtieindex -o outputdirectory

optional flags:
-h this help message
-m mininmum fragment length
-l read lengths

Results are in file "lariat_data_table.txt"
columns are
1-directory
2-inverted alignment type
3-read ID
4-raw read sequence
5-chromosome
6-5'ss
7-3'ss
8-BP
9-raw branch site sequence

indexes included (these can be downloaded from
http://fairbrother.biomed.brown.edu/data/Lariat2016/lariat_scripts.tar.gz)

human (hg19): -i ./hg19/hg19
mouse (mm9): -i ./mm9/mm9
s. pombe (EF2): -i ./EF2/genome

```
#find_lariats.pl

#use warnings;
use Getopt::Std;

use lib ('.');
use aligner; use splicemap; use filter; use analyzer;


my %options=();
getopts("hf:i:m:l:o:",\%options);

if (defined $options{h}){
   &displayHelp();
}

elsif ((defined $options{f})&&(defined $options{i})&&(defined $options{o})){

   $file = $options{f};
   $index = $options{i};
   $outdir = $options{o};

   if (defined $options{m}){$minfraglen = $options{m};}
   else {$minfraglen = 8;}

   if (defined $options{l}){$readlen = $options{l};}
   else {$readlen = 76;}

   ##Align fragments to index

   $aligner = aligner->new($file,$index,$readlen,$minfraglen,$outdir);
   $aligner->align();

   ##Filter reads

   $filter = filter->new($file,$outdir);
   $filter->outOfOrder();

   ##Map alignments to splicemap

   $splicemap = splicemap->new($outdir,$index);

   #outoforder, both map
   $splicemap->mapSS("outoforder.txt");
   $splicemap->sameTranscript("outoforder_ss.txt");
   $splicemap->findLariats("outoforder_ss_filter.txt");
```

```
    $splicemap->resolveGaps("outoforder_ss_filter_lariats_gap.txt");




    ##Format data into bed files of bp/read alignments, and fasta files of bpseq

    $analyzer = analyzer-
>new($outdir,$index,"outoforder_ss_filter_lariats.txt","outoforder_ss_filter_lariats_overlap.txt","outofo
rder_ss_filter_lariats_gap_truelariats.txt");
    $analyzer->exact();
    $analyzer->overlap();
    $analyzer->gap();

}

else {
    &displayHelp();
}

sub displayHelp {
    print "\n\nusage:\nperl find_lariats.pl -f file.fastq -i bowtieindex -o outputdirectory\n\n";
    print "optional flags:\n-h this help message\n-m mininmum fragment length\n-l read lengths\n\n";
}
```

```
#aligner.pm

package aligner;

use warnings;

sub new
{
    my $class = shift;
    my $self = {
        _rawfile => shift,
        _index => shift,
        _readlen => shift,
        _minfraglen => shift,
        _outdir => shift,
    };

    print "\n\n****************\nparsing fastq files\n\n";

    $self->{_rawfile} =~ /(.+).fa?s?t?q/;
    $self->{_base} = $1;
    $self->{_file}=$self->{_outdir}."/seq.fastq";

    $mkdir = "mkdir ".$self->{_outdir};
    $cp = "cp ".$self->{_rawfile}." ".$self->{_file};

    system($mkdir);
    system($cp);

    bless $self, $class;
    return $self;

}

sub forwardAlign {

    my $self = shift;
    my $file = $self->{_file};
    my $index = $self->{_index};

    print "\n\n****************\nconducting forward alignment: \n";

    $command = "bowtie -v3 -p8 -k1 ".$index." ".$file." --un ".$self->{_outdir}."/unaligned.fastq >
".$self->{_outdir}."/aligned.txt";
    print $command,"\n\n";
    system($command);
    $cp1 = "cp ".$self->{_outdir}."/unaligned.fastq ".$self->{_outdir}."/unaligned_left.fastq";
    $cp2 = "cp ".$self->{_outdir}."/unaligned.fastq ".$self->{_outdir}."/unaligned_right.fastq";

    system($cp1);
    system($cp2);

}

sub fragmentAlignLeft {

    my $self = shift;

    print "\n\n****************\nprocessing left side: \n";

    my $file = $self->{_outdir}."/unaligned_left.fastq";
    my $index = $self->{_index};

    for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
        $file =~ /(.*left)/;
        $remainderfile = $1."_".$j.".fastq";
        $maxfile = $1."_multalignments_".$j.".fastq";
        $outfile = $1."_".$j.".txt";
        $alignedfq = $self->{_outdir}."/left_aligned_".$j.".fastq";
        $command = "bowtie -v0 -p8 -a -m1 --trim3 ".$j." ".$index." ".$file." --un ".$remainderfile." --max
".$maxfile." --al ".$alignedfq." > ".$outfile;
        print "\n\n*****\nexecuting: ",$command,"\n";
        system($command);

        $prevrem = $file;

        $rmmax = "rm ".$maxfile;
        $rmprevrem = "rm ".$prevrem;

        system($rmmax);
        system($rmprevrem);

        $file = $remainderfile;
    }

}

sub fragmentAlignRight {

    my $self = shift;
    print "\n\n***************\nprocessing right side: \n";

    my $file = $self->{_outdir}."/unaligned_right.fastq";
    my $index = $self->{_index};

    for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
        $file =~ /(.*right)/;
        $remainderfile = $1."_".$j.".fastq";
        $maxfile = $1."_multalignments_".$j.".fastq";
        $outfile = $1."_".$j.".txt";
        $alignedfq = $self->{_outdir}."/right_aligned_".$j.".fastq";
        $command = "bowtie -v0 -p8 -a -m1 --trim5 ".$j." ".$index." ".$file." --un ".$remainderfile." --max
".$maxfile." --al ".$alignedfq." > ".$outfile;
        print "\n\n*****\nexecuting: ",$command,"\n";
        system($command);

        $prevrem = $file;

        $rmmax = "rm ".$maxfile;
        $rmprevrem = "rm ".$prevrem;

        system($rmmax);
        system($rmprevrem);

        $file = $remainderfile;

    }

}


sub align {

    my $self = shift;

    &forwardAlign($self);
    &fragmentAlignLeft($self);
    &fragmentAlignRight($self);
    &mergeSides($self);
    &rmTempFiles($self);

}


sub mergeSides {
```

```
my $self = shift;

print "\n\n****************\ncompiling left and right sides: \n";

$alldatafile = $self->{_outdir}."/new_alignments.txt";
open (FHout, ">$alldatafile") or die $!;

my %lefthits;

for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
    $leftsam = $self->{_outdir}."/unaligned_left_".$j.".txt";
    open (FH, "$leftsam");
    while (<FH>){
        $data = $_;
        chomp $data;
        $id = (split /\t/,$data)[0];
        if ($id =~ /\@/){}
        else {
            $id = (split /\t/,$data)[0];
            $lefthits{$id}[0] = 0;
            $lefthits{$id}[1] = $data;
            $lefthits{$id}[2] = "0\t0\t0\t0\t0\t0\t0\t0";
        }
    }
    close FH;
}

for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
    $leftfq = $self->{_outdir}."/left_aligned_".$j.".fastq";
    if (-e $leftfq){
    open (FH,"$leftfq");
    $line = 1;
    while (<FH>){
        chomp $_;
        if ($line ==1){
            $id = $_;
            $id =~ s/^@//g;
        }
        elsif (($line==2)&&($_ =~ /([ACGTNacgtn]+)/)){
            $seq = $1;
            $lefthits{$id}[0] = $seq;
        }
        $line++;
        if ($line==5){
            $line = 1;
        }
```

```
  }}
}

for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
  $rightsam = $self->{_outdir}."/unaligned_right_".$j.".txt";
  open (FH, "$rightsam");
  while (<FH>){
    $data = $_;
    chomp $data;
    $id = (split /\t/,$data)[0];
    if ($id =~ /\@/){}
    else {
      $id = (split /\t/,$data)[0];
      if (exists $lefthits{$id}[0]){
        $lefthits{$id}[2] = $data;
      }
      else {
        $lefthits{$id}[0] = 0;
        $lefthits{$id}[2] = $data;
        $lefthits{$id}[1] = "0\t0\t0\t0\t0\t0\t0\t0";
      }
    }
  }
  close FH;
}

for (my $j = $self->{_minfraglen}; $j<= ($self->{_readlen}-$self->{_minfraglen}); $j++){
  $rightfq = $self->{_outdir}."/right_aligned_".$j.".fastq";
  if (-e $rightfq){
  open (FH,"$rightfq");
  $line = 1;
  while (<FH>){
    chomp $_;
    if ($line ==1){
      $id = $_;
      $id =~ s/^@//g;
    }
    elsif (($line==2)&&($_ =~ /([ACGTNacgtn]+)/)){
      $seq = $1;
      $lefthits{$id}[0] = $seq;
    }
    $line++;
    if ($line==5){
      $line = 1;
    }
  }}
```

```
    }

    foreach my $key (keys %lefthits){
        print FHout $key,"\t";
        print FHout $lefthits{$key}[0],"\t";
        print FHout $lefthits{$key}[1],"\t";
        print FHout $lefthits{$key}[2],"\n";
    }
    close FHout;
}

sub rmTempFiles {

    my $self = shift;

    $rm = "rm ".$self->{_outdir}."/*.fastq ".$self->{_outdir}."/unaligned*.txt";
    print "\n",$rm,"\n\n";
    system($rm);

}

1;
```

```
#analyzer.pm


package analyzer;
#use warnings;


sub new
{
   my $class = shift;
   my $self = {
       _outdir => shift,
       _index => shift,
       _exact => shift,
       _overlap => shift,
       _gap => shift,
   };

   $rm = "rm ".$self->{_outdir}."/lariat_data_table.txt";
   system($rm);

   bless $self, $class;
   return $self;
}

sub exact
{
   my $self = shift;
   $file = $self->{_outdir}."/".$self->{_exact};
   $fileout = $self->{_outdir}."/lariat_data_table.txt";
   open (FH, $file) or die $!;
   open (FHout, ">>$fileout") or die $!;
   while (<FH>){
       $line = $_;
       $id = (split /\t/,$line)[0];
       $seq = (split /\t/,$line)[1];
       $readstrand = (split /\t/,$line)[3];
       $chrom = (split /\t/,$line)[4];
       $headseq = (split /\t/,$line)[6];
       $tailseq = (split /\t/,$line)[14];
       $headcoord1 = (split /\t/,$line)[5];
       $headcoord2 = length($headseq)+$headcoord1;
       $tailcoord1 = (split /\t/,$line)[13];
       $tailcoord2 = length($tailseq)+$tailcoord1;
       my @ss;
       for (my $j = 18; $j<=25; $j++){
            push(@ss,((split /\t/,$line)[$j]));
        }
        for (my $j = 0; $j<=8; $j++){
            if ($ss[$j] =~ /\w+:(\d+)_([+-])/){
                $coord = $1;
                $strand = $2;
                $ss_coord[$j][0] = $coord;
                $ss_coord[$j][1] = $strand;
            }
            else {
                $ss_coord[$j][0] = undef;
                $ss_coord[$j][1] = undef;
            }
        }
        #sense, positive strand
        if (($ss_coord[2][0]==$tailcoord1)&&($ss_coord[2][1] eq $readstrand)&&($readstrand eq
"+")&&($ss_coord[2][1] ne undef)){
            $threeprss = $ss_coord[5][0];
            print FHout $self-
>{_outdir},"\texact\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$tailcoord1,"\t",$threeprss,"\t",$headcoord2,"\t";
            $bpseq1 = $headcoord2-5;
            $bpseq2 = $headcoord2+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #sense, negative strand
        if (($ss_coord[3][0]==$tailcoord2)&&($ss_coord[3][1] eq $readstrand)&&($readstrand eq "-
")&&($ss_coord[3][1] ne undef)){
            $threeprss = $ss_coord[4][0];
            print FHout $self->{_outdir},"\texact\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$tailcoord2,"\t",$threeprss,"\t",$headcoord1,"\t";
            $bpseq1 = $headcoord1-5;
            $bpseq2 = $headcoord1+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #antisense, positive strand
        if (($ss_coord[0][0]==$headcoord1)&&($ss_coord[0][1] ne $readstrand)&&($readstrand eq "-
")&&($ss_coord[0][1] ne undef)){
            $threeprss = $ss_coord[7][0];
            print FHout $self-
>{_outdir},"\texact\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$headcoord1,"\t",$threeprss,"\t",$tailcoord2,"\t";
            $bpseq1 = $tailcoord2-5;
            $bpseq2 = $tailcoord2+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #antisense, negative strand
        if (($ss_coord[1][0]==$headcoord2)&&($ss_coord[1][1] ne $readstrand)&&($readstrand eq
"+")&&($ss_coord[1][1] ne undef)){
            $threeprss = $ss_coord[6][0];
            print FHout $self->{_outdir},"\texact\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$headcoord2,"\t",$threeprss,"\t",$tailcoord1,"\t";
            $bpseq1 = $tailcoord1-5;
            $bpseq2 = $tailcoord1+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
    }
    close FHout; close FH;
}

sub overlap {
    $self = shift;
    $file = $self->{_outdir}."/".$self->{_overlap};
    $fileout = $self->{_outdir}."/lariat_data_table.txt";
    open (FH, $file) or die $!;
    open (FHout, ">>$fileout") or die $!;
    while (<FH>){
        $line = $_;
        $id = (split /\t/,$line)[0];
        $seq = (split /\t/,$line)[1];
        $readstrand = (split /\t/,$line)[3];
        $chrom = (split /\t/,$line)[4];
        $headseq = (split /\t/,$line)[6];
        $tailseq = (split /\t/,$line)[14];
        $headcoord1 = (split /\t/,$line)[5];
        $headcoord2 = length($headseq)+$headcoord1;
        $tailcoord1 = (split /\t/,$line)[13];
        $tailcoord2 = length($tailseq)+$tailcoord1;
        $overlap = (length($headseq)+length($tailseq))-length($seq);




        my @ss;
        for (my $j = 18; $j<=25; $j++){
                push(@ss,((split /\t/,$line)[$j]));
            }
        for (my $j = 0; $j<=8; $j++){
            if ($ss[$j] =~ /\w+:(\d+)_([+-])/){
                $coord = $1;
                $strand = $2;
                $ss_coord[$j][0] = $coord;
                $ss_coord[$j][1] = $strand;
            }
            else {
                $ss_coord[$j][0] = undef;
                $ss_coord[$j][1] = undef;
            }
        }
        #sense, positive strand
        #if ((abs($ss_coord[2][0]-$tailcoord1)<=$overlap)&&($ss_coord[2][1] eq
$readstrand)&&($readstrand eq "+")&&($ss_coord[2][1] ne undef)){
        if ((abs($ss_coord[2][0]-
$tailcoord1)<=$overlap)&&($ss_coord[2][0]>=$tailcoord1)&&($ss_coord[2][1] eq
$readstrand)&&($readstrand eq "+")&&($ss_coord[2][1] ne undef)){
            $threeprss = $ss_coord[5][0];
            $excess = $ss_coord[2][0] - $tailcoord1;

            $tailcoord1 = $ss_coord[2][0];
            $headcoord2 = $headcoord2-($overlap-$excess);
            print FHout $self->{_outdir},"\toverlap
",$overlap,"\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$tailcoord1,"\t",$threeprss,"\t",$headcoord2,"\t";
            $bpseq1 = $headcoord2-5;
            $bpseq2 = $headcoord2+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
               $_ =~s/\s//g;
               if ($_ =~ /chr/){}
               elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                  $seq = $1;
                  $seq =~ tr/a-z/A-Z/;
                  print FHout $seq;
               }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #sense, negative strand
        if (((-1)*($ss_coord[3][0]-
$tailcoord2)<=$overlap)&&($tailcoord2>=$ss_coord[3][0])&&($ss_coord[3][1] eq
$readstrand)&&($readstrand eq "-")&&($ss_coord[3][1] ne undef)){

            $threeprss = $ss_coord[4][0];
            $excess = $tailcoord2-$ss_coord[3][0];
            $tailcoord2 = $ss_coord[3][0];
            $headcoord1 = $headcoord1+($overlap-$excess);
            print FHout $self->{_outdir},"\toverlap ",$overlap,"\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$tailcoord2,"\t",$threeprss,"\t",$headcoord1,"\t";
            $bpseq1 = $headcoord1-5;
            $bpseq2 = $headcoord1+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #antisense, positive strand
        if ((($ss_coord[0][0]-$headcoord1)<=$overlap)&&($ss_coord[0][0]
>=$headcoord1)&&($ss_coord[0][1] ne $readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne
undef)){
            $threeprss = $ss_coord[7][0];
            $excess = $ss_coord[0][0]-$headcoord1;
            $headcoord1 = $ss_coord[0][0];
            $tailcoord2 = $tailcoord2-($overlap-$excess);
            print FHout $self->{_outdir},"\toverlap
",$overlap,"\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$headcoord1,"\t",$threeprss,"\t",$tailcoord2,"\t";
            $bpseq1 = $tailcoord2-5;
            $bpseq2 = $tailcoord2+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
        print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
        close FHbed;
        $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
        system($command);
        open (FHfasta, $fhfasta) or die $!;
        while (<FHfasta>){
          $_ =~s/\s//g;
          if ($_ =~ /chr/){}
          elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
            $seq = $1;
            $seq =~ tr/a-z/A-Z/;
            print FHout $seq;
          }
        }
         print FHout "\n";
        close FHfasta;
        $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
        $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
        system($command1);
        system($command2);
      }
      #antisense, negative strand
      if (((-1)*($ss_coord[1][0]-
$headcoord2)<=$overlap)&&($headcoord2>=$ss_coord[1][0])&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
        $threeprss = $ss_coord[6][0];
        $excess = $headcoord2-$ss_coord[1][0];
        $headcoord2 = $ss_coord[1][0];
        $tailcoord1 = $tailcoord1+($overlap-$excess);
        print FHout $self->{_outdir},"\toverlap ",$overlap,"\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$headcoord2,"\t",$threeprss,"\t",$tailcoord1,"\t";
        $bpseq1 = $tailcoord1-5;
        $bpseq2 = $tailcoord1+5;
        $fhbed = $self->{_outdir}."/tempcoord.bed";
        $fhfasta = $self->{_outdir}."/tempseq.fasta";
        open (FHbed, ">$fhbed") or die $!;
        print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
        close FHbed;
        $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
        system($command);
        open (FHfasta, $fhfasta) or die $!;
        while (<FHfasta>){
          $_ =~s/\s//g;
          if ($_ =~ /chr/){}
          elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
            $seq = $1;
```

```
                $seq =~ tr/a-z/A-Z/;
                print FHout $seq;
            }
        }
         print FHout "\n";
        close FHfasta;
        $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
        $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
        system($command1);
        system($command2);
    }
  }
  close FHout; close FH;



}

sub gap {
  $self = shift;
  $file = $self->{_outdir}."/".$self->{_gap};
  $fileout = $self->{_outdir}."/lariat_data_table.txt";
  open (FH, $file) or die $!;
  open (FHout, ">>$fileout") or die $!;
  while (<FH>){
     $line = $_;
     $id = (split /\t/,$line)[0];
     $seq = (split /\t/,$line)[1];
     $readstrand = (split /\t/,$line)[3];
     $chrom = (split /\t/,$line)[4];
     $headseq = (split /\t/,$line)[6];
     $tailseq = (split /\t/,$line)[14];
     $headcoord1 = (split /\t/,$line)[5];
     $headcoord2 = length($headseq)+$headcoord1;
     $tailcoord1 = (split /\t/,$line)[13];
     $tailcoord2 = length($tailseq)+$tailcoord1;
     $gap = length($seq)-(length($headseq)+length($tailseq));
     my @ss;
     for (my $j = 18; $j<=25; $j++){
           push(@ss,((split /\t/,$line)[$j]));
        }
     for (my $j = 0; $j<=8; $j++){
        if ($ss[$j] =~ /\w+:(\d+)_([+-])/){
           $coord = $1;
           $strand = $2;
           $ss_coord[$j][0] = $coord;
```

```
            $ss_coord[$j][1] = $strand;
        }
        else {
            $ss_coord[$j][0] = undef;
            $ss_coord[$j][1] = undef;
        }
    }
    #sense, positive strand
    if ((abs($ss_coord[2][0]-$tailcoord1)<=$gap)&&($ss_coord[2][1] eq $readstrand)&&($readstrand
eq "+")&&($ss_coord[2][1] ne undef)){
        $threeprss = $ss_coord[5][0];
        $remainder = $tailcoord1-$ss_coord[2][0];
        $tailcoord1 = $ss_coord[2][0];
        $headcoord2 = $headcoord2+($gap-$remainder);
        print FHout $self->{_outdir},"\tgap
",$gap,"\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$tailcoord1,"\t",$threeprss,"\t",$headcoord2,"\t";
        $bpseq1 = $headcoord2-5;
        $bpseq2 = $headcoord2+5;
        $fhbed = $self->{_outdir}."/tempcoord.bed";
        $fhfasta = $self->{_outdir}."/tempseq.fasta";
        open (FHbed, ">$fhbed") or die $!;
        print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
        close FHbed;
        $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
        system($command);
        open (FHfasta, $fhfasta) or die $!;
        while (<FHfasta>){
            $_ =~s/\s//g;
            if ($_ =~ /chr/){}
            elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                $seq = $1;
                $seq =~ tr/a-z/A-Z/;
                print FHout $seq;
            }
        }
        print FHout "\n";
        close FHfasta;
        $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
        $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
        system($command1);
        system($command2);
    }
    #sense, negative strand
    if ((abs($ss_coord[3][0]-$tailcoord2)<=$gap)&&($ss_coord[3][1] eq $readstrand)&&($readstrand
eq "-")&&($ss_coord[3][1] ne undef)){
        $threeprss = $ss_coord[4][0];
```

```
            $remainder = $ss_coord[3][0]-$tailcoord2;
            $tailcoord2 = $ss_coord[3][0];
            $headcoord1 = $headcoord1-($gap-$remainder);
            print FHout $self->{_outdir},"\tgap ",$gap,"\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$tailcoord2,"\t",$threeprss,"\t",$headcoord1,"\t";
            $bpseq1 = $headcoord1-5;
            $bpseq2 = $headcoord1+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
            print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #antisense, positive strand
        if ((abs($ss_coord[0][0]-$headcoord1)<=$gap)&&($ss_coord[0][1] ne
$readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne undef)){
            $threeprss = $ss_coord[7][0];
            $remainder = $headcoord1-$ss_coord[0][0];
            $headcoord1 = $ss_coord[0][0];
            $tailcoord2 = $tailcoord2+($gap-$remainder);
            print FHout $self->{_outdir},"\tgap
",$gap,"\t",$id,"\t",$seq,"\t",$chrom,"\t+\t",$headcoord1,"\t",$threeprss,"\t",$tailcoord2,"\t";
            $bpseq1 = $tailcoord2-5;
            $bpseq2 = $tailcoord2+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t+\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
        #antisense, negative strand
        if ((abs($ss_coord[1][0]-$headcoord2)<=$gap)&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
            $threeprss = $ss_coord[6][0];
            $remainder = $ss_coord[1][0]-$headcoord2;
            $headcoord2 = $ss_coord[1][0];
            $tailcoord1 = $tailcoord1-($gap-$remainder);
            print FHout $self->{_outdir},"\tgap ",$gap,"\t",$id,"\t",$seq,"\t",$chrom,"\t-
\t",$headcoord2,"\t",$threeprss,"\t",$tailcoord1,"\t";
            $bpseq1 = $tailcoord1-5;
            $bpseq2 = $tailcoord1+5;
            $fhbed = $self->{_outdir}."/tempcoord.bed";
            $fhfasta = $self->{_outdir}."/tempseq.fasta";
            open (FHbed, ">$fhbed") or die $!;
            print FHbed $chrom,"\t",$bpseq1,"\t",$bpseq2,"\t",$id,"\t0\t-\n";
            close FHbed;
            $command = "bedtools getfasta -s -fi ".$self->{_index}.".fa -bed ".$fhbed." -fo ".$fhfasta;
            system($command);
            open (FHfasta, $fhfasta) or die $!;
            while (<FHfasta>){
                      $_ =~s/\s//g;
                if ($_ =~ /chr/){}
                elsif ($_ =~ /^([ACGTUNacgtun]+)$/){
                    $seq = $1;
                    $seq =~ tr/a-z/A-Z/;
                    print FHout $seq;
                }
            }
             print FHout "\n";
            close FHfasta;
            $command1 = "rm ".$self->{_outdir}."/tempcoord.bed";
            $command2 = "rm ".$self->{_outdir}."/tempseq.fasta";
            system($command1);
            system($command2);
        }
    }
    close FHout; close FH;
}

1;
```

```
#filter.pm

package filter;

use warnings;

sub new
{
   my $class = shift;
   my $self = {
      _rawfile => shift,
      _outdir => shift,
   };

   $self->{_rawfile} =~ /(.+).fa?s?t?q/;
   $self->{_file}=$self->{_outdir}."/new_alignments.txt";

   bless $self, $class;
   return $self;

}

sub outOfOrder
{
   my $self = shift;
   print "\n\n***************\nfiltering out of order reads\n\nopening file: ";
   print $self->{_file},"\n\n";

   open (FH, $self->{_file});

   $halffile = $self->{_outdir}."/halfmap.txt";
   $outoforderfile = $self->{_outdir}."/outoforder.txt";

   open (FHhalf,">$halffile") or die $!;
   open (FHoutoforder,">$outoforderfile") or die $!;

   while (<FH>){

      $line = $_;
      $readid = (split /\t/,$_)[0];
      $readseq = (split /\t/,$_)[1];
      $headstrand = (split /\t/,$_)[3];
      $headchrom = (split /\t/,$_)[4];
      $headcoord = (split /\t/,$_)[5];
      $tailstrand = (split /\t/,$_)[11];
```

```
        $tailchrom = (split /\t/,$_)[12];
        $tailcoord = (split /\t/,$_)[13];

        if (($headchrom ne 0)&&($tailchrom ne 0)){
           if (($headchrom eq $tailchrom)&&($headstrand eq $tailstrand)){
              if (($headstrand eq "+")&&($headcoord > $tailcoord)){
                 print FHoutoforder $line;
              }
              elsif (($headstrand eq "-")&&($headcoord < $tailcoord)){
                 print FHoutoforder $line;
              }
           }
        }
        else {
           print FHhalf $line;
        }
    }
    close FH;
    close FHhalf;
    closeFHoutoforder;
}

sub setFile
{
    my $self = shift;
    $self->{_file} = shift;

}

sub getFile
{
    my $self = shift;
    return $self->{_file};
}

1;
```

```
#splicemap.pm

package splicemap;



#use warnings;

sub new
{
    my $class = shift;
    my $self = {
        _outdir => shift,
        _index => shift,
    };

    my $fh = $self->{_index}."_ss_table.txt";

    print "\n\n****************\nbuilding splice site hash map\n\n";

    my %splice5pr=();
    my %splice3pr=();


    open (FH,"$fh") or die $!;
    while (<FH>){

        $chr = (split /\t/,$_)[1];
        $id = (split /\t/,$_)[0];
        $strand = (split /\t/,$_)[2];
        $exoncount = (split /\t/,$_)[5];

        if ($exoncount =~ /\d+/){
            $exonstarts = (split /\t/,$_)[6];
            $exonstops = (split /\t/,$_)[7];
            @starts = (split /\,/,$exonstarts);
            @stops = (split /\,/,$exonstops);
            chomp $exonstops;

            for (my $j = 1; $j<$exoncount; $j++){
                if ($strand eq "+"){
                    $fiveprid = $chr.":".$stops[$j-1]."_".$strand;
                    $threeprid = $chr.":".$starts[$j]."_".$strand;
                }
                else {
                    $fiveprid = $chr.":".$starts[$j]."_".$strand;
                    $threeprid = $chr.":".$stops[$j-1]."_".$strand;
                }

                if (exists $splice5pr{$fiveprid}){
                    $value = $splice5pr{$fiveprid};
                    $newvalue = $value.",".$id;
                    $splice5pr{$fiveprid} = $newvalue;
                }
                else {
                    $splice5pr{$fiveprid} = $id;
                }

                if (exists $splice3pr{$threeprid}){
                    $value = $splice3pr{$threeprid};
                    $newvalue = $value.",".$id;
                    $splice3pr{$threeprid} = $newvalue;
                }
                else {
                    $splice3pr{$threeprid} = $id;
                }
            }
        }
    }

    $self->{_ss5pr} = \%splice5pr;
    $self->{_ss3pr} = \%splice3pr;

    bless $self, $class;
    return $self;

}

sub mapSS
{
    $self = shift;
    $group = shift;

    $group =~ /(.+).txt/;
    $base = $1;

    $fh = $self->{_outdir}."/".$group;
    $fhout = $self->{_outdir}."/".$base."_ss.txt";

    %ss5 = %{$self->{_ss5pr}};
    %ss3 = %{$self->{_ss3pr}};
```

```
open (FH, $fh) or die $!;
open (FHout, ">$fhout") or die $!;

while (<FH>){
   #loop through and lookup splice hash as you go
   $data = $_;
   chomp $data;

   $seq = (split /\t/,$data)[1];
   $length = length($seq);

   $headchrom = (split /\t/,$data)[4];
   $headstrand = (split /\t/,$data)[3];
   $headcoord1 = (split /\t/,$data)[5];
   $headseq = (split /\t/,$data)[6];
   $headlen = length($headseq);
   $headcoord2 = $headcoord1+$headlen;


   $tailstrand = (split /\t/,$data)[11];
   $tailchrom = (split /\t/,$data)[12];
   $tailcoord1 = (split /\t/,$data)[13];
   $tailseq = (split /\t/,$data)[14];
   $taillen = length($tailseq);
   $tailcoord2 = $tailcoord1+$taillen;


   my @lookup;

   print FHout $data,"\t";

   $lookupkey[0] = $headchrom.":".$headcoord1."_".$headstrand;
   $lookupkey[1] = $headchrom.":".$headcoord2."_".$headstrand;
   $lookupkey[2] = $tailchrom.":".$tailcoord1."_".$tailstrand;
   $lookupkey[3] = $tailchrom.":".$tailcoord2."_".$tailstrand;




   for (my $j = 0; $j<=3; $j++){
      $foundflag = 0;
      $dist = 0;
      if ($lookupkey[$j] =~ /(\w+):(\d+)_([+-])/){
      $chrom = $1; $coord = $2; $strand = $3;
      while (($dist<=1000)&&($foundflag ==0)){
        $newcoord = $coord+$dist;
        $lookup = $chrom.":".$newcoord."_+";
        if ((exists $ss5{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss5{$lookup}),"\t";
            $foundflag = 1;
        }
        $newcoord = $coord-$dist;
        $lookup = $chrom.":".$newcoord."_+";
        if ((exists $ss5{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss5{$lookup}),"\t";
            $foundflag = 1;
        }
        $newcoord = $coord+$dist;
        $lookup = $chrom.":".$newcoord."_-";
        if ((exists $ss5{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss5{$lookup}),"\t";
            $foundflag = 1;
        }
        $newcoord = $coord-$dist;
        $lookup = $chrom.":".$newcoord."_-";
        if ((exists $ss5{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss5{$lookup}),"\t";
            $foundflag = 1;
        }
        $dist++;
    }}
    else {print "error: $lookupkey[$j]\n";}
    if ($foundflag ==0){print FHout "null\t";}
}

for (my $j = 0; $j<=3; $j++){
    $foundflag = 0;
    $dist = 0;
    if ($lookupkey[$j] =~ /(\w+):(\d+)_([+-])/){
    $chrom = $1; $coord = $2; $strand = $3;
    while (($dist<=1000)&&($foundflag ==0)){
        $newcoord = $coord+$dist;
        $lookup = $chrom.":".$newcoord."_+";
        if ((exists $ss3{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss3{$lookup}),"\t";
            $foundflag = 1;
        }
        $newcoord = $coord-$dist;
        $lookup = $chrom.":".$newcoord."_+";
        if ((exists $ss3{$lookup})&&($foundflag==0)){
            print FHout ($lookup.":".$ss3{$lookup}),"\t";
```

```
                $foundflag = 1;
            }
            $newcoord = $coord+$dist;
            $lookup = $chrom.":".$newcoord."_-";
            if ((exists $ss3{$lookup})&&($foundflag==0)){
                print FHout ($lookup.":".$ss3{$lookup}),"\t";
                $foundflag = 1;
            }
            $newcoord = $coord-$dist;
            $lookup = $chrom.":".$newcoord."_-";
            if ((exists $ss3{$lookup})&&($foundflag==0)){
                print FHout ($lookup.":".$ss3{$lookup}),"\t";
                $foundflag = 1;
            }
            $dist++;
        }}
        else {print "error: $lookupkey[$j]\n";}
        if ($foundflag ==0){print FHout "null\t";}
    }
    print FHout "\n";

  }
}

sub sameTranscript {
  $self = shift;
  $group = shift;

  $group =~ /(.+).txt/;
  $fhout = $self->{_outdir}."/".$1."_filter.txt";
  $fh = $self->{_outdir}."/".$group;
  open (FH, $fh) or die $!;
  open (FHout, ">$fhout") or die $!;


  while (<FH>){
    $line = $_;
    my @ss;
    my @genes;
    if ($_ =~ /chr/){
      for (my $j = 18; $j<=25; $j++){
        push(@ss,((split /\t/,$line)[$j]));
      }
      for (my $j = 0; $j<8; $j++){
        if ($ss[$j] =~ /chr\w+:\d+.+:(.+)/){
          $genelist = $1;
```

```
                $genes[$j] = $genelist;
            }
            elsif ($ss[$j] =~ /null/) {
                $genes[$j] = "";
            }
        }
        $headgenes = $genes[0].",".$genes[1].",".$genes[4].",".$genes[5];
        $tailgenes = $genes[2].",".$genes[3].",".$genes[6].",".$genes[7];

        @headgenes = (split /\,/,$headgenes);
        @tailgenes = (split /\,/,$tailgenes);

        $numheadgenes = @headgenes;
        $numtailgenes = @tailgenes;

        $flag = 0;

        for (my $j = 0; $j<$numheadgenes; $j++){
            for (my $k = 0; $k<$numtailgenes; $k++){
                if ($headgenes[$j] eq $tailgenes[$k]){
                    $flag = 1;
                }
            }
        }

        if ($flag==1){
            print FHout $line;
        }
    }
  }
  close FH;


}




sub findLariats {
  $self = shift;
  $group = shift;

  $group =~ /(.+).txt/;
  $fhout = $self->{_outdir}."/".$1."_lariats.txt";
  $fhoutoverlap = $self->{_outdir}."/".$1."_lariats_overlap.txt";
  $fhoutgap = $self->{_outdir}."/".$1."_lariats_gap.txt";
```

```
    $fh = $self->{_outdir}."/".$group;

    open (FH, $fh) or die $!;
    open (FHout,">$fhout") or die $!;
    open (FHoutoverlap,">$fhoutoverlap") or die $!;
    open (FHoutgap, ">$fhoutgap") or die $!;

    print "opening: $fh\n\n";

    while (<FH>){
        $line = $_;
        chomp $line;
        $seq = (split /\t/,$line)[1];
        $headseq = (split /\t/,$line)[6];
        $tailseq = (split /\t/,$line)[14];
        $readstrand = (split /\t/,$line)[3];
        #exact alignment
        if ((length($seq)) == ((length($headseq))+(length($tailseq)))){
            my @ss; my @ss_coord;
            $headcoord1 = (split /\t/,$_)[5];
            $headcoord2 = length($headseq)+$headcoord1;
            $tailcoord1 = (split /\t/,$line)[13];
            $tailcoord2 = length($tailseq)+$tailcoord1;
            for (my $j = 18; $j<=25; $j++){
                push(@ss,((split /\t/,$line)[$j]));
            }
            for (my $j = 0; $j<8; $j++){
                if ($ss[$j] =~ /chr\w+:(\d+)_([+-])/){
                    $coord = $1;
                    $strand = $2;
                    $ss_coord[$j][0] = $coord;
                    $ss_coord[$j][1] = $strand;
                }
                else {
                    $ss_coord[$j][0] = undef;
                    $ss_coord[$j][1] = undef;
                }
            }
            $lariatflag = 0;
            #sense, positive strand
            if (($ss_coord[2][0]==$tailcoord1)&&($ss_coord[2][1] eq $readstrand)&&($readstrand eq
"+")&&($ss_coord[2][1] ne undef)){
                $lariatflag = 1;
            }
            #sense, negative strand
            if (($ss_coord[3][0]==$tailcoord2)&&($ss_coord[3][1] eq $readstrand)&&($readstrand eq "-
")&&($ss_coord[3][1] ne undef)){
                $lariatflag = 1;
            }
            #antisense, positive strand
            if (($ss_coord[0][0]==$headcoord1)&&($ss_coord[0][1] ne $readstrand)&&($readstrand eq "-
")&&($ss_coord[0][1] ne undef)){
                $lariatflag = 1;
            }
            #antisense, negative strand
            if (($ss_coord[1][0]==$headcoord2)&&($ss_coord[1][1] ne $readstrand)&&($readstrand eq
"+")&&($ss_coord[1][1] ne undef)){
                $lariatflag = 1;
            }

            if ($lariatflag==1){
                print FHout $line,"\n";
            }
        }
        #overlapped alignment
        elsif ((length($seq)) < ((length($headseq))+(length($tailseq)))){
            my @ss; my @ss_coord;
            $headcoord1 = (split /\t/,$_)[5];
            $headcoord2 = length($headseq)+$headcoord1;
            $tailcoord1 = (split /\t/,$line)[13];
            $tailcoord2 = length($tailseq)+$tailcoord1;
            $overlap = (length($headseq)+(length($tailseq)))-length($seq);
            for (my $j = 18; $j<=25; $j++){
                push(@ss,((split /\t/,$line)[$j]));
            }
            for (my $j = 0; $j<8; $j++){
                if ($ss[$j] =~ /chr\w+:(\d+)_([+-])/){
                    $coord = $1;
                    $strand = $2;
                    $ss_coord[$j][0] = $coord;
                    $ss_coord[$j][1] = $strand;
                }
                else {
                    $ss_coord[$j][0] = undef;
                    $ss_coord[$j][1] = undef;
                }
            }
            $lariatflag = 0;
            #sense, positive strand
            #if ((abs($ss_coord[2][0]-$tailcoord1)<=$overlap)&&($ss_coord[2][1] eq
$readstrand)&&($readstrand eq "+")&&($ss_coord[2][1] ne undef)){
```

```
          if ((abs($ss_coord[2][0]-
$tailcoord1)<=$overlap)&&($ss_coord[2][0]>=$tailcoord1)&&($ss_coord[2][1] eq
$readstrand)&&($readstrand eq "+")&&($ss_coord[2][1] ne undef)){
              $lariatflag = 1;
          }
          #sense, negative strand
          #if ((abs($ss_coord[3][0]-$tailcoord2)<=$overlap)&&($ss_coord[3][1] eq
$readstrand)&&($readstrand eq "-")&&($ss_coord[3][1] ne undef)){
          if (((-1)*($ss_coord[3][0]-
$tailcoord2)<=$overlap)&&($tailcoord2>=$ss_coord[3][0])&&($ss_coord[3][1] eq
$readstrand)&&($readstrand eq "-")&&($ss_coord[3][1] ne undef)){
                $lariatflag = 1;
          }
          #antisense, positive strand
          #if ((abs($ss_coord[0][0]-$headcoord1)<=$overlap)&&($ss_coord[0][1] ne
$readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne undef)){
          if ((($ss_coord[0][0]-$headcoord1)<=$overlap)&&($ss_coord[0][0]
>=$headcoord1)&&($ss_coord[0][1] ne $readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne
undef)){
               $lariatflag = 1;
          }
          #antisense, negative strand
          #if ((abs($ss_coord[1][0]-$headcoord2)<=$overlap)&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
          if (((-1)*($ss_coord[1][0]-
$headcoord2)<=$overlap)&&($headcoord2>=$ss_coord[1][0])&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
              $lariatflag = 1;
          }

          if ($lariatflag==1){
              print FHoutoverlap $line,"\n";
          }
       }
       #gapped alignment
       elsif ((length($seq)) > ((length($headseq))+(length($tailseq)))){
          my @ss; my @ss_coord;
          $headcoord1 = (split /\t/,$_)[5];
          $headcoord2 = length($headseq)+$headcoord1;
          $tailcoord1 = (split /\t/,$line)[13];
          $tailcoord2 = length($tailseq)+$tailcoord1;
          $gap = length($seq)-(length($headseq)+(length($tailseq)));
          for (my $j = 18; $j<=25; $j++){
              push(@ss,((split /\t/,$line)[$j]));
          }
          for (my $j = 0; $j<8; $j++){
              if ($ss[$j] =~ /chr\w+:(\d+)_([+-])/){
                  $coord = $1;
                  $strand = $2;
                  $ss_coord[$j][0] = $coord;
                  $ss_coord[$j][1] = $strand;
              }
              else {
                  $ss_coord[$j][0] = undef;
                  $ss_coord[$j][1] = undef;
              }
          }
          $lariatflag = 0;
          #sense, positive strand
          if ((abs($ss_coord[2][0]-$tailcoord1)<=$gap)&&($ss_coord[2][1] eq
$readstrand)&&($readstrand eq "+")&&($ss_coord[2][1] ne undef)){
              $lariatflag = 1;
          }
          #sense, negative strand
          if ((abs($ss_coord[3][0]-$tailcoord2)<=$gap)&&($ss_coord[3][1] eq
$readstrand)&&($readstrand eq "-")&&($ss_coord[3][1] ne undef)){
              $lariatflag = 1;
          }
          #antisense, positive strand
          if ((abs($ss_coord[0][0]-$headcoord1)<=$gap)&&($ss_coord[0][1] ne
$readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne undef)){
              $lariatflag = 1;
          }
          #antisense, negative strand
          if ((abs($ss_coord[1][0]-$headcoord2)<=$gap)&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
              $lariatflag = 1;
          }

          if ($lariatflag==1){
              print FHoutgap $line,"\n";
          }
       }

    }
    close FH; close FHoutoverlap; close FHout; close FHoutgap;

}

sub resolveGaps {

    $self = shift;
```

```
$group = shift;

$group =~ /(.+).txt/;
$base = $1;
$fhhead = $self->{_outdir}."/".$base."_heads.bed";
$fhtail = $self->{_outdir}."/".$base."_tails.bed";
$fh = $self->{_outdir}."/".$group;

open (FH, $fh) or die $!;
open (FHhead,">$fhhead") or die $!;
open (FHtail,">$fhtail") or die $!;

 while (<FH>){
    $line = $_;
    chomp $line;
    $id = (split /\t/,$line)[0];
    $chrom = (split /\t/,$line)[4];
    $seq = (split /\t/,$line)[1];
    $headseq = (split /\t/,$line)[6];
    $tailseq = (split /\t/,$line)[14];
    $readstrand = (split /\t/,$line)[3];

    my @ss; my @ss_coord;
    $headcoord1 = (split /\t/,$_)[5];
    $headcoord2 = length($headseq)+$headcoord1;
    $tailcoord1 = (split /\t/,$line)[13];
    $tailcoord2 = length($tailseq)+$tailcoord1;
    $gap = length($seq)-(length($headseq)+(length($tailseq)));

    for (my $j = 18; $j<=25; $j++){
        push(@ss,((split /\t/,$line)[$j]));
    }
    for (my $j = 0; $j<8; $j++){
        if ($ss[$j] =~ /chr\w+:(\d+)_([+-])/){
            $coord = $1;
            $strand = $2;
            $ss_coord[$j][0] = $coord;
            $ss_coord[$j][1] = $strand;
        }
        else {
            $ss_coord[$j][0] = undef;
            $ss_coord[$j][1] = undef;
        }
    }
```

```
        #sense, positive strand
        if ((abs($ss_coord[2][0]-$tailcoord1)<=$gap)&&($ss_coord[2][1] eq $readstrand)&&($readstrand
eq "+")&&($ss_coord[2][1] ne undef)){
            $dist5ss = $tailcoord1-$ss_coord[2][0];
            if ($dist5ss >= 0){
                $tailcoord1 = $ss_coord[2][0];
                $headcoord2 = $headcoord2+($gap-$dist5ss);
                print FHhead $chrom,"\t",$headcoord1,"\t",$headcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
                print FHtail $chrom,"\t",$tailcoord1,"\t",$tailcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
            }

        }
        #sense, negative strand
        elsif ((abs($ss_coord[3][0]-$tailcoord2)<=$gap)&&($ss_coord[3][1] eq
$readstrand)&&($readstrand eq "-")&&($ss_coord[3][1] ne undef)){
            $dist5ss = $ss_coord[3][0] - $tailcoord2;
            if ($dist5ss >= 0){
                $tailcoord2 = $ss_coord[3][0];
                $headcoord1 = $headcoord1 - ($gap-$dist5ss);
                print FHhead $chrom,"\t",$headcoord1,"\t",$headcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
                print FHtail $chrom,"\t",$tailcoord1,"\t",$tailcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
            }
        }
        #antisense, positive strand
        elsif ((abs($ss_coord[0][0]-$headcoord1)<=$gap)&&($ss_coord[0][1] ne
$readstrand)&&($readstrand eq "-")&&($ss_coord[0][1] ne undef)){
            $dist5ss = $headcoord1-$ss_coord[0][0];
            if ($dist5ss >= 0){
                $headcoord1 = $ss_coord[0][0];
                $tailcoord2 = $tailcoord2+($gap-$dist5ss);
                print FHhead $chrom,"\t",$headcoord1,"\t",$headcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
                print FHtail $chrom,"\t",$tailcoord1,"\t",$tailcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
            }
        }
        #antisense, negative strand
        elsif ((abs($ss_coord[1][0]-$headcoord2)<=$gap)&&($ss_coord[1][1] ne
$readstrand)&&($readstrand eq "+")&&($ss_coord[1][1] ne undef)){
            $dist5ss = $ss_coord[1][0] - $headcoord2;
            if ($dist5ss >= 0){
                $headcoord2 = $ss_coord[1][0];
                $tailcoord1 = $tailcoord1 - ($gap-$dist5ss);
                print FHhead $chrom,"\t",$headcoord1,"\t",$headcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
                print FHtail $chrom,"\t",$tailcoord1,"\t",$tailcoord2,"\t",$id,"\t0\t",$readstrand,"\n";
            }
        }
    }
    close FH; close FHhead; close FHtail;

    $command1 = "bedtools getfasta -fi ".$self->{_index}.".fa -bed ".$fhhead." -fo ".$self-
>{_outdir}."/".$base."_heads.fasta -name -tab -s";
    $command2 = "bedtools getfasta -fi ".$self->{_index}.".fa -bed ".$fhtail." -fo ".$self-
>{_outdir}."/".$base."_tails.fasta -name -tab -s";

    print "\n\n$command1\n\n$command2\n\n";

    system($command1);
    system($command2);

    my %gappedreads;
    $fhhead =~ s/bed/fasta/;
    $fhtail =~ s/bed/fasta/;

    print "opening:\t$fhhead\n";
    print "opening:\t$fhtail\n";

    open (FHhead, $fhhead) or die $!;
    open (FHtail, $fhtail) or die $!;
    while (<FHhead>){
        ($id,$seq) = (split /\t/,$_);
        chomp $seq;
        $gappedreads{$id}[0] = $seq;
    }
    while (<FHtail>){
        ($id,$seq) = (split /\t/,$_);
        chomp $seq;
        $gappedreads{$id}[1] = $seq;
    }
    close FHhead; close FHtail;

    open (FH, $fh) or die $!;
    $fhout = $self->{_outdir}."/".$base."_truelariats.txt";
    open (FHout, ">$fhout") or die $!;

    while (<FH>){
        $line = $_;
        chomp $line;

        $id = (split /\t/,$line)[0];
        $seq = (split /\t/,$line)[1];
        $headseq = (split /\t/,$line)[6];
        $tailseq = (split /\t/,$line)[14];
        $headcoord1 = (split /\t/,$_)[5];
        $headcoord2 = length($headseq)+$headcoord1;
        $tailcoord1 = (split /\t/,$line)[13];
        $tailcoord2 = length($tailseq)+$tailcoord1;


        ####print output file
        if (exists $gappedreads{$id}[0]){
            $genomic = $gappedreads{$id}[0].$gappedreads{$id}[1];
            $stringdist = &levenshtein($seq,$genomic);
            if ($stringdist<=3){
                print FHout $line,"\t",$genomic,"\t",$stringdist,"\n";
            }
        }
    }

}




sub levenshtein
{
    my ($s1, $s2) = @_;
    my ($len1, $len2) = (length $s1, length $s2);

    return $len2 if ($len1 == 0);
    return $len1 if ($len2 == 0);

    my %mat;

    for (my $i = 0; $i <= $len1; ++$i)
    {
        for (my $j = 0; $j <= $len2; ++$j)
        {
            $mat{$i}{$j} = 0;
            $mat{0}{$j} = $j;
        }

        $mat{$i}{0} = $i;
    }

    my @ar1 = split(//, $s1);
    my @ar2 = split(//, $s2);

    for (my $i = 1; $i <= $len1; ++$i)
    {
        for (my $j = 1; $j <= $len2; ++$j)
        {
            my $cost = ($ar1[$i-1] eq $ar2[$j-1]) ? 0 : 1;

            $mat{$i}{$j} = min([$mat{$i-1}{$j} + 1,
                                $mat{$i}{$j-1} + 1,
                                $mat{$i-1}{$j-1} + $cost]);
        }
    }

    return $mat{$len1}{$len2};
}


sub min
{
    my @list = @{$_[0]};
    my $min = $list[0];

    foreach my $i (@list)
    {
        $min = $i if ($i < $min);
    }

    return $min;
}


1;
```

2- U2-premRNA alignment motif discovery script:

Run iteratively, after removing sequences that map to discovered motif from each iteration.

```
#Bootstrap_U2snRNA_Aligner.pl
#! /usr/bin/perl

$| = 1;

$trial = 1000;

#bash-3.2$  Bootstrap_u2snRNA_Aligner.pl  Foreground(INPUT) Background(INPUT)    # of trials
run_num
if ($ARGV[2])
{
   $trial = $ARGV[2];
}

$run_num = $ARGV[3];
my $file = join "_",@ARGV ;
$file =~ tr/\//_/;
my $name = "Bootstrap_run".$file."_output";
open (OUT, ">$name");

my $out1 = "Bootstrap_run".$file."_summary";
open (OUT1, ">$out1");
my $out2 =  "Bootstrap_run".$file."_Enrichment_Per_run.distribution";
open (OUT2,">$out2");



# count how many lines of input foreground data lead to how many alignments. How many lines of
background input lead to how many alignments. these will be different #s
foreach my $a (0..1)
{
   open (COM,"wc -l $ARGV[$a]|");
   while (<COM>)
   {
       $_ =~ m/(\d*)\s*$ARGV[$a]/;
       $filelength[$a] = $1;
   }
}

# loads foreground
open (DATA,"$ARGV[0]");
@fore = (<DATA>);
```

```
#loads background
open (BACK, "$ARGV[1]");
@back = (<BACK>);


#START TRIAL:
foreach my $t (1..$trial)
{
   print "run\t",$run_num,"\ttrial\t",$t,"\n";
   # create two hash tables for each trial which track the number of each type of alignment from
foreground and background
   %back = 0;
   %fore =0;



   # FOREGROUND IS SAMPLED
   foreach my $sampled (1 ..$filelength[0])
   {
       $sample = $fore[int(rand($filelength[0]))];
       chomp $sample;
       $input = uc $sample;

       %local =""; # this hash ensures can't report same alignment twice

       foreach my $b (1..3)
       {
          foreach my $a (1..(19 -$b))
          {
              $input =~ m/(^\w{$a})(\w{$b})(\w*)/;
              my $string = $1.$3;
              my $align = $1.(lc $2).$3;


                     foreach my $length (0..(14 -$b))
                     {
                            if ($string =~ m/^\w{$length}T[G|A][C|T]T[G|A][C|T]/)
                            {

                                   my $tag ="";
                                   if ($a <= $length)
                                   {
                                          $length = $length + $b;
                                          $tag = "_._.$length";
                                   }
```

```
                                        elsif ($a >= ($length + 6))
                                        {
                                                $tag = "_._.$length";
                                        }
                                        else
                                        {
                                                $tag = "$a.$b.$length";
                                        }

                                        unless($local{$tag}) #this is so I don't overcount an alignment that falls
outside the loop position(s)
                                        {
                                                $example =  "$ARGV[0]\t$sample\t$tag\n";
                                                unless($global{$example}) #this is so I don't reprint alignments as
they are found in each sample over and over again
                                                {
                                                        print OUT "$ARGV[0]\t$sample\t$tag\n";
                                                        $global{$example}++;
                                                }
                                                $fore{$tag} ++;
                                                $forever{$tag} ++;
                                                $keychain{$tag} ++;
                                                $local{$tag} ++;
                                        }

                                }
                        }
                }
        }
   }
   # BACKGROUND IS SAMPLED
   foreach my $sampled (1..$filelength[0]) #samples equivalent number of background query sequences
   {
      $sample = $back[int(rand($filelength[1]))];
      chomp $sample;
      $input = uc $sample;

      %local ="";

      foreach my $b (1..3)
      {
         foreach my $a (1..(19 -$b))
         {
            $input =~ m/(^\w{$a})(\w{$b})(\w*)/;
            my $string = $1.$3;
            my $align = $1.(lc $2).$3;
            foreach my $length (0..(14 -$b))
            {
            if ($string =~ m/^\w{$length}T[G|A][C|T]T[G|A][C|T]/)
            {

               my $tag ="";
               if ($a <= $length)
               {
                   $length = $length + $b;
                   $tag = "_._.$length";
               }
               elsif ($a >= ($length + 6))
               {
                   $tag = "_._.$length";
               }
               else
               {
                   $tag = "$a.$b.$length";
               }
               unless($local{$tag})
               {
                   $example =  "$ARGV[1]\t$sample\t$tag\n";
                   unless($global{$example})
                   {
                      print OUT "$ARGV[1]\t$sample\t$tag\n";
                      $global{$example}++;
                   }
                   $back{$tag} ++;
                   $backever{$tag} ++;
                   $keychain{$tag} ++;
                   $local{$tag} ++;
               }
            }
            }
        }
    }
}

# COMPARE foreground to background
my @compare = keys %keychain;
foreach my $key (@compare)
{
    if ($back{$key} >= $fore{$key})
    {
       $exceed{$key}++;
    }
```

```
        $enrich[$t]{$key} = $fore{$key}/($back{$key}+1);
    }
    %back = 0;
 # end of bootstrap, repeat $trial number of times
}





#SUMMARY
my @compare = keys %keychain;
foreach my $key (@compare)
{

    my $pval = (int (100000*$exceed{$key}/$trial))/100000;
    unless ($forever{$key})
    {
        $forever{$key} = 0;
    }
    print OUT1 "$key\t$forever{$key}\t$backever{$key}\t$exceed{$key}\t$trial\n";
    #print OUT1 "$key\t$forever{$key}\t$backever{$key}\t". ( (int(100*
($forever{$key}/($backever{$key}+1))))/100)  ."\t$pval\n";

print OUT2 "$key\t";
foreach my $i (1..$trial)
{
print OUT2          "$enrich[$i]{$key}\t";

}

print OUT2 "\n";


    $y++;
}


print OUT1 " there were $y different types of alignments\n";
```

3- U2-premRNA alignment motif matching pipeline:

1.) write_bed.pl
2.) sort bed files | uniq (all three bed files)
3.) bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -bed bp_ds_TS_window_coords_uniq.bed -fo bp_ds_TS_window_coords_uniq_seq.txt -name -tab -s (all three files: bp_ds_TS_window_coords, bp_window_coords, fivepr_window_coords)
4.) make_sequences.pl (format fasta files for patser)
5.) make pwm_list.txt and seq_list.txt for all patser scoring runs
6.) run_patser.pl
7.) parse_patser.pl *_scores.txt > all_scores.txt
8.) perl take_most_sig_motif.pl
9.) mkdir TS. Copy bp_ds_TS_window_seq & fivepr_window_coords_seq. mkdir TS/fivepr_matrices
10.) make_matrices.pl
11.) score_ts.pl

```
#write_bed.pl

use warnings;

$numargs = $#ARGV;
#open (FHout, ">lariats.bed") or die $!;
for (my $j = 0; $j<=$numargs;$j++){

        $file = $ARGV[$j];
        $file =~ /(SRR\d+)/;
        $srr = $1; print $srr,"\n";
        open (FH, $file) or die $!;

        while (<FH>){
                $chrom = (split /\t/,$_)[4];
                $strand = (split /\t/,$_)[5];
                $fivepr = (split /\t/,$_)[6];
                $bp = (split /\t/,$_)[8];
                if ($strand eq "+"){
                        print  $chrom,"\t",$fivepr,"\t",$bp,"\t",$srr,"\t0\t",$strand,"\n";
                }
                elsif ($strand eq "-"){
                        print  $chrom,"\t",$bp,"\t",$fivepr,"\t",$srr,"\t0\t",$strand,"\n";
                }
        }
        close FH;
}


#make_sequences.pl

use warnings;

open (FH, $ARGV[0]) or die $!;
$ARGV[0] =~ /(.+).txt/;
$fhout = $1."_patser.txt";
open (FHout, ">$fhout") or die $!;

while (<FH>){
        ($id,$seq)=(split /\t/,$_);
        chomp $seq;
        $seq =~ tr/[a-z]/[A-Z]/;
        print FHout $id,"\t\\",$seq,"\\\n";
}

close FH;
```

```
#run_patser.pl

use warnings;

open (FH, "pwm_list.txt") or die $!;
while (<FH>){
	$pwm = $_; chomp $pwm;
	open (FH2, "seq_list.txt") or die $!;
	while (<FH2>){
		$seq = $_; chomp $seq;
		$seq =~ /(.+)_patser/;
		$id1 = $1;
		$pwm =~ /files.(.+)_pwm/;
		$id2 = $1;
		$outfile = $id1."_".$id2."_scores.txt";
		$cmd = "patser -m ".$pwm." -f ".$seq." -A a:t 1 c:g 1 -s -t -M -100  > ".$outfile;
		print $cmd,"\n";
		system($cmd);
	}
	close FH2;
}
```

```
#parse_patser.pl

use warnings;

my %data;

$numargs = $#ARGV;

for (my $j = 0; $j<=$numargs;$j++){
        $ARGV[$j] =~ /uniq_(.+)_scores.txt/;
        $motif = $1;
        $pos_low = 0; $pos_high = 0;
        if (($motif eq "canonical")||($motif eq "canonicalCbp")){
                $pos_low = 5; $pos_high = 7;
        }
        elsif ($motif eq "2ntbulge"){
                $pos_low = 4; $pos_high = 7;
        }
        elsif ($motif eq "firstTRY_1bulge"){
                $pos_low = 8; $pos_high = 10;
        }
        elsif ($motif eq "firstTRY_2bulge"){
                $pos_low = 7; $pos_high = 10;
        }
        elsif ($motif eq "firstTRY_3bulge"){
                $pos_low = 6; $pos_high = 10;
        }
        #print $category,"\t",$motif,"\n";
        open (FH,$ARGV[$j]) or die $!;
        while (<FH>){
                if ($_ =~ /position/){
                        if ($_ =~ /^(.+)\s+position\=\s+(\d+).+score=\s+(-?\d+\.?\d*)\s+.*value\)\=\s+(-
?\d+\.?\d*)\s+sequence=\s+([ACGT]+)/){
                                $locus = $1;
                                $position = $2;
                                $score = $3;$lnp = $4;
                                $seq = $5;
                                if (($position >=$pos_low)&&($position <=$pos_high)){
                                        if ((exists $data{$locus}{$motif})&&($data{$locus}{$motif}=~ /\d+/)){
                                                $score2 = (split /\t/,$data{$locus}{$motif})[2];
                                                if ($score > $score2){
                                                        $data{$locus}{$motif} =
$position."\t".$seq."\t".$score."\t".$lnp;
                                                }
                                        }
                                        else {
```

```
                                        $data{$locus}{$motif}=$position."\t".$seq."\t".$score."\t".$lnp;
                                    }
                                }
                                else {$data{$locus}{$motif}="\t\t\t";}
                            }
                            else {
                                print "error: $_";
                            }
                    }
            }
            close FH;
}

print
"loci\tcanonical\t\t\t\tcanonicalCbp\t\t\t\t2ntbulge\t\t\t\tfirstTRY_1bulge\t\t\t\tfirstTRY_2bulge\t\t\t\tfirst
TRY_3bulge\n";

foreach my $l (keys %data){
        print $l,"\t";
        if (exists $data{$l}{"canonical"}){print $data{$l}{"canonical"},"\t";}
        else {print "\t\t\t\t";}
        if (exists $data{$l}{"canonicalCbp"}){print $data{$l}{"canonicalCbp"},"\t";}
        else {print "\t\t\t\t";}
        if (exists $data{$l}{"2ntbulge"}){print $data{$l}{"2ntbulge"},"\t";}
        else {print "\t\t\t\t";}
        if (exists $data{$l}{"firstTRY_1bulge"}){print $data{$l}{"firstTRY_1bulge"},"\t";}
        else {print "\t\t\t\t";}
        if (exists $data{$l}{"firstTRY_2bulge"}){print $data{$l}{"firstTRY_2bulge"},"\t";}
        else {print "\t\t\t\t";}
        if (exists $data{$l}{"firstTRY_3bulge"}){print $data{$l}{"firstTRY_3bulge"},"\n";}
        else {print "\t\t\t\t\n";}

}
```

```
#take_most_sig_motif.pl

use warnings;
open (FH, "all_scores.txt") or die $!;

while (<FH>){
        if ($_ =~ /chr\w+/){
                $loci = (split /\t/,$_)[0];

                $canonical_startpos = (split /\t/,$_)[1];
                $canonical_seq = (split /\t/,$_)[2];
                $canonical_lnp = (split /\t/,$_)[4];
                unless ($canonical_lnp =~ /\d/){$canonical_lnp = 10;}

                $canonicalC_startpos = (split /\t/,$_)[5];
                $canonicalC_seq = (split /\t/,$_)[6];
                $canonicalC_lnp = (split /\t/,$_)[8];
                unless ($canonicalC_lnp =~ /\d/){$canonicalC_lnp = 10;}

                $canonical2nt_startpos = (split /\t/,$_)[9];
                $canonical2nt_seq = (split /\t/,$_)[10];
                $canonical2nt_lnp = (split /\t/,$_)[12];
                unless ($canonical2nt_lnp =~ /\d/){$canonical2nt_lnp = 10;}

                $TRAYTRY_startpos = (split /\t/,$_)[13];
                $TRAYTRY_seq = (split /\t/,$_)[14];
                $TRAYTRY_lnp = (split /\t/,$_)[16];
                unless ($TRAYTRY_lnp =~ /\d/){$TRAYTRY_lnp = 10;}

                $TRANYTRY_startpos = (split /\t/,$_)[17];
                $TRANYTRY_seq = (split /\t/,$_)[18];
                $TRANYTRY_lnp = (split /\t/,$_)[20];
                unless ($TRANYTRY_lnp =~ /\d/){$TRANYTRY_lnp = 10;}

                $TRANNYTRY_startpos = (split /\t/,$_)[21];
                $TRANNYTRY_seq = (split /\t/,$_)[22];
                $TRANNYTRY_lnp = (split /\t/,$_)[24]; chomp $TRANNYTRY_lnp;
                unless ($TRANNYTRY_lnp =~ /\d/){$TRANNYTRY_lnp = 10;}

                $min_lnp = -3.912; #p-val < .02

                print $loci;
                if (($canonical_lnp < $min_lnp)||($canonicalC_lnp < $min_lnp)||($canonical2nt_lnp < 
$min_lnp)||($TRAYTRY_lnp < $min_lnp)||($TRANYTRY_lnp < $min_lnp)||($TRANNYTRY_lnp < 
$min_lnp)){
                        my $flag = 0;
```

```
			if (($canonical_lnp <= $canonicalC_lnp)&&($canonical_lnp <=
$canonical2nt_lnp)&&($canonical_lnp <= $TRAYTRY_lnp)&&($canonical_lnp <=
$TRANYTRY_lnp)&&($canonical_lnp <= $TRANNYTRY_lnp)){
				print
"\tcanonical\t",$canonical_startpos,"\t",$canonical_seq,"\t",$canonical_lnp,"\n"; $flag = 1;
			}
			elsif (($canonicalC_lnp <= $canonical_lnp)&&($canonicalC_lnp <=
$canonical2nt_lnp)&&($canonicalC_lnp <= $TRAYTRY_lnp)&&($canonicalC_lnp <=
$TRANYTRY_lnp)&&($canonicalC_lnp <= $TRANNYTRY_lnp)){
				print
"\tcanonicalC\t",$canonicalC_startpos,"\t",$canonicalC_seq,"\t",$canonicalC_lnp,"\n"; $flag = 1;
			}
			elsif (($canonical2nt_lnp <= $canonicalC_lnp)&&($canonical2nt_lnp <=
$canonical_lnp)&&($canonical2nt_lnp <= $TRAYTRY_lnp)&&($canonical2nt_lnp <=
$TRANYTRY_lnp)&&($canonical2nt_lnp <= $TRANNYTRY_lnp)){
				print
"\tcanonical2nt\t",$canonical2nt_startpos,"\t",$canonical2nt_seq,"\t",$canonical2nt_lnp,"\n"; $flag = 1;
			}
			elsif (($TRAYTRY_lnp <= $canonicalC_lnp)&&($TRAYTRY_lnp <=
$canonical2nt_lnp)&&($TRAYTRY_lnp <= $canonical_lnp)&&($TRAYTRY_lnp <=
$TRANYTRY_lnp)&&($TRAYTRY_lnp <= $TRANNYTRY_lnp)){
				print
"\tTRAYTRY\t",$TRAYTRY_startpos,"\t",$TRAYTRY_seq,"\t",$TRAYTRY_lnp,"\n"; $flag = 1;
			}
			elsif (($TRANYTRY_lnp <= $canonicalC_lnp)&&($TRANYTRY_lnp <=
$canonical2nt_lnp)&&($TRANYTRY_lnp <= $TRAYTRY_lnp)&&($TRANYTRY_lnp <=
$canonical_lnp)&&($TRANYTRY_lnp <= $TRANNYTRY_lnp)){
				print
"\tTRANYTRY\t",$TRANYTRY_startpos,"\t",$TRANYTRY_seq,"\t",$TRANYTRY_lnp,"\n"; $flag =
1;
			}
			elsif (($TRANNYTRY_lnp <= $canonicalC_lnp)&&($TRANNYTRY_lnp <=
$canonical2nt_lnp)&&($TRANNYTRY_lnp <= $TRAYTRY_lnp)&&($TRANNYTRY_lnp <=
$TRANYTRY_lnp)&&($TRANNYTRY_lnp <= $canonical_lnp)){
				print
"\tTRANNYTRY\t",$TRANNYTRY_startpos,"\t",$TRANNYTRY_seq,"\t",$TRANNYTRY_lnp,"\n";
$flag = 1;
			}
			else {print "error: $loci \n";}
			if ($flag==0){print "\tnone\n";}
		}
		else {print "\tnone\n";}

	}
}
```

```
#make_matrices.pl

use warnings;
open (FH, "fivepr_window_coords_seq_uniq.txt") or die $!;
while (<FH>){
        ($id,$seq)=(split /\t/,$_); chomp $seq;
        $seq =~ tr/[a-z]/[A-Z]/;
        $fhout = "./fivepr_matrices/".$id.".txt";
        open (FHout,">$fhout") or die $!;
        my @s;
        $seq =~ /(\w)(\w)(\w)(\w)(\w)(\w)/;
        $s[0] = $1; $s[1] = $2; $s[2] = $3;
        $s[3] = $4; $s[4] = $5; $s[5] = $6;
        #A
        print FHout "A";
        for (my $j = 0;$j<6;$j++){
                if ($s[$j] eq "A"){print FHout "\t1";}
                else {print FHout "\t0";}
        }
        print FHout "\n";
        #C
        print FHout "C";
        for (my $j = 0;$j<6;$j++){
                if ($s[$j] eq "C"){print FHout "\t1";}
                else {print FHout "\t0";}
        }
        print FHout "\n";
        #G
        print FHout "G";
        for (my $j = 0;$j<6;$j++){
                if ($s[$j] eq "G"){print FHout "\t1";}
                else {print FHout "\t0";}
        }
        print FHout "\n";
        #T
        print FHout "T";
        for (my $j = 0;$j<6;$j++){
                if ($s[$j] eq "T"){print FHout "\t1";}
                else {print FHout "\t0";}
        }
        print FHout "\n";
        print FHout "N";
        for (my $j = 0;$j<6;$j++){
                if ($s[$j] eq "N"){print FHout "\t1";}
                else {print FHout "\t0";}
```

```
	}
	print FHout "\n";
	close FHout;
}


#score_ts.pl

use warnings;

open (FH,"bp_ds_TS_window_coords_seq_uniq.txt") or die $!;
open (FHscores,">TS_scores_all.txt") or die $!;
while (<FH>){
	($id,$seq)=(split /\t/,$_); chomp $seq; $seq =~ tr/[a-z]/[A-Z]/;
	$matrix = "./fivepr_matrices/".$id.".txt";
	open (FHout,">temp_seq.txt") or die $!;
	print FHout "seq \\",$seq,"\\";
	close FHout;
	$cmd = "patser -m ".$matrix." -A a 2612 c 2268 t 2577 g 2543 n 0.03 -s -M -1000 <
temp_seq.txt";
	$output = `$cmd`;
	 if ($output =~ /score\=\s+(\-?\d+[[:punct:]]?\d*).+value\)\=\s+(\-
?\d+\.\d*)\s+sequence\=\s*([ACGTNacgtn]+)/){
	print FHscores $id,"\t",$1,"\t",$2,"\t",$3,"\n";
	}
	else {
	print "error: $id,\n";
	print $output,"\n\n\n\n";
	}
}
```

Branchpoint Motif Matrices- derived from discovered motifs, formatted for patser scoring:

canonical_pwm.txt
```
A	0	195	1	1	131	562	0
C	0	0	575	0	141	0	399
G	0	492	0	0	341	21	0
T	774	0	199	774	136	0	375
```

canonicalCbp_pwm.txt
```
A	0	30	0	0	25	0	3
C	0	0	95	0	0	107	107
G	0	85	0	0	90	0	1
T	115	0	20	115	0	8	4
```

2ntbulge_pwm.txt
```
A	0	34	0	0	15	72	42	0
C	0	0	86	0	25	2	0	46
G	0	65	0	0	53	2	57	0
T	99	0	13	99	6	23	0	53
```

firstTRY_1bulge_pwm.txt
```
A	0	65	128	0	0	54	0
C	0	0	27	117	0	0	52
G	0	113	2	0	0	124	0
T	178	0	21	61	178	0	126
```

firstTRY_2bulge_pwm.txt
```
A	0	51	104	24	0	0	42	0
C	0	0	18	51	80	0	0	42
G	0	92	5	31	0	0	101	0
T	143	0	16	37	63	143	0	101
```

firstTRY_3bulge_pwm.txt
```
A	0	70	160	31	43	0	0	63	0
C	0	0	23	83	49	83	0	0	64
G	0	131	5	25	34	0	0	138	0
T	201	0	13	62	75	118	201	0	137
```

4- Formatting lariat excel tables:

1.) copy TS_scores_all.txt and most_sig_motif.txt to lariat_table directory. cp hg19_introns or mm9_introns to lariat_table_directory.
2.) cat SR*/lariat_data_table.txt > StudyID_lariat_data_table.txt
3.) ~/read_seq/map_mutation.pl Study_lariat_data_table.txt
4.) make_bed_files.pl lariat_data_table_wReadSeq.txt
5.) cut -f5-9 *_lariat_data_table_wReadSeq.txt | sort | uniq > all_coords_uniq.txt
6.) perl ./../make_circle_bed.pl all_coords_uniq.txt > circle.bed
7.) perl ./../make_TS.bed > TS.bed
8.) perl get_human_table1.pl > BPs_apparent_real_wFoundFiles_final.txt
9.) perl make_table_1.pl > BP_table_1_final.txt
10.) perl make_BP_table2.pl > BP_table_2_final.txt
11.) perl assign_3ss_and_category.pl BP_table_2_final.txt > BP_table_3_final.txt
12.) perl make_table_4.pl > BP_table_4_final.txt
13.) perl make_table_5.pl

```
#map_mutation.pl

use warnings;

$ARGV[0] =~ /(.+).txt/;
$fileout = $1."_wReadSeq.txt";
$base = $1;

open (FHout,">$fileout") or die $!;
open (FH, "$ARGV[0]") or die $!;

while (<FH>){
	$line = $_; chomp $line;
	$seq = (split /\t/,$_)[3];
	$chr = (split /\t/,$_)[4];
	$strand = (split /\t/,$_)[5];
	$fivepr = (split /\t/,$_)[6];
	$bp = (split /\t/,$_)[8];
	$bpseq = (split /\t/,$_)[9]; chomp $bpseq;
	$refstart = ""; $refend = "";
	$genomic = "";
	if ($strand eq "+"){

		$refstart = $bp-100;
		$refend = $fivepr+100;
		$fhbed = $base."_temp.bed";
		open (FHbed,">$fhbed") or die $!;
		print FHbed $chr,"\t",$refstart,"\t",$bp,"\thead\t0\t+\n";
		print FHbed $chr,"\t",$fivepr,"\t",$refend,"\ttail\t0\t+\n";
		close FHbed;

		$cmd = "bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -tab -s -name -bed
".$fhbed." -fo ".$base."_tempseq.txt";
		system($cmd);
		$fhseq = $base."_tempseq.txt";
		open (FHseq,"$fhseq") or die $!;
		while (<FHseq>){
			if ($_ =~ /head/){
				$headseq = (split /\t/,$_)[1]; chomp $headseq;
			}
			if ($_ =~ /tail/){
				$tailseq = (split /\t/,$_)[1]; chomp $tailseq;
			}
		}
		close FHseq;
		$cmd = "rm ".$fhbed." ".$fhseq;
```

```
        system($cmd);

        $genomic = $headseq.$tailseq;

    }

    if ($strand eq "-"){

        $refstart = $bp+100;
        $refend = $fivepr-100;
        $fhbed = $base."_temp.bed";
        open (FHbed,">$fhbed") or die $!;
        print FHbed $chr,"\t",$bp,"\t",$refstart,"\thead\t0\t-\n";
        print FHbed $chr,"\t",$refend,"\t",$fivepr,"\ttail\t0\t-\n";
        close FHbed;

        $cmd = "bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -tab -s -name -bed
".$fhbed." -fo ".$base."_tempseq.txt";
        system($cmd);
        $fhseq = $base."_tempseq.txt";
        open (FHseq,"$fhseq") or die $!;
        while (<FHseq>){
            if ($_ =~ /head/){
                $headseq = (split /\t/,$_)[1]; chomp $headseq;
            }
            if ($_ =~ /tail/){
                $tailseq = (split /\t/,$_)[1]; chomp $tailseq;
            }
        }
        close FHseq;
        $cmd = "rm ".$fhbed." ".$fhseq;
        system($cmd);

        $genomic = $headseq.$tailseq;
    }

    $seq =~ tr/[a-z]/[A-Z]/;
    $genomic =~ tr/[a-z]/[A-Z]/;

    $seq_forward = $seq;


    $len = length($seq_forward);
    $flag = 0;
    my %levdist; $mini = 100;
    for (my $j = 0; $j<=(200-$len);$j++){
        $gen_substr = substr($genomic,$j,$len);
        $stringdist = &levenshtein($seq_forward,$gen_substr);
        if ($stringdist <=3){
            $levdist{$stringdist}=$j;
            $flag = 1;
            if ($stringdist<$mini){$mini = $stringdist;}
        }
    }
    if ($flag==1){
        $j = $levdist{$mini};
        $i = 100-$j-5;
        $seqsubstr = substr($seq_forward,$i,10);
        print FHout $line,"\t",$seq_forward,"\t",$seqsubstr,"\t",$bpseq,"\n";
    }

    if ($flag==0){
        $seq_forward = reverse $seq; $seq_forward =~ tr/ACGT/TGCA/;
        for (my $j = 0; $j<=(200-$len);$j++){
            $gen_substr = substr($genomic,$j,$len);
            $stringdist = &levenshtein($seq_forward,$gen_substr);
            if ($stringdist <=3){
                $levdist{$stringdist}=$j;
                if ($stringdist<$mini){$mini=$stringdist;}
                $flag = 1;
            }
        }
        if ($flag==1){
            $j = $levdist{$mini};
            $i = 100-$j-5;
            $seqsubstr = substr($seq_forward,$i,10);
            print FHout $line,"\t",$seq_forward,"\t",$seqsubstr,"\t",$bpseq,"\n";
        }
    }

    if ($flag==0){
        print FHout "error\t: $line\n";
    }


}

sub levenshtein
{
  my ($s1, $s2) = @_;
  my ($len1, $len2) = (length $s1, length $s2);
  return $len2 if ($len1 == 0);
  return $len1 if ($len2 == 0);

  my %mat;

  for (my $i = 0; $i <= $len1; ++$i)
  {
      for (my $j = 0; $j <= $len2; ++$j)
      {
          $mat{$i}{$j} = 0;
          $mat{0}{$j} = $j;
      }

      $mat{$i}{0} = $i;
  }

  my @ar1 = split(//, $s1);
  my @ar2 = split(//, $s2);

  for (my $i = 1; $i <= $len1; ++$i)
  {
      for (my $j = 1; $j <= $len2; ++$j)
      {
          my $cost = ($ar1[$i-1] eq $ar2[$j-1]) ? 0 : 1;

          $mat{$i}{$j} = min([$mat{$i-1}{$j} + 1,
                              $mat{$i}{$j-1} + 1,
                              $mat{$i-1}{$j-1} + $cost]);
      }
  }

  return $mat{$len1}{$len2};
}

sub min
{
  my @list = @{$_[0]};
  my $min = $list[0];

  foreach my $i (@list)
  {
      $min = $i if ($i < $min);
  }

  return $min;
}
```

```
#make_bed_files.pl

#output:
###bed file for all reads
###bed file for uniq reads
###bed file for mismatch at BP all reads
###bed file for mismatch at BP uniq reads

use warnings;

my %read;
my %read_BPmismatch;

open (FHallreads,">all_reads.bed") or die $!;
open (FHallreadsmut,">all_reads_BPmismatch.bed") or die $!;
#read in lariat_data_table_wReadseq.txt
open (FH, "$ARGV[0]") or die $!;
while (<FH>){
        $readid = (split /\t/,$_)[2];
        $readid =~ /(SRR\d+)/;
        $srr = $1;
        $read = (split /\t/,$_)[3];
        $chrom = (split /\t/,$_)[4];
        $strand = (split /\t/,$_)[5];
        $fivepr = (split /\t/,$_)[6];
        $bp = (split /\t/,$_)[8];
        $readseq = (split /\t/,$_)[11]; $genseq = (split /\t/,$_)[12]; chomp $genseq;
        $readseq =~ /^....(.)/;
        $readBP = $1;
        $genseq =~ /^....(.)/;
        $genBP = $1;
        $mut = 0;
        if ($readBP ne $genBP){$mut = 1;}
        $bedline = "";
        if ($strand eq "+"){
                $bedline = $chrom."\t".$fivepr."\t".$bp."\t".$srr."\t0\t+\n";
        }
        elsif ($strand eq "-"){
                $bedline = $chrom."\t".$bp."\t".$fivepr."\t".$srr."\t0\t-\n";
        }
        print FHallreads $bedline;
        if ($mut ==1){
                print FHallreadsmut $bedline;
        }
        $id = $srr."\t".$read;
        $read{$id}=$bedline;
        if ($mut==1){
                $read_BPmismatch{$id}=$bedline;
        }
}
close FH;


open (FHuniqreads,">uniq_reads.bed") or die $!;
open (FHuniqreadsmut,">uniq_reads_BPmismatch.bed") or die $!;

foreach my $r (keys %read){
        print FHuniqreads $read{$r};
}
foreach my $r (keys %read_BPmismatch){
        print FHuniqreadsmut $read_BPmismatch{$r};
}
```

```
#make_circle_bed.pl

use warnings;
#read in uniq coords, chrom strand fivepr threepr bp
open (FH, $ARGV[0]) or die $!;

my %circle;

while (<FH>){
        ($chrom,$strand,$fivepr,$threepr,$bp)=(split /\t/,$_); chomp $bp;
        if ($threepr =~ /\d/){
        if ($threepr ==$bp){
                if ($strand eq "+"){
                        print $chrom,"\t",$fivepr,"\t",$threepr,"\tcircle\t0\t+\n";
                }
                elsif ($strand eq "-"){
                        print $chrom,"\t",$threepr,"\t",$fivepr,"\tcircle\t0\t-\n";
                }
        }}
}
close FH;


#make_TS_bed.pl

use warnings;

open (FH, "TS_scores_all.txt") or die $!;

while (<FH>){
        ($event,$score,$lnp,$seq)=(split /\t/,$_);
        chomp $seq;
        $event =~ /(chr\w+)\:(\d+)_(\d+)_([+-])/;
        $chrom = $1; $fivepr = $2; $threepr = $3; $strand = $4;
        $min_lnp = -3.912; #pval<.02
        if ($lnp<=$min_lnp){
                if ($strand eq "+"){
                        print $chrom,"\t",$fivepr,"\t",$threepr,"\tTS\t0\t+\n";
                }
                elsif ($strand eq "-"){
                        print $chrom,"\t",$threepr,"\t",$fivepr,"\tTS\t0\t-\n";
                }
        }
}
close FH;
```

```
#get_human_table1.pl

use warnings;

my %lariats;
my %lariat_file;

my %lariat_file_uniq;
open (FH, "all_reads.bed") or die $!;
while (<FH>){
        $file = "";
        if ($_ =~ /(Mattick\(SRR\d+\))/){
                $file = $1;
        }
        elsif ($_ =~ /(SRR\d+)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_wt)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_K700E)/){
                $file = $1;
        }
        else {print "error: $_\n";}

                ($chrom,$start,$end,$id,$score,$strand)=(split /\t/,$_); chomp $strand;
                $bp = "";
                if ($strand eq "+"){$bp = $end;}
                if ($strand eq "-"){$bp = $start;}
                $idnew = $chrom.":".$bp."_".$strand; $idnew =~ s/\s//g;
                #print $idnew,"\n";
                if (exists $lariat_file{$idnew}{$file}){$lariat_file{$idnew}{$file}++;}
                else {$lariat_file{$idnew}{$file}=1;}
}
close FH;

open (FH, "uniq_reads.bed") or die $!;


while (<FH>){
       $file = "";
        if ($_ =~ /(Mattick\(SRR\d+\))/){
                $file = $1;
        }
        elsif ($_ =~ /(SRR\d+)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_wt)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_K700E)/){
                $file = $1;
        }
        else {print "error: $_\n";}

              ($chrom,$start,$end,$id,$score,$strand)=(split /\t/,$_); chomp $strand;
              $bp = "";
              if ($strand eq "+"){$bp = $end;}
              if ($strand eq "-"){$bp = $start;}
              $idnew = $chrom.":".$bp."_".$strand; $idnew =~ s/\s//g;
              #print $idnew,"\n";
              if (exists $lariat_file_uniq{$idnew}{$file}){$lariat_file_uniq{$idnew}{$file}++;}
              else {$lariat_file_uniq{$idnew}{$file}=1;}
}
close FH;

my %lariat_file_mut;
open (FH, "all_reads_BPmismatch.bed") or die $!;
while (<FH>){
       $file = "";
        if ($_ =~ /(Mattick\(SRR\d+\))/){
                $file = $1;
        }
        elsif ($_ =~ /(SRR\d+)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_wt)/){
                $file = $1;
        }
        elsif ($_ =~ /(fairbrother_K700E)/){
                $file = $1;
        }
        else {print "error: $_\n";}

              ($chrom,$start,$end,$id,$score,$strand)=(split /\t/,$_); chomp $strand;
              $bp = "";
              if ($strand eq "+"){$bp = $end;}
              if ($strand eq "-"){$bp = $start;}
              $idnew = $chrom.":".$bp."_".$strand; $idnew =~ s/\s//g;
              #print $idnew,"\n";
              if (exists $lariat_file_mut{$idnew}{$file}){$lariat_file_mut{$idnew}{$file}++;}
```

```
                else {$lariat_file_mut{$idnew}{$file}=1;}
}
close FH;


my %lariat_file_mut_uniq;
open (FH, "uniq_reads_BPmismatch.bed") or die $!;
while (<FH>){
        $file = "";
         if ($_ =~ /(Mattick\(SRR\d+\))/){
                 $file = $1;
         }
         elsif ($_ =~ /(SRR\d+)/){
                 $file = $1;
         }
         elsif ($_ =~ /(fairbrother_wt)/){
                 $file = $1;
         }
         elsif ($_ =~ /(fairbrother_K700E)/){
                 $file = $1;
         }
         else {print "error: $_\n";}

                ($chrom,$start,$end,$id,$score,$strand)=(split /\t/,$_); chomp $strand;
                $bp = "";
                if ($strand eq "+"){$bp = $end;}
                if ($strand eq "-"){$bp = $start;}
                $idnew = $chrom.":".$bp."_".$strand; $idnew =~ s/\s//g;
                #print $idnew,"\n";
                if (exists $lariat_file_mut_uniq{$idnew}{$file}){$lariat_file_mut_uniq{$idnew}{$file}++;}
                else {$lariat_file_mut_uniq{$idnew}{$file}=1;}
}
close FH;




open (FH,"most_sig_motif.txt") or die $!;
while (<FH>){
         $line = $_; $line =~ s/^\s+//;
         $id = (split /\t/,$line)[0]; $motif = (split /\t/,$line)[1]; chomp $motif;
         $id =~ s/\s//g;
         $id =~ /(\w+)\:(\d+)_([+-])/;
         $chrom = $1; $bp_apparent = $2; $strand = $3;
         $id2 = $chrom.":".$bp_apparent."_".$strand;
         if ($motif eq "none"){
                         $lariats{$id2}{"motif"} = "none";
                         $lariats{$id2}{"bp_real"} = "null";
                         $lariats{$id2}{"bp_seq"} = "null";
         }
         else {
                 $pos = (split /\t/,$line)[2]; $seq = (split /\t/,$_)[3];
                 $lariats{$id2}{"motif"} = $motif;
                 if (($strand eq "+")&&(($motif eq "canonical")||($motif eq "canonicalC"))){
                         $real_bp = ($pos-5)+$bp_apparent;
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "-")&&(($motif eq "canonical")||($motif eq "canonicalC"))){
                         $real_bp = $bp_apparent - ($pos-5);
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "+")&&(($motif eq "canonical2nt"))){
                         $real_bp = ($pos-4)+$bp_apparent;
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "-")&&(($motif eq "canonical2nt"))){
                         $real_bp = $bp_apparent - ($pos-4);
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "+")&&(($motif eq "TRAYTRY"))){
                         $real_bp = ($pos-8)+$bp_apparent;
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "-")&&(($motif eq "TRAYTRY"))){
                         $real_bp = $bp_apparent - ($pos-8);
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "+")&&(($motif eq "TRANYTRY"))){
                         $real_bp = ($pos-7)+$bp_apparent;
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "-")&&(($motif eq "TRANYTRY"))){
                         $real_bp = $bp_apparent - ($pos-7);
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "+")&&(($motif eq "TRANNYTRY"))){
                         $real_bp = ($pos-6)+$bp_apparent;
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
                 elsif (($strand eq "-")&&(($motif eq "TRANNYTRY"))){
                         $real_bp = $bp_apparent - ($pos-6);
                         $lariats{$id2}{"bp_real"} = $real_bp;
                         $lariats{$id2}{"bp_seq"} = $seq;
                 }
         }

 }


 open (FH, "circle.bed") or die $!;
 while (<FH>){
        ($chrom,$c1,$c2,$circ,$score,$strand)=(split /\t/,$_); chomp $strand;
        $bp = "";
        if ($strand eq "+"){$bp = $c2;}
        elsif ($strand eq "-"){$bp = $c1;}
        $id = $chrom.":".$bp."_".$strand;
        if (exists $lariats{$id}{"motif"}){
               $lariats{$id}{"motif"}="circle";
               $lariats{$id}{"bp_real"} = "null";
               $lariats{$id}{"bp_seq"} = "null";
        }
        else {print "error: $id\n";}
 }

 open (FH, "TS.bed") or die $!;
 while (<FH>){
         ($chrom,$c1,$c2,$TS,$score,$strand)=(split /\t/,$_); chomp $strand;
         $bp = "";
         if ($strand eq "+"){$bp = $c2;}
         elsif ($strand eq "-"){$bp = $c1;}
         $id = $chrom.":".$bp."_".$strand;
         if (exists $lariats{$id}{"motif"}){
                 $lariats{$id}{"motif"}="template_switching";
                 $lariats{$id}{"bp_real"} = "null";
                 $lariats{$id}{"bp_seq"} = "null";
         }
         else {print "error: $id\n";}
 }
```

```
foreach my $key (keys %lariats){
        $found = ""; $total_reads = 0; $uniq_reads = 0; $mismatch_reads = 0; $mismatch_uniq_reads = 0;
        foreach my $key2 (keys %{$lariat_file{$key}}){
                $total_reads = $total_reads + $lariat_file{$key}{$key2};
                if (exists $lariat_file_uniq{$key}{$key2}){$uniq_reads =
$uniq_reads+$lariat_file_uniq{$key}{$key2};}
                else {$lariat_file_uniq{$key}{$key2}=0;}
                if (exists $lariat_file_mut{$key}{$key2}){$mismatch_reads =
$mismatch_reads+$lariat_file_mut{$key}{$key2};}
                else {$lariat_file_mut{$key}{$key2}=0;}
                if (exists $lariat_file_mut_uniq{$key}{$key2}){$mismatch_uniq_reads =
$mismatch_uniq_reads+$lariat_file_mut_uniq{$key}{$key2};}
                else {$lariat_file_mut_uniq{$key}{$key2}=0;}
                $nextkey = $key2."(".$lariat_file{$key}{$key2}.";".$lariat_file_uniq{$key}{$key2}.")";
                $found = $found.$nextkey.",";
        }
        if ($found =~ /SRR/){
        print
$key,"\t",$lariats{$key}{"motif"},"\t",$lariats{$key}{"bp_real"},"\t",$lariats{$key}{"bp_seq"},"\t",$fo
und,"\t",$total_reads,"\t",$uniq_reads,"\t",$mismatch_reads,"\t",$mismatch_uniq_reads,"\n";
        }
}
```

```
#make_table_1.pl

use warnings;

open (FH,"BPs_apparent_real_wFoundFiles_final.txt") or die $!;

my %branchpoint;

while (<FH>){
	($app,$motif,$real,$seq,$found,$numreads,$uniqreads,$mutreads,$mutuniqreads)=(split /\t/,$_);
chomp $mutuniqreads;
	$app =~ /(\w+):(\d+)_([+-])/;
	$chrom = $1; $appnt = $2; $strand = $3;
	unless ($real eq "null"){
		$dist = abs($appnt-$real);}
	else {
		$dist = 0; $real = $appnt;
	}
	$id = $chrom."\t".$real."\t".$strand."\t".$motif;
	@list = (split /\,/,$found);

	if (exists $branchpoint{$id}{"read"}{$dist}){
		$branchpoint{$id}{"read"}{$dist}=$branchpoint{$id}{$dist}+$numreads;
	}
	else {$branchpoint{$id}{"read"}{$dist}=$numreads;}

	if (exists $branchpoint{$id}{"readuniq"}{$dist}){
		$branchpoint{$id}{"readuniq"}{$dist}=$branchpoint{$id}{$dist}+$uniqreads;
	}
	else {$branchpoint{$id}{"readuniq"}{$dist}=$uniqreads;}

	if (exists $branchpoint{$id}{"mutread"}{$dist}){
		$branchpoint{$id}{"mutread"}{$dist}=$branchpoint{$id}{$dist}+$mutreads;
	}
	else {$branchpoint{$id}{"mutread"}{$dist}=$mutreads;}

	if (exists $branchpoint{$id}{"mutreaduniq"}{$dist}){
		$branchpoint{$id}{"mutreaduniq"}{$dist}=$branchpoint{$id}{$dist}+$mutuniqreads;
	}
	else {$branchpoint{$id}{"mutreaduniq"}{$dist}=$mutuniqreads;}

	foreach $a (@list){
		if ($a =~ /\w+/){
			$a =~ /(.+)\((\d+)\;(\d+)\)/;
			$one = $1; $two = $2; $three = $3;
			$anew = $one."(".$appnt.";".$two.";".$three.")";
```

```
                    $branchpoint{$id}{"found"}{$anew}=1;
                }
        }
        $branchpoint{$id}{"seq"}=$seq;
}

foreach my $bp (keys %branchpoint){
        print $bp,"\t",$branchpoint{$bp}{"seq"},"\t";
        if (exists $branchpoint{$bp}{"read"}{5}){print $branchpoint{$bp}{"read"}{5},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"read"}{4}){print $branchpoint{$bp}{"read"}{4},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"read"}{3}){print $branchpoint{$bp}{"read"}{3},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"read"}{2}){print $branchpoint{$bp}{"read"}{2},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"read"}{1}){print $branchpoint{$bp}{"read"}{1},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"read"}{0}){print $branchpoint{$bp}{"read"}{0},"\t";}
        else {print "0\t";}

        if (exists $branchpoint{$bp}{"readuniq"}{5}){print $branchpoint{$bp}{"readuniq"}{5},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"readuniq"}{4}){print $branchpoint{$bp}{"readuniq"}{4},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"readuniq"}{3}){print $branchpoint{$bp}{"readuniq"}{3},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"readuniq"}{2}){print $branchpoint{$bp}{"readuniq"}{2},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"readuniq"}{1}){print $branchpoint{$bp}{"readuniq"}{1},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"readuniq"}{0}){print $branchpoint{$bp}{"readuniq"}{0},"\t";}
        else {print "0\t";}

        if (exists $branchpoint{$bp}{"mutread"}{5}){print $branchpoint{$bp}{"mutread"}{5},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"mutread"}{4}){print $branchpoint{$bp}{"mutread"}{4},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"mutread"}{3}){print $branchpoint{$bp}{"mutread"}{3},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"mutread"}{2}){print $branchpoint{$bp}{"mutread"}{2},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"mutread"}{1}){print $branchpoint{$bp}{"mutread"}{1},",";}
        else {print "0,";}
        if (exists $branchpoint{$bp}{"mutread"}{0}){print $branchpoint{$bp}{"mutread"}{0},"\t";}
        else {print "0\t";}
```

```
	if (exists $branchpoint{$bp}{"mutreaduniq"}{5}){print
$branchpoint{$bp}{"mutreaduniq"}{5},",";}
	else {print "0,";}
	if (exists $branchpoint{$bp}{"mutreaduniq"}{4}){print
$branchpoint{$bp}{"mutreaduniq"}{4},",";}
	else {print "0,";}
	if (exists $branchpoint{$bp}{"mutreaduniq"}{3}){print
$branchpoint{$bp}{"mutreaduniq"}{3},",";}
	else {print "0,";}
	if (exists $branchpoint{$bp}{"mutreaduniq"}{2}){print
$branchpoint{$bp}{"mutreaduniq"}{2},",";}
	else {print "0,";}
	if (exists $branchpoint{$bp}{"mutreaduniq"}{1}){print
$branchpoint{$bp}{"mutreaduniq"}{1},",";}
	else {print "0,";}
	if (exists $branchpoint{$bp}{"mutreaduniq"}{0}){print
$branchpoint{$bp}{"mutreaduniq"}{0},"\t";}
	else {print "0\t";}



	foreach my $source (keys %{$branchpoint{$bp}{"found"}}){
		print $source,",";
	}
	print "\n";
}
```

```
#make_BP_table2.pl

use warnings;

open (FH, "BP_table_1_final.txt") or die $!;
while (<FH>){
        unless ($_ =~ /branchpoint/){
        $line = $_;
        ($chrom,$bp,$strand,$motif,$seq,$pos1,$pos2,$pos3,$pos4,$source)=(split /\t/,$_);
        chomp $source;
        $pos1new = "";
        $pos2new = "";
        $pos3new = "";
        $pos4new = "";

        if (($motif eq "canonical")||($motif eq "canonicalC")||($motif eq "TRAYTRY")){
                if ($pos1 =~ /0\,0\,0\,(\d+)\,(\d+)\,(\d+)/){
                        $pos1new = $1.",".$2."(".$3.")";
                }
                else {print "error: $line";}
        }
        elsif (($motif eq "canonical2nt")||($motif eq "TRANYTRY")){
                if ($pos1 =~ /0\,0\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
                        $pos1new = $1.",".$2."(".$3.",".$4.")";
                }
                else {print "error: $line";}
        }
        elsif (($motif eq "TRANNYTRY")){
                if ($pos1 =~ /0\,(\d+)\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
                        $pos1new = $1.",".$2."(".$3.",".$4.",".$5.")";
                }
                else {print "error: $line";}
        }
        elsif ($motif eq "none"){
                if ($pos1 =~ /0\,0\,0\,0\,0\,(\d+)/){
                        $pos1new = $1;
                }
                else {print "error: $line";}
        }
        elsif ($motif eq "template_switching"){
              if ($pos1 =~ /0\,0\,0\,0\,0\,(\d+)/){
                    $pos1new = $1;
              }
              else {print "error: $line";}
       }
       elsif ($motif eq "circle"){
```

```
	if ($pos1 =~ /0\,0\,0\,0\,0\,(\d+)/){
		$pos1new = $1;
	}
	else {print "error: $line";}
}



if (($motif eq "canonical")||($motif eq "canonicalC")||($motif eq "TRAYTRY")){
	if ($pos2 =~ /0\,0\,0\,(\d+)\,(\d+)\,(\d+)/){
		$pos2new = $1.",".$2."(".$3.")";
	}
	else {print "error: $line";}
}
elsif (($motif eq "canonical2nt")||($motif eq "TRANYTRY")){
	if ($pos2 =~ /0\,0\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
		$pos2new = $1.",".$2."(".$3.",".$4.")";
	}
	else {print "error: $line";}
}
elsif (($motif eq "TRANNYTRY")){
	if ($pos2 =~ /0\,(\d+)\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
		$pos2new = $1.",".$2."(".$3.",".$4.",".$5.")";
	}
	else {print "error: $line";}
}
elsif ($motif eq "none"){
	if ($pos2 =~ /0\,0\,0\,0\,0\,(\d+)/){
		$pos2new = $1;
	}
	else {print "error: $line";}
}
elsif ($motif eq "template_switching"){
	if ($pos2 =~ /0\,0\,0\,0\,0\,(\d+)/){
		$pos2new = $1;
	}
	else {print "error: $line";}
}
elsif ($motif eq "circle"){
	if ($pos2 =~ /0\,0\,0\,0\,0\,(\d+)/){
		$pos2new = $1;
	}
	else {print "error: $line";}
}
```

```
if (($motif eq "canonical")||($motif eq "canonicalC")||($motif eq "TRAYTRY")){
        if ($pos3 =~ /0\,0\,0\,(\d+)\,(\d+)\,(\d+)/){
                $pos3new = $1.",".$2."(".$3.")";
        }
        else {print "error: $line";}
}
elsif (($motif eq "canonical2nt")||($motif eq "TRANYTRY")){
        if ($pos3 =~ /0\,0\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
                $pos3new = $1.",".$2."(".$3.",".$4.")";
        }
        else {print "error: $line";}
}
elsif (($motif eq "TRANNYTRY")){
        if ($pos3 =~ /0\,(\d+)\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
                $pos3new = $1.",".$2."(".$3.",".$4.",".$5.")";
        }
        else {print "error: $line";}
}
elsif ($motif eq "none"){
        if ($pos3 =~ /0\,0\,0\,0\,0\,(\d+)/){
                $pos3new = $1;
        }
        else {print "error: $line";}
}
elsif ($motif eq "template_switching"){
        if ($pos3 =~ /0\,0\,0\,0\,0\,(\d+)/){
                $pos3new = $1;
        }
        else {print "error: $line";}
}
elsif ($motif eq "circle"){
        if ($pos3 =~ /0\,0\,0\,0\,0\,(\d+)/){
                $pos3new = $1;
        }
        else {print "error: $line";}
}
```

```
if (($motif eq "canonical")||($motif eq "canonicalC")||($motif eq "TRAYTRY")){
		if ($pos4 =~ /0\,0\,0\,(\d+)\,(\d+)\,(\d+)/){
			$pos4new = $1.",".$2."(".$3.")";
		}
		else {print "error: $line";}
	}
	elsif (($motif eq "canonical2nt")||($motif eq "TRANYTRY")){
		if ($pos4 =~ /0\,0\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
			$pos4new = $1.",".$2."(".$3.",".$4.")";
		}
		else {print "error: $line";}
	}
	elsif (($motif eq "TRANNYTRY")){
		if ($pos4 =~ /0\,(\d+)\,(\d+)\,(\d+)\,(\d+)\,(\d+)/){
			$pos4new = $1.",".$2."(".$3.",".$4.",".$5.")";
		}
		else {print "error: $line";}
	}
	elsif ($motif eq "none"){
		if ($pos4 =~ /0\,0\,0\,0\,0\,(\d+)/){
			$pos4new = $1;
		}
		else {print "error: $line";}
	}
	elsif ($motif eq "template_switching"){
		if ($pos4 =~ /0\,0\,0\,0\,0\,(\d+)/){
			$pos4new = $1;
		}
		else {print "error: $line";}
	}
	elsif ($motif eq "circle"){
		if ($pos4 =~ /0\,0\,0\,0\,0\,(\d+)/){
			$pos4new = $1;
		}
		else {print "error: $line";}
	}


	print
$chrom,"\t",$bp,"\t",$strand,"\t",$motif,"\t",$seq,"\t",$pos1new,"\t",$pos2new,"\t",$pos3new,"\t",$pos4
new,"\t",$source,"\n";
	}

	else {print $_;}
}
```

```
#assign_3ss_and_category.pl

use warnings;

my %ss;

open (FH, "hg19_introns.bed") or die $!;
while (<FH>){
   $chrom = (split /\t/,$_)[0];
   $coord1 = (split /\t/,$_)[1];
   $coord2 = (split /\t/,$_)[2];
   $strand = (split /\t/,$_)[5]; chomp $strand;
   if ($strand eq "+"){
      $threess = $chrom."_".$strand."_".$coord2;
      $ss{$threess}=1;
   }
   if ($strand eq "-"){
      $threess = $chrom."_".$strand."_".$coord1;
      $ss{$threess}=1;
   }
} close FH;
open (FH, $ARGV[0]) or die $!;
while (<FH>){
   $line = $_;

       ($chrom,$bp,$strand,$motif,$seq,$pos1,$pos2,$pos3,$pos4,$source)=(split /\t/,$_);chomp
$source;

   $flag = 0;
   my $j= 0;
   $category = "";
   $true3ss = "";
   $dist = "";
   while ($flag==0){
       $threess = "";
       if ($strand eq "+"){
      $threess = $chrom."_".$strand."_".($bp+$j);
      }
      elsif ($strand eq "-"){
      $threess = $chrom."_".$strand."_".($bp-$j);
      }

      if (exists $ss{$threess}){

         if ($strand eq "+"){
                                    $threess = $bp+$j;
```

```
                    $flag = 1;
                    $true3ss = $threess;

                }
                if ($strand eq "-"){
                                        $threess = $bp-$j;
                    $flag = 1;
                    $true3ss = $threess;

                }
            }
            $j++;
            if ($j > 1000000){$flag = 2;}
        }
    if ($flag ==2){$dist = "null"; $true3ss = "null"; $category = "null";}
    else {
        if ($strand eq "+"){
            $dist = $bp - $true3ss;
        }
        elsif ($strand eq "-"){
            $dist = $true3ss - $bp;
        }
        if ($dist > 0){
            $category = "inside 3ss";
        }
        elsif ($dist ==0){
             $category = "circle";
        }
        elsif ($dist >=-10){
            $category = "proximal";
        }
        elsif ($dist >= -60){
            $category = "expected";
        }
        else {
            $category = "distal";
        }}

          print
$chrom,"\t",$bp,"\t",$strand,"\t",$motif,"\t",$seq,"\t",$pos1,"\t",$pos2,"\t",$pos3,"\t",$pos4,"\t",$true3ss,
"\t",$dist,"\t",$category,"\t",$source,"\n";

}
```

```
#make_table_4.pl

use warnings;

open (FH, "BP_table_3_final.txt") or die $!;

while (<FH>){
	($chrom,$bp,$strand,$motif,$seq,$readtot,$readuniq,$misread,$misuniq,$threeprss,$bpdist,$categ
ory,$sources)=(split /\t/,$_);
	$line = $_;
	chomp $sources;
	$seq2 = "";$bpnt = "";
	if ($line =~ /circle/){
		$motif = "circle";
		$c1 = $bp - 5; $c2 = $bp+5;
		open (FHout, ">temp.bed") or die $!;
		print FHout $chrom,"\t",$c1,"\t",$c2,"\tbed\t0\t",$strand,"\n";
		close FHout;
		$cmd = "bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -bed temp.bed -s -tab -
fo temp.fa";
		system($cmd);
		open (FH2,"temp.fa") or die $!;
		$seqcirc = "";
		while (<FH2>){
			($id,$seqcirc)=(split /\t/,$_); chomp $seqcirc;
		}
		close FH2;
		$seqcirc =~ tr/[a-z]/[A-Z]/;
		$seqcirc =~ /^(....)(.)(.....)/;
		$seq2 = $1.$2."*".$3;
		$bpnt = $2;
	}

	elsif ($motif eq "none"){
		$c1 = $bp - 5; $c2 = $bp+5;
		open (FHout, ">temp.bed") or die $!;
		print FHout $chrom,"\t",$c1,"\t",$c2,"\tbed\t0\t",$strand,"\n";
		close FHout;
		$cmd = "bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -bed temp.bed -s -tab -
fo temp.fa";
		system($cmd);
		open (FH2,"temp.fa") or die $!;
		$seqcirc = "";
		while (<FH2>){
			($id,$seqcirc)=(split /\t/,$_); chomp $seqcirc;
		}
```

```
			close FH2;
			$seqcirc =~ tr/[a-z]/[A-Z]/;
			$seqcirc =~ /^(....)(.)(.....)/;
			$seq2 = $1.$2."*".$3;
			$bpnt = $2;
		}
		elsif ($motif eq "template_switching"){
			$c1 = $bp - 5; $c2 = $bp+5;
			open (FHout, ">temp.bed") or die $!;
			print FHout $chrom,"\t",$c1,"\t",$c2,"\tbed\t0\t",$strand,"\n";
			close FHout;
			$cmd = "bedtools getfasta -fi /home/annotations/bowtie/hg19/hg19.fa -bed temp.bed -s -tab -
fo temp.fa";
			system($cmd);
			open (FH2,"temp.fa") or die $!;
			$seqcirc = "";
			while (<FH2>){
				($id,$seqcirc)=(split /\t/,$_); chomp $seqcirc;
			}
			close FH2;
			$seqcirc =~ tr/[a-z]/[A-Z]/;
			$seqcirc =~ /^(....)(.)(.....)/;
			$seq2 = $1.$2."*".$3;
			$bpnt = $2;
		}
		elsif ($motif eq "canonical"){
			if ($seq =~ /^(.....)(.)(.)$/){
				$us = $1; $bulge = $2; $ds = $3;
				$bpnt = $bulge;
				$seq2 = $us."(".$bulge."*)".$ds;
			}
			else {print "error: $line";}
			$readtot =~ s/\(/\,\(/;
			$readuniq =~ s/\(/\,\(/;
			$misread =~ s/\(/\,\(/;
			$misuniq =~ s/\(/\,\(/;
		}
		elsif ($motif eq "canonicalC"){
			if ($seq =~ /^(.....)(.)(.)$/){
				$us = $1; $bulge = $2; $ds = $3;
				$bpnt = $bulge;
				$seq2 = $us."(".$bulge."*)".$ds;
			}
			else {print "error: $line";}
			$readtot =~ s/\(/\,\(/;
			$readuniq =~ s/\(/\,\(/;
        $misread =~ s/\(/\,\(/;
        $misuniq =~ s/\(/\,\(/;
}
elsif ($motif eq "canonical2nt"){
        if ($seq =~ /^(.....)(..)(.)$/){
                $us = $1; $bulge = $2; $ds = $3;
                $bpnts = $bulge;
                if ($readuniq =~ /(\d+)\,(\d+)\((\d+)\,(\d+)\)/){
                        $min2 = $1; $min1 = $2; $b1 = $3; $b2 = $4;
                        $pos1score = $min2+$min1+$b1;
                        $pos2score = $min1+$b1+$b2;
                        if ($pos2score == $pos1score){
                                if ($b1>$b2){
                                        $bulge =~ /(.)(.)/;
                                        $b1nt = $1; $b2nt = $2;
                                        $seq2 = $us."(".$b1nt."*".$b2nt.")".$ds;
                                        $bpnt = $b1nt;
                                        if ($strand eq "+"){
                                                $bp = $bp-1; $bpdist = $bpdist+1;
                                        }
                                        elsif ($strand eq "-"){
                                                $bp = $bp+1; $bpdist = $bpdist+1;
                                        }
                                }
                                else {
                                        $bulge =~ /(.)(.)/;
                                        $b1nt = $1; $b2nt = $2;
                                        $seq2 = $us."(".$b1nt.$b2nt."*)".$ds;
                                        $bpnt = $b2nt;
                                }
                        }
                        elsif ($pos2score>$pos1score){

                                $bulge =~ /(.)(.)/;
                                $b1nt = $1; $b2nt = $2;
                                $seq2 = $us."(".$b1nt.$b2nt."*)".$ds;
                                $bpnt = $b2nt;
                        }
                        else {
                                $bulge =~ /(.)(.)/;
                                $b1nt = $1; $b2nt = $2;
                                $seq2 = $us."(".$b1nt."*".$b2nt.")".$ds;
                                $bpnt = $b1nt;
                                if ($strand eq "+"){
                                        $bp = $bp-1; $bpdist = $bpdist+1;
                                }
                        elsif ($strand eq "-"){
                                $bp = $bp+1; $bpdist = $bpdist+1;
                        }

                }
        }
        else {print "error: $line";}
    }
    else {print "error: $line";}
    $readtot =~ s/\(/\,\(/;
    $readuniq =~ s/\(/\,\(/;
    $misread =~ s/\(/\,\(/;
    $misuniq =~ s/\(/\,\(/;
}
elsif ($motif eq "TRAYTRY"){
    if ($seq =~ /^(..)(.)(....)$/){
        $us = $1; $bulge = $2; $ds = $3;
        $bpnt = $bulge;
        $seq2 = $us."(".$bulge."*)".$ds;
    }
    else {print "error: $line";}
    $readtot =~ s/\(/\,\(/;
    $readuniq =~ s/\(/\,\(/;
    $misread =~ s/\(/\,\(/;
    $misuniq =~ s/\(/\,\(/;
}
elsif ($motif eq "TRANYTRY"){
    if ($seq =~ /^(..)(..)(....)$/){
        $us = $1; $bulge = $2; $ds = $3;
        $bpnts = $bulge;
        if ($readuniq =~ /(\d+)\,(\d+)\((\d+)\,(\d+)\)/){
            $min2 = $1; $min1 = $2; $b1 = $3; $b2 = $4;
            $pos1score = $min2+$min1+$b1;
            $pos2score = $min1+$b1+$b2;
            if ($pos2score == $pos1score){
                if ($b1>$b2){
                    $bulge =~ /(.)(.)/;
                    $b1nt = $1; $b2nt = $2;
                    $seq2 = $us."(".$b1nt."*".$b2nt.")".$ds;
                    $bpnt = $b1nt;
                    if ($strand eq "+"){
                        $bp = $bp-1; $bpdist = $bpdist+1;
                    }
                    elsif ($strand eq "-"){
                        $bp = $bp+1; $bpdist = $bpdist+1;
                    }
```

```
					}
					else {
						$bulge =~ /(.)(.)/;
						$b1nt = $1; $b2nt = $2;
						$seq2 = $us."(".$b1nt.$b2nt."*)".$ds;
						$bpnt = $b2nt;
					}
				}
				elsif ($pos2score>$pos1score){

					$bulge =~ /(.)(.)/;
					$b1nt = $1; $b2nt = $2;
					$seq2 = $us."(".$b1nt.$b2nt."*)".$ds;
					$bpnt = $b2nt;
				}
				else {
					$bulge =~ /(.)(.)/;
					$b1nt = $1; $b2nt = $2;
					$seq2 = $us."(".$b1nt."*".$b2nt.")".$ds;
					$bpnt = $b1nt;
					if ($strand eq "+"){
						$bp = $bp-1; $bpdist = $bpdist+1;
					}
					elsif ($strand eq "-"){
						$bp = $bp+1; $bpdist = $bpdist+1;
					}

				}
			}
			else {print "error: $line";}
		}
		else {print "error: $line";}
		$readtot =~ s/\(/\,\(/;
		$readuniq =~ s/\(/\,\(/;
		$misread =~ s/\(/\,\(/;
		$misuniq =~ s/\(/\,\(/;
	}
	elsif ($motif eq "TRANNYTRY"){
		if ($seq =~ /^(..)(...)(....)$/){
			$us = $1; $bulge = $2; $ds = $3;
			$bpnts = $bulge;
			if ($readuniq =~ /(\d+)\,(\d+)\((\d+)\,(\d+)\,(\d+)\)/){
				$min2 = $1; $min1 = $2; $b1 = $3; $b2 = $4; $b3 = $5;
				$pos1score = $min2+$min1+$b1;
				$pos2score = $min1+$b1+$b2;
				$pos3score = $b1+$b2+$b3;
```

```
#allscoresequal
if (($pos1score==$pos2score)&&($pos2score==$pos3score)){

        if (($b1 ==$b2)&&($b2==$b3)){
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
                $bpnt = $b3nt;
        }
        elsif (($b1>$b2)&&($b1>$b3)){
                if ($strand eq "+"){
                        $bp = $bp-2;
                        $bpdist = $bpdist+2;
                }
                elsif ($strand eq "-"){
                        $bp = $bp+2;
                        $bpdist = $bpdist+2;
                }
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt."*".$b2nt.$b3nt.")".$ds;
                $bpnt = $b1nt;
        }
        elsif (($b2 > $b3)&&($b2>$b1)){
                if ($strand eq "+"){
                        $bp = $bp-1;
                        $bpdist = $bpdist+1;
                }
                elsif ($strand eq "-"){
                        $bp = $bp+1;
                        $bpdist = $bpdist+1;
                }
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt.$b2nt."*".$b3nt.")".$ds;
                $bpnt = $b2nt;
        }
        elsif (($b3>$b2)&&($b3>$b1)){
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
                $bpnt = $b3nt;
        }
        elsif (($b1==$b2)&&($b1>$b3)){
                if ($strand eq "+"){
                        $bp = $bp-1;
```

```
					$bpdist = $bpdist+1;
				}
				elsif ($strand eq "-"){
					$bp = $bp+1;
					$bpdist = $bpdist+1;
				}
				$bulge =~ /(.)(.)(.)/;
				$b1nt = $1; $b2nt = $2;$b3nt = $3;
				$seq2 = $us."(".$b1nt.$b2nt."*".$b3nt.")".$ds;
				$bpnt = $b2nt;
			}
			else {
				$bulge =~ /(.)(.)(.)/;
				$b1nt = $1; $b2nt = $2;$b3nt = $3;
				$seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
				$bpnt = $b3nt;
			}
		}
		#pos3&2 > pos1, but are equal
		elsif (($pos3score>$pos1score)&&($pos3score==$pos2score)){
			if ($b2>$b3){
				if ($strand eq "+"){
					$bp = $bp-1;
					$bpdist = $bpdist+1;
				}
				elsif ($strand eq "-"){
					$bp = $bp+1;
					$bpdist = $bpdist+1;
				}
				$bulge =~ /(.)(.)(.)/;
				$b1nt = $1; $b2nt = $2;$b3nt = $3;
				$seq2 = $us."(".$b1nt.$b2nt."*".$b3nt.")".$ds;
				$bpnt = $b2nt;
			}
			else {
				$bulge =~ /(.)(.)(.)/;
				$b1nt = $1; $b2nt = $2;$b3nt = $3;
				$seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
				$bpnt = $b3nt;
			}
		}
		elsif (($pos3score ==$pos1score)&&($pos1score>$pos2score)){
			if ($b1>$b3){
				if ($strand eq "+"){
					$bp = $bp-2;
					$bpdist = $bpdist+2;
```

```
                }
                elsif ($strand eq "-"){
                        $bp = $bp+2;
                        $bpdist = $bpdist+2;
                }
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt."*".$b2nt.$b3nt.")".$ds;
                $bpnt = $b1nt;
        }
        else {
                $bulge =~ /(.)(.)(.)/;
                $b1nt = $1; $b2nt = $2;$b3nt = $3;
                $seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
                $bpnt = $b3nt;
        }
}
#pos1 >2/3
elsif (($pos1score>$pos2score)&&($pos1score>$pos3score)){
        if ($strand eq "+"){
                $bp = $bp-2; $bpdist = $bpdist+2;
        }
        elsif ($strand eq "-"){
                $bp = $bp+2; $bpdist = $bpdist+2;
        }
        $bulge =~ /(.)(.)(.)/;
        $b1nt = $1; $b2nt = $2;$b3nt = $3;
        $seq2 = $us."(".$b1nt."*".$b2nt.$b3nt.")".$ds;
        $bpnt = $b1nt;
}
#pos2 > 1/3
elsif (($pos2score>$pos1score)&&($pos2score>$pos3score)){
        if ($strand eq "+"){
                $bp = $bp-1; $bpdist = $bpdist+1;
        }
        elsif ($strand eq "-"){
                $bp = $bp+1; $bpdist = $bpdist+1;
        }
        $bulge =~ /(.)(.)(.)/;
        $b1nt = $1; $b2nt = $2;$b3nt = $3;
        $seq2 = $us."(".$b1nt.$b2nt."*".$b3nt.")".$ds;
        $bpnt = $b2nt;
}
#pos3 > 1/2
elsif (($pos3score>$pos1score)&&($pos3score>$pos2score)){
        $bulge =~ /(.)(.)(.)/;
```

```
					$b1nt = $1; $b2nt = $2;$b3nt = $3;
					$seq2 = $us."(".$b1nt.$b2nt.$b3nt."*)".$ds;
					$bpnt = $b3nt;
				}
				#pos1=2, > 3
				elsif (($pos1score==$pos2score)&&($pos1score>$pos3score)){

					if ($b1>$b3){
						if ($strand eq "+"){
							$bp = $bp-2;
							$bpdist = $bpdist+2;
						}
						elsif ($strand eq "-"){
							$bp = $bp+2;
							$bpdist = $bpdist+2;
						}
						$bulge =~ /(.)(.)(.)/;
						$b1nt = $1; $b2nt = $2;$b3nt = $3;
						$seq2 = $us."(".$b1nt."*".$b2nt.$b3nt.")".$ds;
						$bpnt = $b1nt;
					}
					else {
						$bulge =~ /(.)(.)(.)/;
						$b1nt = $1; $b2nt = $2;$b3nt = $3;
						$seq2 = $us."(".$b1nt.$b2nt."*".$b3nt.")".$ds;
						$bpnt = $b2nt;
					}
				}
				else {print "logic bpfind error: $line";}

			}
			else {print "error: $line";}
		}
		else {print "error: $line";}
		$readtot =~ s/\(/\,\(/;
		$readuniq =~ s/\(/\,\(/;
		$misread =~ s/\(/\,\(/;
		$misuniq =~ s/\(/\,\(/;
}

$mutation = "no"; $bias = "no";
$multiple = "no";
$totalreadevidence = 0; $totaluniqreadevidence = 0;
$t = $readtot; $u = $readuniq; $m = $misread;
$t =~ s/[[:punct:]]/\t/g;
$u =~ s/[[:punct:]]/\t/g;
```

```
	$m =~ s/[[:punct:]]/\t/g;
	@tl = (split /\t/,$t); @ul = (split /\t/,$u); @ml = (split /\t/,$m);
	foreach (@tl){
		if ($_ =~ /(\d+)/){
		$totalreadevidence = $totalreadevidence+$1;
		}
	}
	foreach (@ul){
		if ($_ =~ /(\d+)/){
		$totaluniqreadevidence = $totaluniqreadevidence+$1;
		}
	}
	foreach (@ml){
		if ($_ =~ /(\d+)/){
		if ($_ >0){$mutation = "yes";}
		}
	}
	@sourcelist = (split /\,/,$sources);
	$num = 0;
	foreach (@sourcelist){
		if ($_ =~ /\w+/){
			$num++;
		}
		if ($_ =~ /fairbrother/){}
		elsif ($_ =~ /Mattick/){}
		else {$bias = "yes";}
	}
	if ($num > 1){$multiple = "yes";}


	print
$chrom,"\t",$bp,"\t",$strand,"\t",$motif,"\t",$seq2,"\t",$bpnt,"\t",$threeprss,"\t",$bpdist,"\t",$category,"\t
",$totalreadevidence,"\t",$totaluniqreadevidence,"\t",$mutation,"\t",$multiple,"\t",$bias,"\t",$readtot,"\t"
,$readuniq,"\t",$misread,"\t",$misuniq,"\t",$sources,"\n";
}
#make_table_5.pl

open (FH, "BP_table_4_final.txt") or die $!;

open (FHout, ">BP_table_5_final.txt") or die $!;

while (<FH>){
	($chrom,$bp,$strand,$motif,$seq,$bpnt,$threess,$bpdist,$pos,$count,$countuniq,$q1,$q2,$q3,$c1,
$c2,$m1,$m2,$files)=(split /\t/,$_); chomp $files;
	my %fs;
	@f = (split /\,/,$files);
```

```
        foreach $a (@f){
                $a =~ /(.+)\(\d+\;(\d+)\;(\d+)\)/;
                $file = $1; $read = $2; $uniq = $3;
                if (exists $fs{$file}){
                        $fs{$file}[0] = $fs{$file}[0]+$read;
                        $fs{$file}[1] = $fs{$file}[1]+$uniq;
                }
                else {
                        $fs{$file}[0] = $read;
                        $fs{$file}[1] = $uniq;
                }
        }
        $filestring = "";
        foreach my $key (keys %fs){
                $filestring = $filestring.$key."(".$fs{$key}[0].";".$fs{$key}[1]."),";
        }
        print FHout
$chrom,"\t",$bp,"\t",$strand,"\t",$motif,"\t",$seq,"\t",$bpnt,"\t",$threess,"\t",$bpdist,"\t",$pos,"\t",$count
,"\t",$countuniq,"\t",$q1,"\t",$q2,"\t",$q3,"\t",$c1,"\t",$c2,"\t",$m1,"\t",$m2,"\t",$filestring,"\n";
}
```