Supplemental Code

Reconstructing double-stranded DNA fragments on a single molecule level reveals patterns of degradation in ancient samples

Lukas Bokelmann, Isabelle Glocke and Matthias Meyer

Table of Contents

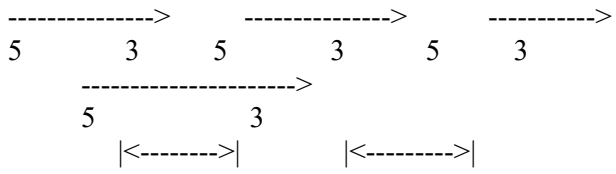
Supplemental code

alignment_distance.pl

```
#!/usr/bin/perl -w
use strict;
use File::Basename;
use Getopt::Long;

my $qual_cutoff = 0;
my $minread = 35;
my $direction = 1;
my $range = 100;
my ($output, $side, $drop);

GetOptions ("quality=s" => \$qual_cutoff,
            "minread=i" => \$minread,
          "range=i" => \$range,
            "drop" => \$drop,
          "output=i" => \$output);

&help unless scalar @ARGV == 1;
&help unless $ARGV[0] =~ /\.bam$/;
if ($output) {
   my $outname = basename($ARGV[0]);
   $outname =~ s/\.bam//;
   open WRITE5, "| samtools view -S -b - >$outname.1st_dist$output.bam" or die "could not write file
$! \n";
   open WRITE3, "| samtools view -S -b - >$outname.2nd_dist$output.bam" or die "could not write
file $! \n";
}

open BAM, "samtools view -hX -F p $ARGV[0] |" or die "could not open file $!\n";

my (%dist_hist, %gap_base, %gap_before, %gap_after);
my ($counter, $interval) = (0, 0);
my $prev_chr = "";
my $prev_end_plus = 0;
my $prev_end_minus = 0;
my ($prev_line_plus, $prev_line_minus);
while (my $line = <BAM>) {
   $counter++; $interval++;
   if ($line =~ /^\@/) {
         if ($output) {
            print WRITE5 $line;
            print WRITE3 $line;
         }
         next;
   }
   if ($interval == 100_000) {
         print STDERR "\r$counter";
         $interval = 0;
   }
   chomp $line;
   my @fields = split /\t/, $line;
   my ($header, $flag, $chr, $start, $mapqual, $cigar, $junk1, $junk2, $junk3, $sequence) = split /\t/,
$line;
```

```
    my $length = length($sequence);
    unless ($drop) {
            next unless $chr =~ /^(\d{1,2}|X)$/;
    }
    next if $length < $minread;
    next if $mapqual < $qual_cutoff;
    my $end = aln_end($start, $cigar, $length);
    if ($flag =~ /r/) {
            if ($chr eq $prev_chr && $prev_end_minus > 0 && $prev_end_minus < $start) {
               my $distance = $start - $prev_end_minus;
               $dist_hist{$distance}++;
               if ($output && $distance == $output) {
               }
            }
            $prev_end_minus = $end;
            $prev_line_minus = $line;
    } else {
            if ($chr eq $prev_chr && $prev_end_plus > 0 && $prev_end_plus < $start) {
               my $distance = $start - $prev_end_plus;
               $dist_hist{$distance}++;
               if ($output && $distance == $output) {
                     print WRITE3 "$line\n";
                     print WRITE5 "$prev_line_plus\n" if $prev_line_plus;
               }
            }
            $prev_end_plus = $end;
            $prev_line_plus = $line;
    }
    $prev_chr = $chr;
}
foreach my $dist (1 .. $range) {
    print "$dist\t", $dist_hist{$dist} || 0, "\n";
}

sub aln_end {
    my ($start, $cigar, $length) = @_;
    my $end;
    if ($cigar =~ /[ID]/) {
       my ($insertions, $deletions, $matches) = (0,0,0);
       my $cigar_backup = $cigar;
       while ($cigar =~ s/(\d+)([MID])//) {
          my ($num, $type) = ($1, $2);
          $insertions += $num if $type eq "I";
          $deletions += $num if $type eq "D";
          $matches += $num if $type eq "M";
       }
       $end = $start + $matches + $deletions - 1;
    } else {
       $end = $start + $length - 1;
    }
    return $end;
}

sub help {
print STDERR "
```

This script computes a histogram of the distance between the 3' end of each sequence and the closest 5' end of another sequence on the same strand. A 1-bp gap corresponds to a distance of 2. Input are sorted alignment files in BAM format produced by BWA. There is a strict requirement for single reads, paired reads are filtered out.



[usage]
./alignment_distance.pl [-options] in.bam
[options]
-quality map quality cutoff [0]
-minread minimal length cutoff [35]
-range range of values to report
-output write out sequences that have another sequence in distance X from their 5' or 3' end respectively.
-drop drop filtering of sequences that are not mapped to the autosomes and chromosome X (recommended when using
a reference genome other than hg19)
[output]
infile.1st_distX.bam first (left) sequence whose 3' end is in distance X to the 5' end of the following sequence
infile.2nd_distX.bam second (right) sequences whose 5' end is in distance X to the 3' of the sequence to the left
screen alignment distance histogram
DEPENDENCIES
This perl script has been successfully used with perl 5 (version 22). It requires 2 perl modules to be installed (File::Basename and Getopt::Long) as well as samtools (successfully used with version 1.3.1).
";
exit;
}

complementary_strands2.pl

```
#!/usr/bin/perl -w

use strict "vars";

use strict "subs";

use File::Basename;

use Getopt::Long;

use List::Util 'shuffle';


my $qual_cutoff = 0;

my $minread = 30;

my $direction = 1;

my $range = 100;

my $mt;
```

```
my $output;
my $control;

GetOptions ("quality=s" => \$qual_cutoff,
"minread=i" => \$minread,
"range=i" => \$range,
"mt" => \$mt,
"control" => \$control,
"output=i" => \$output);

&help unless scalar @ARGV == 1;
&help unless $ARGV[0] =~ /\.bam$/;

my (@seqs_to_analyze, @all_seqs, $bam_header, $counter, $interval, %dist_hist,
%structure_hist, %match_hist, %filehandles);
my ($last_ms, $last_me, $last_ps, $last_pe, $last_chr) = (0, 0, 0, 0);
my ($first_ms, $first_me, $first_ps, $first_pe);

my $outname = basename($ARGV[0]);
$outname =~ s/\.bam//;
open BAM, "samtools view -h $ARGV[0] |" or die "could not open file\n";

&add_sequence;
while (scalar @seqs_to_analyze > 0) {
my $current_seq = shift @seqs_to_analyze;
my ($orientation, $chr, $start, $end, $line, $seqlength) = @{$current_seq};
$counter++, $interval++;
if ($interval == 10_000) {
printf STDERR "\r$counter %20s", "processing chromosome $chr ";
$interval = 0;
}
while ("remove sequences from previous chromosome") {
my $first_seq = shift @all_seqs;
my ($seq_orientation, $seq_chr, $seq_start, $seq_end, $seq_line) = @{$first_seq};
if ($seq_chr ne $chr) {
($last_ps, $last_ms) = (undef, undef);
```

```
next;
}
unshift @all_seqs, $first_seq;
last;
}
while ("keep adding sequences until enough") {
if ($last_chr ne $chr) {
last;
}
if ($last_ps && $last_ms && $last_ps > $end && $last_ms > $end) {
last;
}
&add_sequence || last;
}
my @all_seqs_cleaned;
my ($left_dist_ps, $left_dist_pe, $left_dist_ms, $left_dist_me);
my ($first_pe, $first_me);
while ("searching left and eliminating unused sequences") {
my $next_seq = pop @all_seqs || last;
unshift @all_seqs_cleaned, $next_seq;
my ($seq_orientation, $seq_chr, $seq_start, $seq_end, $seq_line) = @{$next_seq};
$first_pe = $seq_end if $seq_orientation eq "plus";
$first_me = $seq_end if $seq_orientation eq "minus";
next if $seq_line eq $line; #do not compare sequences to themselves
next if $seq_chr ne $chr;
if ($seq_start - $start <= 0) {
if ($seq_orientation eq "plus") {
$left_dist_ps = $seq_start - $start unless defined $left_dist_ps;
} else {
$left_dist_ms = $seq_start - $start unless defined $left_dist_ms;
}
}
if ($seq_end - $end <= 0) {
```

```
if ($seq_orientation eq "plus") {
$left_dist_pe = $seq_end - $end unless defined $left_dist_pe;
} else {
$left_dist_me = $seq_end - $end unless defined $left_dist_me;
}
}
if (defined $left_dist_ps && defined $left_dist_ms && defined $left_dist_pe && defined $left_dist_me) {
last if (defined $first_pe && defined $first_me && $first_pe < $start && $first_me < $start);
}
}
@all_seqs = @all_seqs_cleaned;
my ($right_dist_ps, $right_dist_pe, $right_dist_ms, $right_dist_me);
while ("searching right sequences") {
IT: foreach my $next_seq (@all_seqs) {
my ($seq_orientation, $seq_chr, $seq_start, $seq_end, $seq_line) = @{$next_seq};
next if $seq_line eq $line;
next if $seq_chr ne $chr;
if ($seq_start - $start > 0) {
if ($seq_orientation eq "plus") {
$right_dist_ps = $seq_start - $start unless $right_dist_ps;
} else {
$right_dist_ms = $seq_start - $start unless $right_dist_ms;
}
}
if ($seq_end - $end > 0) {
if ($seq_orientation eq "plus") {
$right_dist_pe = $seq_end - $end unless $right_dist_pe;
} else {
$right_dist_me = $seq_end - $end unless $right_dist_me;
}
}
if (defined $right_dist_ps && defined $right_dist_ms && defined $right_dist_pe && defined $right_dist_me) {
```

```
last IT;
}
}
last;
}
my ($dist_5, $dist_3, $dist_5_control, $dist_3_control);
if ($orientation eq "plus") {
$dist_5 = &store_distance($left_dist_ms, $right_dist_ms, "dist_ps_ms", "dist_5");
$dist_3 = &store_distance($left_dist_me, $right_dist_me, "dist_pe_me", "dist_3");
$dist_5_control = &store_distance($left_dist_ps, $right_dist_ps, "dist_ps_ps",
"dist_5_control");
$dist_3_control = &store_distance($left_dist_pe, $right_dist_pe, "dist_pe_pe",
"dist_3_control");
} elsif ($orientation eq "minus") {
$dist_3 = &store_distance($left_dist_ps, $right_dist_ps, "dist_ms_ps", "dist_3", "NEG");
$dist_5 = &store_distance($left_dist_pe, $right_dist_pe, "dist_me_pe", "dist_5", "NEG");
$dist_3_control = &store_distance($left_dist_ms, $right_dist_ms, "dist_ms_ms",
"dist_3_control", "NEG");
$dist_5_control = &store_distance($left_dist_me, $right_dist_me, "dist_me_me",
"dist_5_control", "NEG");
}
if (defined $output && abs($dist_5) <= $output) {
my $string = "5p-";
$string .= "blunt" if $dist_5 == 0;
$string .= "overhang-${dist_5}bp" if $dist_5 > 0;
$string .= "recessed-" . abs($dist_5) . "bp" if $dist_5 < 0;
my $fhdist = "fh_dist5_$dist_5";
unless (exists $filehandles{$fhdist}) {
open ($fhdist, "| samtools view -bS - >$outname.$string.bam") or die "could not write file\n";
print $fhdist $bam_header;
print $fhdist $line;
$filehandles{$fhdist}++;
} else {
print $fhdist $line;
```

```
}
}
if (defined $output && abs($dist_3) <= $output) {
my $string = "3p-";
$string .= "blunt" if $dist_3 == 0;
$string .= "overhang-" . abs($dist_3) . "bp" if $dist_3 < 0;
$string .= "recessed-" . abs($dist_3) . "bp" if $dist_3 > 0;
my $fhdist = "fh_dist3_$dist_3";
unless (exists $filehandles{$fhdist}) {
open ($fhdist, "| samtools view -bS - >$outname.$string.bam") or die "could not write file\n";
print $fhdist $bam_header;
print $fhdist $line;
$filehandles{$fhdist}++;
} else {
print $fhdist $line;
}
}
if ($control && defined $output && abs($dist_5_control) <= $output) {
my $string = "control_5p-";
$string .= "blunt" if $dist_5_control == 0;
$string .= "overhang-${dist_5_control}bp" if $dist_5_control > 0;
$string .= "recessed-" . abs($dist_5_control) . "bp" if $dist_5_control < 0;
my $fhdist = "fh_dist_5_control_$dist_5_control";
unless (exists $filehandles{$fhdist}) {
open ($fhdist, "| samtools view -bS - >$outname.$string.bam") or die "could not write file\n";
print $fhdist $bam_header;
print $fhdist $line;
$filehandles{$fhdist}++;
} else {
print $fhdist $line;
}
}
```

```
if ($control && defined $output && abs($dist_3_control) <= $output) {
my $string = "control_3p-";
$string .= "blunt" if $dist_3_control == 0;
$string .= "overhang-" . abs($dist_3_control) . "bp" if $dist_3_control < 0;
$string .= "recessed-" . abs($dist_3_control) . "bp" if $dist_3_control > 0;
my $fhdist = "fh_dist3_control_$dist_3_control";
unless (exists $filehandles{$fhdist}) {
open ($fhdist, "| samtools view -bS - >$outname.$string.bam") or die "could not write file\n";
print $fhdist $bam_header;
print $fhdist $line;
$filehandles{$fhdist}++;
} else {
print $fhdist $line;
}
}
$structure_hist{"$dist_5,$dist_3"}++;
my $min_dist = abs($dist_3);
$min_dist = abs($dist_5) if abs($dist_5) < abs ($dist_3);
foreach ($min_dist .. 100) {
$match_hist{$seqlength}{$_}++;
}
$match_hist{$seqlength}{"obs"}++;
}

open HIST, ">$outname.comp_strands.hist.txt" or die "could not write histogram\n";
print HIST join ("\t", qw /Dist 5p_end %5p_end 3p_end %3p_end 5p_end_control
%5p_end_control 3p_end 3p_end_control/), "\n";
foreach my $num (-$range..$range) {
print HIST "$num";
foreach my $type (qw/dist_5 dist_3 dist_5_control dist_3_control/) {
print HIST "\t", $dist_hist{$type}{$num} || 0;
my $percent;
my $obs = $dist_hist{$type}{"obs"};
if ($obs) {
```

```
my $count = $dist_hist{$type}{$num} || 0;
$percent = sprintf "%.3f", $count / $obs * 100;
} else {
$percent = 0;
}
print HIST "\t$percent";
}
print HIST "\n";
}

open STRUC, ">$outname.ends.txt" or die "could not write ends\n";
print STRUC "|5p|\-3p-\t", join ("\t", -100 .. 100), "\n";
foreach my $dist5 (-100 .. 100) {
print STRUC $dist5;
foreach my $dist3 (-100 .. 100) {
print STRUC "\t", $structure_hist{"$dist5,$dist3"} || 0;
}
print STRUC "\n";
}

open MATCH, ">$outname.matching_hist.txt" or die "could not write matching stats\n";
print MATCH join ("\t", "Length", 0 .. 30), "\n";
foreach my $seqlength (30 .. 100) {
print MATCH $seqlength;
foreach my $min_dist (0 .. 50) {
my $num = $match_hist{$seqlength}{$min_dist} || 0;
my $obs = $match_hist{$seqlength}{"obs"} || 0;
if ($obs) {
my $fraction = sprintf ("%.5f", $num / $obs);
print MATCH "\t$fraction";
} else {
print MATCH "\tNA";
}
}
```

```
print MATCH "\n";
}

sub store_distance {
my $distance;
my ($left_dist, $right_dist, $key1, $key2, $negative) = @_;
if (defined $left_dist && defined $right_dist && abs($left_dist) == abs($right_dist)) {
my @tmp = shuffle ($left_dist, $right_dist);
$distance = shift @tmp;
} elsif (defined $left_dist && defined $right_dist) {
$distance = $left_dist;
$distance = $right_dist if abs($right_dist) < abs($left_dist);
} elsif (defined $left_dist || defined $right_dist) {
$distance = $left_dist;
$distance = $right_dist if defined $right_dist;
} else {
warn "OH: $left_dist, $right_dist\n";
}
$distance = -$distance if $negative;
$dist_hist{$key1}{$distance}++;
$dist_hist{$key1}{"obs"}++;
$dist_hist{$key2}{$distance}++;
$dist_hist{$key2}{"obs"}++;
return $distance;
}

sub add_sequence {
while (1) {
my $line = <BAM> || return 0;
if ($line =~ /^\@/) {
$bam_header .= "$line";
next;
}
my ($header, $flag, $chr, $start, $mapqual, $cigar, $junk1, $mate_start, $tlen, $sequence) =
split /\t/, $line;
```

```
$flag = decode_flag($flag);
my $length = length($sequence);
unless ($mt) {
next unless $chr =~ /^(chr)?(\d{1,2}|X)$/;
}
next if $length < $minread;
next if $mapqual < $qual_cutoff;
next if $flag =~ /u/i;
my $end;
if ($flag =~ /p/) {
next unless $flag =~ /1/; ##only ever look at the first read in a pair
next if $tlen > 2_000;
next if $tlen < 30;
unless ($flag =~ /r/) { ##read 1 on plus strand
# die "unexpected paired read orientation\n:FLAG:$flag LINE:$line\n" if $tlen < 0;
next if $tlen < 0;
$end = $start + $tlen - 1;
} else { ##read 1 on minus strand
next if $tlen > 0;
# warn "unexpected reverse read orientation:\nFLAG:$flag LINE:$line" if $tlen > 0;
$start = $mate_start;
$end = $mate_start + abs($tlen) - 1;
}
} else { #single reads
$end = &aln_end($start, $cigar, $length);
}
$last_chr = $chr;
my $orientation;
if ($flag =~ /r/) {
$orientation = "minus";
$last_ms = $start;
} else {
$orientation = "plus";
```

```
$last_ps = $start;
}
my @seq = ($orientation, $chr, $start, $end, $line, $length);
push (@seqs_to_analyze, \@seq);
push (@all_seqs, \@seq);
return 1;
}
}

sub aln_end {
my ($start, $cigar, $length) = @_;
my $end;
if ($cigar =~ /[ID]/) {
my ($insertions, $deletions, $matches) = (0,0,0);
my $cigar_backup = $cigar;
while ($cigar =~ s/(\d+)([MID])//) {
my ($num, $type) = ($1, $2);
$insertions += $num if $type eq "I";
$deletions += $num if $type eq "D";
$matches += $num if $type eq "M";
}
$end = $start + $matches + $deletions - 1;
} else {
$end = $start + $length - 1;
}
return $end;
}

sub decode_flag {
my $hex = shift(@_);
my $bin = sprintf "%b", $hex;
my $literal = "qdfs21RrUuPp";
my $string;
foreach my $pos (-length($bin) .. -1) {
```

```
$string .= substr $literal, $pos, 1 if (substr $bin, $pos, 1);
}
return $string || "_";
}

sub help {
print STDERR "
```

This script computes a histogram of the distance between the 5' and 3' end of each strand to 3' and 5' end of the closest complementary strand. It also computes the closest distance between 5'/3' ends that are on the same strand for comparison. If desired, strands that are in distance X to the closest 3' or 5' end of the complementary strand can be written to separate bam files. Only alignments to the autosomes or chromosome X are analyzed. Input are sorted alignment files in BAM format.

CAUTION: For mate pairs, only the first read of the pair is currently written to bam output, but histograms are computed correctly.

```
5' 3' 5' 3' 5' 3' 5' 3'
------Seq1--------> --------Seq2-------> -----Seq3-------> -----Seq4----->
<---------------------- <----------- <-------- <--------------
3' 5' 3' 5' 3' 5' 3' 5'

5' distance: -2 +3 -14 0
<> <-> <------------> |

3' distance: +3 -5 +21 0
<-> <---> <------------------->

5'control +36 +36 +22 +22
distance <---------------------------------> <-------------------->
```

[usage]

./complementary_strands2.pl [-option] in.bam

[options]

-quality map quality cutoff [0]

-minread minimal length cutoff [30]

-range range of values to report [100]

-output write out sequences that have another sequence in distance X from their 5' or 3' end respectively.

-control write out sequences that have another sequence in distance X from their 5' or 3' end respectively IN THE SAME ORIENTATION

-mt analyze mtDNA (drop mapping restrictions to X and autosomes; also required when using a reference genome other than hg19)

[output]

-in.hist.txt table with histogram of closest distances, in number of observations and fraction of sequences

-in.ends.txt table showing the observed combinations between 5' and 3' molecule structures

-in.matching_hist.txt table providing the fraction of sequences with a maximum distance of X between complementary strands, binned by sequence length

-bam files in.{5p/3p}-{overhang/recessed/blunt}-[X]bp.bam

DEPENDENCIES

This perl script has been successfully used with perl 5 (version 22). It requires 3 perl modules to be installed (File::Basename, Getopt::Long and List::Util) as well as samtools (successfully used with version 1.3.1).

";

exit;

}

substitution_patterns2.pl

```
#!/usr/bin/perl -w

use strict;

use Getopt::Long;

use File::Basename;


my $quality = 0;

my $minread = 30;

my $maxread = 1000;

my $help;

my $graphic = 0;

my $parameter = "-F pu";

my $unpaired;

my $subset;

my $xrange = 30;

my $yrange;

my $report_interval = 100_000;
```

```
my $refgenome;
my %out;
my %out_ref;

GetOptions ("quality=s" => \$quality,
"minread=i" => \$minread,
"maxread=i" => \$maxread,
"unpaired" => \$unpaired,
"subset=i" => \$subset,
"graphical" => \$graphic,
"xrange=i" => \$xrange,
"yrange=f" => \$yrange,
"reference=s" => \$refgenome,
"reportinterval=i" => \$report_interval,
"help" => \$help);

my @infiles = @ARGV;
$parameter = "-F u" if $unpaired;

&help if $help; &help if scalar @infiles < 1;

my (%data, %cpg, %noncpg, %cpg_ga, %noncpg_ga, %dinuc_left, %dinuc_right);
FILE:foreach my $file (@infiles) {
print STDERR "processing $file\n";
open READ, "samtools view $parameter $file |" or die "could not read file $file\n";
my $outname = basename($file);
$outname =~ s/\.bam$// or die "input file $file is not a bam file\n";
$outname =~ s/\.reference_aln$//;
$outname .= ".L${minread}-${maxread}_MQ$quality";
if (-e "$outname.plots.pdf") {
print STDERR "plots for $outname already exist, skipping...\n"; next;
} elsif (-e "$outname.5p_substitutions.txt") {
print STDERR "tables for $outname already exist, skipping...\n"; next;
}
my (%summary, $counter, $interval, $report_frequency);
```

```
%data = (); %cpg = (); %noncpg = (); %cpg_ga = (); %noncpg_ga = (); %dinuc_left = (); %dinuc_right = ();
while (my $line = <READ>) {
$counter++; $interval++;$report_frequency++;
if ($interval == 1_000) {
print STDERR "\r$counter";
$interval = 0;
}
if ($report_frequency == $report_interval) {
&report($outname);
$report_frequency = 0;
}
if ($subset) {
if ($counter >= $subset) {
&report($outname);
next FILE;
}
}
chomp $line;
$summary{"all"}++;
my @tmp = split /\t/, $line;
my ($sequence, $locus, $aln_start, $mapqual, $flag, $cigar) = ("\U$tmp[9]", $tmp[2], $tmp[3], $tmp[4], $tmp[1], $tmp[5]);
my $orientation = decode_flag($flag);
next if $locus eq "*";
next if length($sequence) > $maxread;
next if length($sequence) < $minread;
next if $cigar =~ /[SH]/; #remove clipped sequences
my $md;
foreach my $element (@tmp) {
if ($element =~ /MD:[ZA]:(\S+)$/) {
$md = $1;
}
}
```

```
die "no MD field seen in\n $line\n" unless $md;
if ($md =~ /\^[RY]/) {
warn "I have seen a weird MD field: $md\nignoring sequence...\n";
next;
}
$summary{"allL"}++;
if ($quality) {
next if $mapqual < $quality;
}
$summary{"allLQ"}++;
next if $locus =~ /\*/; #unmapped read
my $refseq;
($refseq, $sequence) = &buildref($sequence, $cigar, $md);
my $aln_seq_length = length ($sequence);
$summary{"allLQref"}++;
my $extended_refseq;
if ($refgenome) {
my $aln_end = $aln_start + $aln_seq_length - 1;
my ($retrieval_start, $retrieval_end) = (($aln_start - 50), ($aln_end + 50));
my $refseq_faidx;
if ($retrieval_start > 0) {
open REF, "samtools faidx $refgenome \"${locus}\":${retrieval_start}-${retrieval_end} |";
while (my $line = <REF>) {
chomp $line;
next if $line =~ /^>/;
$refseq_faidx .= uc($line);
$extended_refseq = $refseq_faidx unless length($refseq_faidx) != ($aln_seq_length + 100);
}
}
}
unless ($extended_refseq) {
$extended_refseq = "N" x 50 . $refseq . "N" x 50;
}
```

```
$refseq = $extended_refseq;
next unless $refseq =~ /[ACGT]/; #There was a case with only N's in the reference
if ($orientation =~ /r/) {
$sequence = reverse($sequence);
$sequence =~ tr/ACGT/TGCA/;
$refseq = reverse($refseq);
$refseq =~ tr/ACGT/TGCA/;
$summary{"allLQref-"}++;
} else {
$summary{"allLQref+"}++;
}
my @refseq = split "", $refseq;
my @seq = split "", $sequence;
my $position;
my $previousrefbase = "-";
my $previousbase = "-";
for ($position = -50; $position < 0; $position++) {# chew on 5' reference sequence
my $refbase = shift @refseq or die "refbase missing?\n";
if ($refbase =~ /[ACGT]/) {
$data{"5p"}{$position}{"$refbase"}++;
$data{"5p"}{$position}{"X"}++;
}
my $dinuc_ref = $previousrefbase . $refbase;
if ($refbase =~ /[ACGT]/ && $previousrefbase =~ /[ACGT]/) {
$dinuc_left{"5p"}{$position}{$dinuc_ref}++;
$dinuc_left{"5p"}{$position}{"XX"}++;
}
$previousrefbase = $refbase;
}
my $position_3p = - length ($sequence);
my $dinuc_position_3p = $position_3p - 1;
my %simcheck;
$position = -1;
```

```
while (my $base = shift @seq) {
$position++; $position_3p++; $dinuc_position_3p++;
my $refbase = shift @refseq or die "refbase missing here\n";
$simcheck{"match"}++ if $base eq $refbase;
$simcheck{"all"}++;
if ($refbase =~ /[ACGT-]/ && $base =~ /[ACGT-]/) {
$data{"5p"}{$position}{"$refbase"}++;
$data{"5p"}{$position}{"X"}++;
$data{"5p"}{$position}{"$refbase$base"}++;
$data{"5p"}{$position}{"${refbase}X"}++;
$data{"3p"}{$position_3p}{"$refbase"}++;
$data{"3p"}{$position_3p}{"X"}++;
$data{"3p"}{$position_3p}{"$refbase$base"}++;
$data{"3p"}{$position_3p}{"${refbase}X"}++;
my $nextrefbase = $refseq[0];
if ($refbase =~ /[ACGT]/ && $nextrefbase =~ /[ACGT]/) {
if ($refbase eq "C" && $nextrefbase eq "G") {
$cpg{"5p"}{$position}{"$refbase$base"}++;
$cpg{"5p"}{$position}{"${refbase}X"}++;
$cpg{"3p"}{$position_3p}{"$refbase$base"}++;
$cpg{"3p"}{$position_3p}{"${refbase}X"}++;
}
if ($refbase eq "C" && $nextrefbase ne "G") {
$noncpg{"5p"}{$position}{"$refbase$base"}++;
$noncpg{"5p"}{$position}{"${refbase}X"}++;
$noncpg{"3p"}{$position_3p}{"$refbase$base"}++;
$noncpg{"3p"}{$position_3p}{"${refbase}X"}++;
}
}
$previousrefbase = " " unless $previousrefbase;
if ($refbase =~ /[ACGT]/ && $previousrefbase =~ /[ACGT]/) {
if ($refbase eq "G" && $previousrefbase eq "C") {
$cpg_ga{"5p"}{$position}{"$refbase$base"}++;
```

```
$cpg_ga{"5p"}{$position}{"X${refbase}"}++;
$cpg_ga{"3p"}{$position_3p}{"$refbase$base"}++;
$cpg_ga{"3p"}{$position_3p}{"X${refbase}"}++;
}
if ($refbase eq "G" && $previousrefbase ne "C") {
$noncpg_ga{"5p"}{$position}{"$refbase$base"}++;
$noncpg_ga{"5p"}{$position}{"X${refbase}"}++;
$noncpg_ga{"3p"}{$position_3p}{"$refbase$base"}++;
$noncpg_ga{"3p"}{$position_3p}{"X${refbase}"}++;
}
}
}
my $dinuc_ref_left = $previousrefbase . $refbase;
$previousbase = $previousrefbase if $position == 0;
my $dinuc_bases_left = $previousbase . $base;
if ($dinuc_ref_left =~ /^[ACGT]{2}$/) {
$dinuc_left{"5p"}{$position}{$dinuc_ref_left}++;
$dinuc_left{"5p"}{$position}{"XX"}++;
$dinuc_left{"3p"}{$dinuc_position_3p}{$dinuc_ref_left}++;
$dinuc_left{"3p"}{$dinuc_position_3p}{"XX"}++;
if ($dinuc_bases_left =~ /^[ACGT-]{2}$/) {
$dinuc_left{"5p"}{$position}{"$dinuc_ref_left$dinuc_bases_left"}++;
$dinuc_left{"5p"}{$position}{"${dinuc_ref_left}XX"}++;
$dinuc_left{"3p"}{$dinuc_position_3p}{"$dinuc_ref_left$dinuc_bases_left"}++;
$dinuc_left{"3p"}{$dinuc_position_3p}{"${dinuc_ref_left}XX"}++;
}
}
$previousrefbase = $refbase;
$previousbase = $base;
}
print "HERE: $line\n" unless $simcheck{"all"} > 0;
my $identity = ($simcheck{"match"} || 0) / $simcheck{"all"} *100;
$summary{"bad_alignment"}++ if $identity < 80;
```

```
for ($position_3p = 1; $position_3p < 51; $position_3p++) { #chew on 3' reference sequence
my $refbase = shift @refseq or warn "weird $cigar...\n$sequence\n$refseq\n";
if ($refbase =~ /[ACGT]/) {
$data{"3p"}{$position_3p}{"$refbase"}++;
$data{"3p"}{$position_3p}{"X"}++;
if ($position_3p == 1) {
$dinuc_left{"3p"}{0}{"$previousrefbase$refbase$previousbase$refbase"}++;
$dinuc_left{"3p"}{0}{"$previousrefbase${refbase}XX"}++;
}
}
my $dinuc_position_3p = $position_3p -1;
my $dinuc_ref_left = $previousrefbase . $refbase;
if ($dinuc_ref_left =~ /[ACGT]{2}/) {
$dinuc_left{"3p"}{$dinuc_position_3p}{"$dinuc_ref_left"}++;
$dinuc_left{"3p"}{$dinuc_position_3p}{"XX"}++;
}
}
if (scalar @refseq > 0) {
print "\n@refseq\n";
print "$refseq\n$sequence\n";
die;
}
die "reference sequence @refseq left\n" if scalar @refseq > 0;
}
print STDERR "all\t", $summary{"all"} || 0, "\n";
print STDERR "allL\t", $summary{"allL"} || 0, "\n";
print STDERR "allLQ\t", $summary{"allLQ"} || 0, "\n";
print STDERR "allLQref\t", $summary{"allLQref"} || 0, "\n";
print STDERR "allLQref+\t", $summary{"allLQref+"} || 0, "\n";
print STDERR "allLQref-\t", $summary{"allLQref-"} || 0, "\n";
print STDERR "bad_align\t", $summary{"bad_alignment"} || 0, "\n";
&report($outname);
&plot($outname) if $graphic;
```

```
}
sub buildref {
my ($sequence, $cigar, $md) = @_;
my $aln_sequence;
while ($cigar =~ s/^(\d+)([MID])//) {
my ($number, $type) = ($1, $2);
next if $number == 0;
if ($type eq "M") {
foreach (1..$number) {
$sequence =~ s/^(.)// or die "parsing error in cigar field\n";
$aln_sequence .= $1;
}
} elsif ($type eq "I") {
foreach (1..$number) {
$sequence =~ s/^.// or die "parsing error2 in cigar field\n";
}
} elsif ($type eq "D") {
foreach (1..$number) {
$aln_sequence .= "-";
}
}
}
die "parsing error with sequence: $sequence\n" if length $sequence > 0;
die "parsing error cigar field: $cigar\n" unless length $cigar == 0;
my @source = split "", $aln_sequence;
my $aln_reference;
until (length $md == 0) {
if ($md =~ s/^(\d+)//) {
my $number = $1;
next if $number == 0;
foreach (1..$number) {
my $base = shift @source;
$aln_reference .= $base;
```

```
}
} elsif ($md =~ s/^([ABCGTRYSKMWNHwnryskmbh])//) {
my $base = $1;
$base = "N" if $base =~ /[RY]/i;
$aln_reference .= $base;
my $junk = shift @source;
} elsif ($md =~ s/^\^([ACGTNnk]+)//) {
my $bases = $1;
my @bases = split "", $bases;
my $number = scalar @bases;
foreach (1..$number) {
my $junk = shift @source;
}
$aln_reference .= $bases;
} else {
die "weird residue in MD field: $md\n";
}
}
if (length($aln_reference) != length($aln_sequence)) {
die "length different:\nref: $aln_reference\nseq: $aln_sequence\n\n";
}
return ($aln_reference, $aln_sequence);
}

sub report {
my $outname = shift @_;
my @combinations = qw(AC AG AT CA CG CT GA GC GT TA TC TG A- C- G- T-);
foreach my $position (-50 .. 50) {
foreach my $pattern ("A", "C", "G", "T") {
$out{"5p_ref_$pattern"}{$position}{"num"} = $data{"5p"}{$position}{"$pattern"} || 0;
$out{"5p_ref_$pattern"}{$position}{"all"} = $data{"5p"}{$position}{"X"};
$out{"3p_ref_$pattern"}{$position}{"num"} = $data{"3p"}{$position}{"$pattern"} || 0;
$out{"3p_ref_$pattern"}{$position}{"all"} = $data{"3p"}{$position}{"X"};
```

```
}
}
foreach my $position (0 .. 50) {
foreach my $pattern (@combinations) {
$out{"5p_subs_$pattern"}{$position}{"num"} = $data{"5p"}{$position}{"$pattern"} || 0;
$pattern =~ m/(^.)/ or die "no";
my $refbase = $1;
$out{"5p_subs_$pattern"}{$position}{"all"} = $data{"5p"}{$position}{"${refbase}X"};
}
$out{"5p_subs_cpg"}{$position}{"num"} = $cpg{"5p"}{$position}{"CT"} || 0;
$out{"5p_subs_cpg"}{$position}{"all"} = $cpg{"5p"}{$position}{"CX"};
$out{"5p_subs_non_cpg"}{$position}{"num"} = $noncpg{"5p"}{$position}{"CT"} || 0;
$out{"5p_subs_non_cpg"}{$position}{"all"} = $noncpg{"5p"}{$position}{"CX"};
$out{"5p_subs_cpg_ga"}{$position}{"num"} = $cpg_ga{"5p"}{$position}{"GA"} || 0;
$out{"5p_subs_cpg_ga"}{$position}{"all"} = $cpg_ga{"5p"}{$position}{"XG"};
$out{"5p_subs_non_cpg_ga"}{$position}{"num"} = $noncpg_ga{"5p"}{$position}{"GA"} ||
0;
$out{"5p_subs_non_cpg_ga"}{$position}{"all"} = $noncpg_ga{"5p"}{$position}{"XG"};
foreach my $base ("A", "C", "G", "T") {
$out_ref{"5p_subs_ref"}{$position}{$base} = $data{"5p"}{$position}{"${base}X"} || 0;
}
}
foreach my $position (-50 .. 0) {
foreach my $pattern (@combinations) {
$out{"3p_subs_$pattern"}{$position}{"num"} = $data{"3p"}{$position}{"$pattern"} || 0;
$pattern =~ m/(^.)/ or die "no";
my $refbase = $1;
$out{"3p_subs_$pattern"}{$position}{"all"} = $data{"3p"}{$position}{"${refbase}X"};
}
$out{"3p_subs_cpg"}{$position}{"num"} = $cpg{"3p"}{$position}{"CT"} || 0;
$out{"3p_subs_cpg"}{$position}{"all"} = $cpg{"3p"}{$position}{"CX"};
$out{"3p_subs_non_cpg"}{$position}{"num"} = $noncpg{"3p"}{$position}{"CT"} || 0;
$out{"3p_subs_non_cpg"}{$position}{"all"} = $noncpg{"3p"}{$position}{"CX"};
```

```
$out{"3p_subs_cpg_ga"}{$position}{"num"} = $cpg_ga{"3p"}{$position}{"GA"} || 0;
$out{"3p_subs_cpg_ga"}{$position}{"all"} = $cpg_ga{"3p"}{$position}{"XG"};
$out{"3p_subs_non_cpg_ga"}{$position}{"num"} = $noncpg_ga{"3p"}{$position}{"GA"} ||
0;
$out{"3p_subs_non_cpg_ga"}{$position}{"all"} = $noncpg_ga{"3p"}{$position}{"XG"};
foreach my $base ("A", "C", "G", "T") {
$out_ref{"3p_subs_ref"}{$position}{"$base"} = $data{"3p"}{$position}{"${base}X"} || 0;
}
}
&binom;
open WRITEP5, ">$outname.5p_refbase_composition.txt" or die "could not write report\n";
print WRITEP5 join("\t", "#pos", "A", "C", "G", "T", "A_low", "A_high", "C_low",
"C_high", "G_low", "G_high", "T_low", "T_high"), "\n";
open WRITEP3, ">$outname.3p_refbase_composition.txt";
print WRITEP3 join("\t", "#pos", "A", "C", "G", "T", "A_low", "A_high", "C_low",
"C_high", "G_low", "G_high", "T_low", "T_high"), "\n";
foreach my $position (-50 .. 50) {
print WRITEP5 "$position\t";
print WRITEP3 "$position\t";
foreach my $pattern ("A", "C", "G", "T") {
print WRITEP5 $out{"5p_ref_$pattern"}{$position}{"fraction"}, "\t";
print WRITEP3 $out{"3p_ref_$pattern"}{$position}{"fraction"}, "\t";
}
foreach my $pattern ("A", "C", "G", "T") {
print WRITEP5 $out{"5p_ref_$pattern"}{$position}{"low"}, "\t";
print WRITEP5 $out{"5p_ref_$pattern"}{$position}{"high"}, "\t";
print WRITEP3 $out{"3p_ref_$pattern"}{$position}{"low"}, "\t";
print WRITEP3 $out{"3p_ref_$pattern"}{$position}{"high"}, "\t";
}
print WRITEP5 "\n";
print WRITEP3 "\n";
}
open SUBS5, ">$outname.5p_substitutions.txt";
print SUBS5 join("\t", "#pos", @combinations, "", "CG->TG", "C[ACT]->T[ACT]", "", "CG-
```

```
>CA", "[ATG]G->[ATG]A", "", "refA","refC", "refG", "refT", "CT_95L", "CT_95H", "CG->TG_95L", "CG->TG_95H", "C[ACT]->T[ACT]_95L", "C[ACT]->T[ACT]_95H", "\n");
foreach my $position (0 .. 50) {
print SUBS5 "$position\t";
foreach my $pattern (@combinations) {
print SUBS5 $out{"5p_subs_$pattern"}{$position}{"fraction"}, "\t";
}
print SUBS5 "\t", $out{"5p_subs_cpg"}{$position}{"fraction"}, "\t";
print SUBS5 $out{"5p_subs_non_cpg"}{$position}{"fraction"}, "\t";
print SUBS5 "\t", $out{"5p_subs_cpg_ga"}{$position}{"fraction"}, "\t";
print SUBS5 $out{"5p_subs_non_cpg_ga"}{$position}{"fraction"}, "\t";
print SUBS5 "\t", $out_ref{"5p_subs_ref"}{$position}{"A"}, "\t";
print SUBS5 $out_ref{"5p_subs_ref"}{$position}{"C"}, "\t";
print SUBS5 $out_ref{"5p_subs_ref"}{$position}{"G"}, "\t";
print SUBS5 $out_ref{"5p_subs_ref"}{$position}{"T"}, "\t";
print SUBS5 $out{"5p_subs_CT"}{$position}{"low"}, "\t";
print SUBS5 $out{"5p_subs_CT"}{$position}{"high"}, "\t";
print SUBS5 $out{"5p_subs_cpg"}{$position}{"low"}, "\t";
print SUBS5 $out{"5p_subs_cpg"}{$position}{"high"}, "\t";
print SUBS5 $out{"5p_subs_non_cpg"}{$position}{"low"}, "\t";
print SUBS5 $out{"5p_subs_non_cpg"}{$position}{"high"}, "\n";
}
open SUBS3, ">$outname.3p_substitutions.txt";
print SUBS3 join("\t", "#pos", @combinations, "", "CG->TG", "C[ACT]->T[ACT]", "", "CG->CA", "[ATG]G->[ATG]A", "", "refA","refC", "refG", "refT", "CT_95L", "CT_95H", "CG->TG_95L", "CG->TG_95H", "C[ACT]->T[ACT]_95L", "C[ACT]->T[ACT]_95H", "\n");
foreach my $position (-50 .. 0) {
print SUBS3 "$position\t";
foreach my $pattern (@combinations) {
print SUBS3 $out{"3p_subs_$pattern"}{$position}{"fraction"}, "\t";
}
print SUBS3 "\t", $out{"3p_subs_cpg"}{$position}{"fraction"}, "\t";
print SUBS3 $out{"3p_subs_non_cpg"}{$position}{"fraction"}, "\t";
print SUBS3 "\t", $out{"3p_subs_cpg_ga"}{$position}{"fraction"}, "\t";
```

```
print SUBS3 $out{"3p_subs_non_cpg_ga"}{$position}{"fraction"}, "\t";
print SUBS3 "\t", $out_ref{"3p_subs_ref"}{$position}{"A"}, "\t";
print SUBS3 $out_ref{"3p_subs_ref"}{$position}{"C"}, "\t";
print SUBS3 $out_ref{"3p_subs_ref"}{$position}{"G"}, "\t";
print SUBS3 $out_ref{"3p_subs_ref"}{$position}{"T"}, "\t";
print SUBS3 $out{"3p_subs_CT"}{$position}{"low"}, "\t";
print SUBS3 $out{"3p_subs_CT"}{$position}{"high"}, "\t";
print SUBS3 $out{"3p_subs_cpg"}{$position}{"low"}, "\t";
print SUBS3 $out{"3p_subs_cpg"}{$position}{"high"}, "\t";
print SUBS3 $out{"3p_subs_non_cpg"}{$position}{"low"}, "\t";
print SUBS3 $out{"3p_subs_non_cpg"}{$position}{"high"}, "\n";
}
my (@dinuc_combinations_left, @dinuc_combinations_right);
foreach my $base1 (qw (A C G T)) {
foreach my $base2 (qw (A C G T)) {
my $refcomb = $base1 . $base2;
foreach my $mut (qw (A C G T)) {
push (@dinuc_combinations_left, "$refcomb$base1$mut") unless $mut eq $base2;
}
}
}
foreach my $base1 (qw (A C G T)) {
foreach my $base2 (qw (A C G T)) {
my $refcomb = $base1 . $base2;
foreach my $mut (qw (A C G T)) {
push (@dinuc_combinations_right, "$refcomb$mut$base2") unless $mut eq $base1;
}
}
}
open DINUCP5, ">$outname.5p_dinucleotide_refbase_composition.txt" or die "could not
write dinuc 5p refcomp\n";
print DINUCP5 join ("\t", "", qw(AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG
TT), "\n");
foreach my $position (-50 .. 50) {
```

```
print DINUCP5 "$position\t";
foreach my $pattern (qw(AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT)) {
my $number = $dinuc_left{"5p"}{$position}{$pattern} || 0;
my $all = $dinuc_left{"5p"}{$position}{"XX"};
if ($all) {
my $fraction = $number / $all;
print DINUCP5 "$fraction\t";
} else {
print DINUCP5 "0\t";
}
}
print DINUCP5 "\n";
}
close DINUCP5;
open DINUCP3, ">$outname.3p_dinucleotide_refbase_composition.txt" or die "could not write dinuc 5p refcomp\n";
print DINUCP3 join ("\t", "", qw(AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT), "\n");
foreach my $position (-50 .. 50) {
print DINUCP3 "$position\t";
foreach my $pattern (qw(AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT)) {
my $number = $dinuc_left{"3p"}{$position}{$pattern} || 0;
my $all = $dinuc_left{"3p"}{$position}{"XX"};
if ($all) {
my $fraction = $number / $all;
print DINUCP3 "$fraction\t";
} else {
print DINUCP3 "0\t";
}
}
print DINUCP3 "\n";
}
close DINUCP3;
```

```
open DNSUBS5, ">$outname.5p_dinucleotide_substitutions.txt";
print DNSUBS5 join("\t", "#pos", @dinuc_combinations_left, @dinuc_combinations_right,
"\n");
foreach my $position (0 .. 50) {
my $position_right = $position + 1; ###TBD
print DNSUBS5 "$position\t";
foreach my $pattern (@dinuc_combinations_left) {
my $number = $dinuc_left{"5p"}{$position}{$pattern} || 0;
$pattern =~ m/(^..)/ or die "no";
my $refbase = $1;
my $all = $dinuc_left{"5p"}{$position}{"${refbase}XX"} || 0;
if ($all) {
my $fraction = $number / $all;
print DNSUBS5 "$fraction\t";
} else {
print DNSUBS5 "0\t";
}
}
foreach my $pattern (@dinuc_combinations_right) {
my $number = $dinuc_left{"5p"}{$position_right}{$pattern} || 0;
$pattern =~ m/(^..)/ or die "no";
my $refbase = $1;
my $all = $dinuc_left{"5p"}{$position_right}{"${refbase}XX"} || 0;
if ($all) {
my $fraction = $number / $all;
print DNSUBS5 "$fraction\t";
} else {
print DNSUBS5 "0\t";
}
}
print DNSUBS5 "\n";
}
close DNSUBS5;
```

```
open DNSUBS3, ">$outname.3p_dinucleotide_substitutions.txt";
print DNSUBS3 join("\t", "#pos", @dinuc_combinations_left, @dinuc_combinations_right,
"\n");
foreach my $position (-50 .. 0) {
my $position_left = $position - 1;
print DNSUBS3 "$position\t";
foreach my $pattern (@dinuc_combinations_left) {
my $number = $dinuc_left{"3p"}{$position_left}{$pattern} || 0;
$pattern =~ m/(^..)/ or die "no";
my $refbase = $1;
my $all = $dinuc_left{"3p"}{$position_left}{"${refbase}XX"} || 0;
if ($all) {
my $fraction = $number / $all;
print DNSUBS3 "$fraction\t";
} else {
print DNSUBS3 "0\t";
}
}
foreach my $pattern (@dinuc_combinations_right) {
my $number = $dinuc_left{"3p"}{$position}{$pattern} || 0;
$pattern =~ m/(^..)/ or die "no";
my $refbase = $1;
my $all = $dinuc_left{"3p"}{$position}{"${refbase}XX"} || 0;
if ($all) {
my $fraction = $number / $all;
print DNSUBS3 "$fraction\t";
} else {
print DNSUBS3 "0\t";
}
}
print DNSUBS3 "\n";
}
close DNSUBS3;
```

```
}

sub binom {
open BINOM, ">binom.temp.R" or die "Could not write binom.temp.R\n";
print BINOM '#!/usr/bin/r', "\n";
print BINOM "#args=(commandArgs(TRUE))\n";
print BINOM 'a=as.numeric(strsplit(argv[1],"/")[[1]])', "\n";
my @input;
my $counter = 1;
foreach my $type (sort {$a cmp $b} keys %out) {
foreach my $position (sort {$a <=> $b} keys %{$out{$type}}) {
unless ($out{$type}{$position}{"all"}) {
# print "OUT: $type $position ", $out{$type}{$position}{"all"}, " \n";####debug
$out{$type}{$position}{"fraction"} = "NA";
$out{$type}{$position}{"low"} = "NA";
$out{$type}{$position}{"high"} = "NA";
next;
}
die "WTF $type $position\n" unless exists $out{$type}{$position}{"num"};
# print "$type $position\n";####debug
my $num = $out{$type}{$position}{"num"} || 0;
my $all = $out{$type}{$position}{"all"};
my $fraction = sprintf ("%.3f", $num / $all);
$out{$type}{$position}{"fraction"} = $fraction;
push (@input, $num, $all);
print BINOM 'cat(binom.test(a[', $counter , '],a[', ($counter + 1) , '])$conf.int, "\n")', "\n";
$counter += 2;
}
}
close BINOM;
system "chmod u+x binom.temp.R";
my $command = "./binom.temp.R ". join ("/", @input) . " >binom.temp.out";
system $command;
open BINOREAD, "binom.temp.out" or die "could not read binom temp\n";
```

```
foreach my $type (sort {$a cmp $b} keys %out) {
foreach my $position (sort {$a <=> $b} keys %{$out{$type}}) {
next unless $out{$type}{$position}{"all"};
my $result = <BINOREAD>;
chomp $result;
my ($low, $high) = split " ", $result;
$low = sprintf ("%.3f", $low);
$high = sprintf ("%.3f", $high);
$out{$type}{$position}{"low"} = $low;
$out{$type}{$position}{"high"} = $high;
my $num = $out{$type}{$position}{"num"} || 0;
my $all = $out{$type}{$position}{"all"} || 0;
push (@input, $num, $all);
$counter += 2;
}
}
unlink ("binom.temp.R");
unlink ("binom.temp.out");
}

sub plot {
my $outname = shift (@_);
foreach my $end (5, 3) {
my $gnu_xrange;
if ($end == 5) {
$gnu_xrange = "[0:$xrange]";
} else {
$gnu_xrange = "[-$xrange:0]";
}
open GNUTEMP, ">gnutemp.delme" or die "could not write temp file for gnuplot";
print GNUTEMP
"set terminal pdf
set output '$outname.${end}p_substitutions.pdf'
```

```
set title (\"${end}p substitution plot \\n$outname\")
set xlabel 'Distance from ${end}` end'
set ylabel 'Substitution frequencies'
set xtic 5
set grid
set xrange $gnu_xrange
set style fill transparent solid 0.1 noborder
ShadecolorCT = '#e56b5d'
plot '$outname.${end}p_substitutions.txt' using 1:7 title 'C->T' with linespoints lt 1 lw 1 lc rgb 'red', \\
" using 1:26:27 with filledcurve fc rgb ShadecolorCT notitle, \\
'$outname.${end}p_substitutions.txt' using 1:8 title 'G->A' with linespoints lt 1 lw 1 lc rgb 'black', \\
'$outname.${end}p_substitutions.txt' using 1:2 title 'A->C' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:3 title 'A->G' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:4 title 'A->T' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:5 title 'C->A' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:6 title 'C->G' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:9 title 'G->C' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:10 title 'G->T' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:11 title 'T->A' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:12 title 'T->C' with lines lw 1 lt 1 lc rgb 'grey', \\
'$outname.${end}p_substitutions.txt' using 1:13 title 'T->G' with lines lw 1 lt 1 lc rgb 'grey'";
close GNUTEMP;
system "gnuplot gnutemp.delme 2>/dev/null";# or die "could not generate plots\n";
unlink ("gnutemp.delme");
open GNUTEMP2, ">gnutemp2.delme" or die "could not write temp file for gnuplot";
if ($yrange) {
print GNUTEMP2 "set yrange [0:$yrange]\n";
}
print GNUTEMP2
"set terminal pdf
set output '$outname.${end}p_refbase_composition.pdf'
set title (\"${end}p reference base composition plot\\n$outname\")
```

```
set xlabel 'Distance from ${end}` end'
set ylabel 'Reference base frequencies'
set xtic 5
set grid
set xrange [-$xrange:$xrange]
set arrow from graph 0.5,0 to graph 0.5,1 nohead lw 1
set style fill transparent solid 0.1 noborder
ShadecolorA = '#27ad81'
ShadecolorC = '#e56b5d'
ShadecolorG = '#000004'
ShadecolorT = '#1c1044'
plot '$outname.${end}p_refbase_composition.txt' using 1:2 title 'A' with lines lt 1 lw 1 lc rgb 'green', \\
" using 1:6:7 with filledcurve fc rgb ShadecolorA notitle, \\
'$outname.${end}p_refbase_composition.txt' using 1:3 title 'C' with lines lt 1 lw 1 lc rgb 'red', \\
" using 1:8:9 with filledcurve fc rgb ShadecolorC notitle, \\
'$outname.${end}p_refbase_composition.txt' using 1:4 title 'G' with lines lt 1 lw 1 lc rgb 'black', \\
" using 1:10:11 with filledcurve fc rgb ShadecolorG notitle, \\
'$outname.${end}p_refbase_composition.txt' using 1:5 title 'T' with lines lt 1 lw 1 lc rgb 'blue', \\
" using 1:12:13 with filledcurve fc rgb ShadecolorT notitle";
close GNUTEMP2;
system "gnuplot gnutemp2.delme 2>/dev/null";
unlink ("gnutemp2.delme");
open GNUTEMP, ">gnutemp.delme10" or die "could not write temp file for gnuplot";
print GNUTEMP
"set terminal pdf
set output '$outname.${end}p_dinucleotide_substitutions_left.pdf'
set title (\"${end}p dinucleotide substitution plot (1)\\n$outname\")
set xlabel 'Distance from ${end}` end'
set ylabel 'Substitution frequencies'
set xtic 5
```

```
set grid

set xrange $gnu_xrange

plot '$outname.${end}p_dinucleotide_substitutions.txt' using 1:2 title 'aA->aC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:3 title 'aA->aG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:4 title 'aA->aT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:5 title 'aC->aA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:6 title 'aC->aG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:7 title 'aC->aT' with lines lw 1 lt 1 lc rgb 'blue', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:8 title 'aG->aA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:9 title 'aG->aC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:10 title 'aG->aT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:11 title 'aT->aA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:12 title 'aT->aC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:13 title 'aT->aG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:14 title 'cA->cC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:15 title 'cA->cG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:16 title 'cA->cT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:17 title 'cC->cA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:18 title 'cC->cG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:19 title 'cC->cT' with lines lw 1 lt 1 lc rgb 'red', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:20 title 'cG->cA' with lines lw 1 lt 1 lc rgb 'grey', \\
```

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:21 title 'cG->cC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:22 title 'cG->cT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:23 title 'cT->cA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:24 title 'cT->cC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:25 title 'cT->cG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:26 title 'gA->gC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:27 title 'gA->gG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:28 title 'gA->gT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:29 title 'gC->gA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:30 title 'gC->gG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:31 title 'gC->gT' with lines lw 1 lt 1 lc rgb 'green', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:32 title 'gG->gA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:33 title 'gG->gC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:34 title 'gG->gT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:35 title 'gT->gA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:36 title 'gT->gC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:37 title 'gT->gG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:38 title 'tA->tC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:39 title 'tA->tG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:40 title 'tA->tT' with lines lw 1 lt 1 lc rgb 'grey', \\

```
'$outname.${end}p_dinucleotide_substitutions.txt' using 1:41 title 'tC->tA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:42 title 'tC->tG' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:43 title 'tC->tT' with lines lw 1 lt 1 lc rgb 'black', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:44 title 'tG->tA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:45 title 'tG->tC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:46 title 'tG->tT' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:47 title 'tT->tA' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:48 title 'tT->tC' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:49 title 'tT->tG' with lines lw 1 lt 1 lc rgb 'grey'";

close GNUTEMP;

system "gnuplot gnutemp.delme10 2>/dev/null";# or die "could not generate plots\n";

unlink ("gnutemp.delme10");

open GNUTEMP, ">gnutemp.delme11" or die "could not write temp file for gnuplot";

print GNUTEMP

"set terminal pdf

set output '$outname.${end}p_dinucleotide_substitutions_right.pdf'

set title (\"${end}p dinucleotide substitution plot (2)\\n$outname\")

set xlabel 'Distance from ${end}` end'

set ylabel 'Substitution frequencies'

set xtic 5

set grid

set xrange $gnu_xrange

plot '$outname.${end}p_dinucleotide_substitutions.txt' using 1:50 title 'Aa->Ca' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:51 title 'Aa->Ga' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:52 title 'Aa->Ta' with lines lw 1 lt 1 lc rgb 'grey', \\
```

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:53 title 'Ac->Cc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:54 title 'Ac->Gc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:55 title 'Ac->Tc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:56 title 'Ag->Cg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:57 title 'Ag->Gg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:58 title 'Ag->Tg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:59 title 'At->Ct' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:60 title 'At->Gt' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:61 title 'At->Tt' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:62 title 'Ca->Aa' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:63 title 'Ca->Ga' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:64 title 'Ca->Ta' with lines lw 1 lt 1 lc rgb 'blue', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:65 title 'Cc->Ac' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:66 title 'Cc->Gc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:67 title 'Cc->Tc' with lines lw 1 lt 1 lc rgb 'red', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:68 title 'Cg->Ag' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:69 title 'Cg->Gg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:70 title 'Cg->Tg' with lines lw 1 lt 1 lc rgb 'green', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:71 title 'Ct->At' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:72 title 'Ct->Gt' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:73 title 'Ct->Tt' with lines lw 1 lt 1 lc rgb 'black', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:74 title 'Ga->Aa' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:75 title 'Ga->Ca' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:76 title 'Ga->Ta' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:77 title 'Gc->Ac' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:78 title 'Gc->Cc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:79 title 'Gc->Tc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:80 title 'Gg->Ag' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:81 title 'Gg->Cg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:82 title 'Gg->Tg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:83 title 'Gt->At' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:84 title 'Gt->Ct' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:85 title 'Gt->Tt' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:86 title 'Ta->Aa' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:87 title 'Ta->Ca' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:88 title 'Ta->Ga' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:89 title 'Tc->Ac' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:90 title 'Tc->Cc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:91 title 'Tc->Gc' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:92 title 'Tg->Ag' with lines lw 1 lt 1 lc rgb 'grey', \\

```
'$outname.${end}p_dinucleotide_substitutions.txt' using 1:93 title 'Tg->Cg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:94 title 'Tg->Gg' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:95 title 'Tt->At' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:96 title 'Tt->Ct' with lines lw 1 lt 1 lc rgb 'grey', \\

'$outname.${end}p_dinucleotide_substitutions.txt' using 1:97 title 'Tt->Gt' with lines lw 1 lt 1 lc rgb 'grey'";

close GNUTEMP;

system "gnuplot gnutemp.delme11 2>/dev/null";# or die "could not generate plots\n";

unlink ("gnutemp.delme11");

}

print STDERR "plots created, generating PDF\n";

system "gs -dNOPAUSE -sDEVICE=pdfwrite -sOUTPUTFILE=$outname.plots.pdf -dBATCH $outname.5p_substitutions.pdf $outname.3p_substitutions.pdf $outname.5p_refbase_composition.pdf $outname.3p_refbase_composition.pdf $outname.5p_dinucleotide_substitutions_left.pdf $outname.5p_dinucleotide_substitutions_right.pdf $outname.3p_dinucleotide_substitutions_left.pdf $outname.3p_dinucleotide_substitutions_right.pdf 1>/dev/null";

system "rm $outname.[53]p_*.pdf";

}

sub decode_flag {

my $hex = shift(@_);

my $bin = sprintf "%b", $hex;

my $literal = "qdfs21RrUuPp";

my $string;

foreach my $pos (-length($bin) .. -1) {

$string .= substr $literal, $pos, 1 if (substr $bin, $pos, 1);

}

return $string || "_";

}

sub help {

print "
```

This script reads a bam file and computes the reference base composition and substitution frequencies around 5p and 3p ends, including for di-nucleotides. Substitutions frequencies include all possible types; in addition CG->TG substitutions are considered separately. Reference base composition around alignment starts and ends can only be computed if the reference genome is provided. Input are sorted alignment files in BAM format.

[usage]

./substitution_patterns [-options] in.bam in2.bam ...

[options]

-reference provide reference genome in order to be able to compute reference base composition

outside of sequence alignments [optional; choose whole genome fasta file]

-minread lower sequence length cutoff [default 30]

-maxread upper sequence length cutoff [default 1000]

-quality map quality cutoff [default 0]

-unpaired do not exclude unpaired reads [off]

-graphical produce graphical output

-xrange define x-axis range for plotting [default 50]

-help generates this help message

-report define interval at which files are written [default 100_000]

[output]

in.5p_refbase_composition.txt reference base composition around 5p end (position 0 = terminal 5p base)

in.3p_refbase_composition.txt reference base composition around 3p end (position 0 = terminal 3p base)

in.5p_substitutions.txt substitution rates at 5p end

in.3p_substitutions.txt substitution rates at 3p end

in.Xp_dinucleotide_substitutions.txt di-nucleotide substitution rates

in.Xp_dinucleotide_refbase_compositions di-nucleotide reference base composition

STDOUT number of sequences going into analysis, number of sequences filtered out

DEPENDENCIES

This perl script has been successfully used with perl 5 (version 22). It requires 2 perl modules to be installed (File::Basename, Getopt::Long) as well as samtools (successfully used with version 1.3.1). If graphical output is turned on, gnuplot (version 5.0) and GPL Ghostscript (version 9.26) have to be installed.

";

```
exit;
}
```