```perl
#!/usr/bin/perl
open(FILE, "accepted_hits_sorted.sam") or die("Unable to open file");
open(OUTPUT, ">final-alignment.sam") or die("Unable to open file");

use strict;
my $i;
my $line;
my $line1;
my @group;
my @sam_coord;
my @sam_coord0;
my @ID0;
my @sam_coord1;
my @ID1;
my @record;
my $j;
my @as0;
my @as;
my $index;
my $maxval;
my $number;
my $location;
my $location1;
my @sam_coord2;
my $m;
my $x;
my @unique;
my @fl;
my $y;
my $bestline;
my $mut_type = "T->C";
my @sam_coord3;
my @sam_coord4;
my @sam_coord5;
my $m3,
my $x3;
my @MD3;
my $strand3;
my $CIGAR3;
my $seq3;
my $m4;
my $x4;
my @MD4;
my $CIGAR4;
my $strand4;
my $seq4;
my @ID3;
my @read3;
my @ID4;
my @read4;
my $aa;
my $bb;
my $cc;
my $dd;
my @mut_pos3=();
my @mut_pos4=();
my $length;
my $neg1;
my $neg2;
my $n=0;
my @header;
my $element;

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

if ($line=~/^\@.*/) { print OUTPUT "$line\n";
                      next;
                    }

#@sam_coord=split(/\s+/,"$line");
#if ($sam_coord[2] eq "chrM") {next;}

# put all aligments together from one single original read

  $i=0;
  $group[$i] = $line;
  @sam_coord0=split(/\s+/,"$group[$i]");

  if ($sam_coord0[0] =~ /\./) {@ID0=split(/\./, "$sam_coord0[0]");}
  elsif ($sam_coord0[0] =~ /\#/) {@ID0=split(/\#/, "$sam_coord0[0]");}
  else {$ID0[0]=$sam_coord0[0];}

  if(eof FILE)
  {
   print OUTPUT "$line\n";
   next;
  }

  do {
      $line = <FILE>;
      $line1=$line;
      chomp($line);
      @sam_coord1=split(/\s+/,"$line");
      if ($sam_coord1[0] =~ /\./) {@ID1=split(/\./, "$sam_coord1[0]");}
      elsif ($sam_coord1[0] =~ /\#/) {@ID1=split(/\#/, "$sam_coord1[0]");}
      else {$ID1[0]=$sam_coord1[0];}

      if ($ID1[0] eq $ID0[0])
        {$i=$i+1;
         $group[$i] = $line;
        }
     } while ($ID1[0] eq $ID0[0]) ;

#  for ( 0 .. $#group )
#      {print "$group[$_]\n";}
#  print "\n\n";

  unless ($ID1[0] eq "") {seek(FILE, -length($line1), 1);}

# find the alignment score for each member
  $j=0;
  do { @record = split(/\s+/,"$group[$j]");
      @as0= split(/\:/,"$record[11]");
      $as[$j] = $as0[2];
      $j = $j+1;
     } until ($j eq scalar@group);

# find the largest alignment scores
  $index = 0;
  $maxval = $as[$index];
  for ( 0 .. $#as )
              {
               if ( $maxval < $as[$_] )
                 {
                   $index = $_;
                   $maxval = $as[$_];
                 }
              }

# count how many largest alignemnt scores in the group
  $number=0;
  for ( 0 .. $#as )
              {
               if ( $as[$_] eq $maxval )
                 {
                   $number= $number+1;
                 }
              }


   if ($number>0) {
          $location = 0;
          $location1= 0;

         # only keep alignments in @fl which are uniquely aligned, with highest
aligning score
          for ( 0 .. $#as )
            {
              if( $as[$_] eq $maxval)
                {
                  $location= $_;
                  @sam_coord2=split(/\s+/,"$group[$location]");

                  $m = scalar@sam_coord2;
                  do {$m = $m-1;} until ($sam_coord2[$m] =~ /^NH.*/);
                  $x = $m;
                  @unique=split(/\:/, "$sam_coord2[$x]");

                  if ($unique[2] eq 1) {$fl[$location1]= $group[$location];
                                $location1= $location1+1;  }
                }
             }


          $length = scalar@fl;

#           for ( 0 .. $#fl )
#              {print "$fl[$_]\n";}
#           print "\n\n";


          if ($length > 1) {

              #sort the array consisted of best unique alignemnts
                sort_alignment (\@fl);

                $y=0;
                $bestline = $fl[$y];
                $neg1=0;
                $neg2=0;
```

```
     while(($y < ($length-1)) and ($neg1 eq 0) and ($neg2 eq 0)) {
       $y = $y+1;

       @sam_coord3=split(/\s+/,"$bestline");
       @sam_coord4=split(/\s+/,"$fl[$y]");

       $m3= scalar@sam_coord3;
       do {$m3 = $m3-1;} until ($sam_coord3[$m3] =~ /^MD.*/);
       $x3 = $m3;
       @MD3=split(/\:/, "$sam_coord3[$x3]");

#      print "$MD3[2]\n";

       if ($sam_coord3[1] eq 0) { $strand3= "+";}
       if ($sam_coord3[1] eq 16) { $strand3= "-";}
       $CIGAR3 = $sam_coord3[5];
#      may need to consider other numbers, besides 0 and 16
       $seq3= $sam_coord3[9];


       $m4= scalar@sam_coord4;
       do {$m4 = $m4-1;} until ($sam_coord4[$m4] =~ /^MD.*/);
       $x4 = $m4;
       @MD4=split(/\:/, "$sam_coord4[$x4]");

#      print "$MD4[2]\n";

       if ($sam_coord4[1] eq 0) { $strand4= "+";}
       if ($sam_coord4[1] eq 16) { $strand4= "-";}
       $CIGAR4 = $sam_coord4[5];
       $seq4= $sam_coord4[9];


       if ($sam_coord3[0] =~ /\./) {@read3=split(/\./, "$sam_coord3[0]");}
       else {$read3[1]=0;}

       if ($sam_coord4[0] =~ /\./) {@read4=split(/\./, "$sam_coord4[0]");}
       else {$read4[1]=0;}


#      @ID3=split(/\#/, "$sam_coord3[0]");
#      if ($ID3[1]=~/\./) {@read3 = split(/\./, "$ID3[1]");}
#      else {$read3[1]=0;}
#
#      @ID4=split(/\#/, "$sam_coord4[0]");
#      if ($ID4[1]=~/\./) {@read4 = split(/\./, "$ID4[1]");}
```

```
            #       else {$read4[1]=0;}


                  if (($sam_coord3[1] eq $sam_coord4[1]) and ($sam_coord3[2] eq
$sam_coord4[2]))
                   {
                     $aa = abs ($sam_coord3[3]-$sam_coord4[3]);
                     $bb = abs ($read3[1]- $read4[1]);

                     if ($read3[1] < $read4[1]) {
                     $cc = length($sam_coord3[9])+$read3[1];
                     $dd = abs ($cc -$read4[1]);
                     }else {
                     $cc = length($sam_coord3[9])+$read4[1];
                     $dd = abs ($cc -$read3[1]);
                     }

                     if ( (abs($aa-$bb)<3) or (abs($aa-$dd)<3))
                      {


read_mut($mut_type,$CIGAR3,$seq3,$MD3[2],\@mut_pos3,$strand3);

read_mut($mut_type,$CIGAR4,$seq4,$MD4[2],\@mut_pos4,$strand4);

                       if((scalar@mut_pos3 eq 0) and (scalar@mut_pos4 > 0)) {$bestline
= $fl[$y];}

                      } else {$neg2=$neg2+1;}
                    }else {$neg1=$neg1+1;}
               }

              if (($neg1 eq 0) and ($neg2 eq 0))
                { @sam_coord=split(/\s+/,"$bestline");
                  unless ($sam_coord[2] eq "chrM") {print OUTPUT "$bestline\n";}
                }

          }

          if ($length eq 1)
            { @sam_coord5=split(/\s+/,"$fl[0]");
              unless ($sam_coord5[2] eq "chrM") {print OUTPUT "$fl[0]\n";}
            }

    }
```

```
@group=();
@sam_coord=();
@sam_coord0=();
@ID0=();
@sam_coord1=();
@ID1=();
@record=();
@as0=();
@as=();
$location=0;
$location1=0;
@sam_coord2=();
$m=0;
$x=0;
@unique=();
@fl=();
$y=0;
@sam_coord3=();
@sam_coord4=();
@sam_coord5=();
$m3=0,
$x3=0;
@MD3=();
$m4=0;
$x4=0;
@MD4=();
@ID3=();
@read3=();
@ID4=();
@read4=();
$aa=0;
$bb=0;
$cc=0;
$dd=0;
@mut_pos3=();
@mut_pos4=();
$length=0;
$neg1=0;
$neg2=0;
$n=0;
@header=();


}

exit;
```

```
sub read_mut
{
 my ($mut_type,$CIGAR,$seq,$MD,$mut_pos_ref,$strand)=@_;
 my @mut_pos=();
 my $ref_pos=0;
 my $tag_pos=0;
 my ($regex,$match,$temp);

 if ($CIGAR=~/([0-9]+)S.*[0-9]+M/) {$seq=substr($seq,$1);} # offset soft-clipping
 $CIGAR=~s/[0-9]+H//g; # offset hard-clipping

 while ($CIGAR=~/([0-9]+)([MDI])/g)
 {
  if ($2 eq "M")
  {
   $ref_pos+=$1;
   $tag_pos+=$1;
  }elsif ($2 eq "I")
  {
   {if ($mut_type=~/Ins|all/) {push @$mut_pos_ref,($ref_pos+1);}}
   substr($seq,$tag_pos,$1)="";
   $tag_pos+=$1;
  }else
  {
   {if ($mut_type=~/Del|all/) {push @$mut_pos_ref,($ref_pos+1);}}
   $ref_pos+=$1;
  }
 }

 $ref_pos=0;
 $tag_pos=0;

 while ($MD=~/([0-9]+|[ACGTN]|\^[ACGTN]+)/g)
 {
  $match=$1;
  if ($match=~/[0-9]+/)
  {
   $ref_pos+=$match;
   $tag_pos+=$match;
  }elsif ($match=~/^[ACGTN]$/)
  {
   $ref_pos+=1;
   $temp=substr($seq,$tag_pos,1);
```

```
      if ($strand eq "-")
      {
        $match=transform($match);
        $temp=transform($temp);
      }

      $temp=$match."->".$temp;
      $regex=qr/$temp/;
      $tag_pos+=1;
      if ($mut_type=~$regex) {push @$mut_pos_ref,$ref_pos;}
    }else
    {
      $ref_pos+=length($match)-1;
    }
  }
}

sub transform # negative strand to positive strand
{
  my $base=$_[0];

  if ($base eq "A")
  {
    $base="T";
  }elsif ($base eq "T")
  {
    $base="A";
  }elsif ($base eq "C")
  {
    $base="G";
  }elsif ($base eq "G")
  {
    $base="C";
  }

  return($base);
}

sub sort_alignment
{
  my ($align)=@_;
  my $line1;
  my @a=();
  my @b=();
  my @array=();
  my @sarray=();
  my $key='';
  my $key1='';
  my ($i,$s1);
  my %hash='';
  my %hash1='';


  foreach (@$align)
  {
    $line1= $_;
    @a=split(/\s+/, "$line1");
    @b=split(/\./,"$a[0]");
    #$hash{$a[0]}=$line1;
    if($b[1] eq '')
    {
       $b[1]=0;
    }
    push (@{$hash1{$b[0]}},"$b[1]");
    $key1=$b[0].".".$b[1];
  # print "$key1\n";
    $hash{$key1}=$line1;
  # print "$b[0]\t$b[1]\n";
  }

  @array= @{$hash1{$b[0]}};
  @sarray=sort{$a <=> $b} (@array);
# print "@array\n";
# print "@sarray\n";
  $i=0;

  foreach $s1 (@sarray)
  {
       $key=$b[0].".".$s1;
     #  print "$key\n";
       @$align[$i]=$hash{$key};
       $i = $i +1;
  }


}
```