#!/usr/bin/perl -w

use strict;

use File::Basename qw(basename dirname);

my $help=<<USAGE;

Note: Remove reads with adapter and low-quality reads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author : zhangqing

Modifier : meijunpu, meijunpu\@genomics.org.cn

Modified date: 2011.7.28,2011.8.5,2011.8.18

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

USAGE: PE: perl $0 <adapter\_file1> <adapter\_file2> <sample\_file1> <sample\_file2> <out\_file1> <out\_file2> <out\_dir> [<LowQual> <Q\_rate>]

SE: perl $0 <adapter\_file1> <sample\_file1> <out\_file1> <out\_dir> [<LowQual> <Q\_rate>]

adapter\_file1: must be \_1 adapter file

sample\_file1: must be \_1 fq file

LowQual/Q\_rate: Remove low-quality reads in which bases with Q <=5 account for more than 50% of the total[5,0.5]

USAGE

die "$help" unless(@ARGV==7 or @ARGV==9 or @ARGV==4 or @ARGV==6); # PE or SE

print "Remove adapter.. START TIME: ",`date`;

my ($adapter\_file1, $adapter\_file2, $sample\_file1, $sample\_file2, $out\_file1, $out\_file2, $out\_dir, @aQual);

if(@ARGV == 7 or @ARGV == 9){# PE

($adapter\_file1,$adapter\_file2,$sample\_file1,$sample\_file2,$out\_file1,$out\_file2,$out\_dir,@aQual) = @ARGV;

$aQual[0]=$ARGV[7]||5; $aQual[1]=$ARGV[8]||0.5;

}elsif(@ARGV == 4 or @ARGV == 6){ # SE

($adapter\_file1,$sample\_file1,$out\_file1,$out\_dir,@aQual) = @ARGV;

$aQual[0]=$ARGV[4]||5; $aQual[1]=$ARGV[5]||0.5;

}

my @baseQual = qw/@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ \_ ` a b c d e f g h i/; # ASCLL - 64 = 0-40 # B-f/2-38

my $STR;

my @quality;

for(my $i=2;$i<=$aQual[0];$i++){

if($baseQual[$i] =~ /\w/){

$STR .= "$baseQual[$i]";

}else{

$STR .= "\\$baseQual[$i]";

}

}

# -----------------------------------------------------------------

my %hReadRm=(); # all reads in the hash will be removeda

my ($readLen1,$readLen2) = (0,0); # read length

my($AdaReNum1, $AdaReNum2) = (0,0); # adapter reads number

my($lowQReNum1, $lowQReNum2) = (0,0); # low-quality reads number

my($OReadNum1, $OReadNum2, $MReadNum1, $MReadNum2) = (0,0,0,0); # Original/Modified reads number

my ($Oq20Base,$Mq20Base) = (0,0);

my ($or\_GC,$cl\_GC) = (0,0);

# read adapter files and fq files

ReadFiles($adapter\_file1, \$AdaReNum1, $sample\_file1, \$lowQReNum1, \$readLen1, \%hReadRm);

ReadFiles($adapter\_file2, \$AdaReNum2, $sample\_file2, \$lowQReNum2, \$readLen2, \%hReadRm) if($sample\_file2); # PE

# remove reads

rmReads($sample\_file1, $out\_file1, \$OReadNum1, \$MReadNum1, \%hReadRm);

rmReads($sample\_file2, $out\_file2, \$OReadNum2, \$MReadNum2, \%hReadRm) if($sample\_file2); # PE

my $OReadNum = $OReadNum1+$OReadNum2;

my $stat\_file="$out\_dir/rmAdapter.stat";

open STAT,">$stat\_file" or die $!;

print STAT "Original reads number:\t$OReadNum\n";

print STAT "Original bases number:\t",$OReadNum1\*$readLen1 + $OReadNum2\*$readLen2,"\n";

print STAT "Modified reads number:\t",$MReadNum1+$MReadNum2,"\n";

print STAT "Modified bases number:\t",$MReadNum1\*$readLen1 + $MReadNum2\*$readLen2,"\n";

print STAT "Adapter reads number:\t",$AdaReNum1+$AdaReNum2,"\n";

print STAT "Low-quality reads number:\t",$lowQReNum1+$lowQReNum2,"\n";

print STAT "Low-quality reads rate(\%):\t",100\*($lowQReNum1+$lowQReNum2)/$OReadNum,"\n";

print STAT "Adapter reads rate(\%):\t",100\*($AdaReNum1+$AdaReNum2)/$OReadNum,"\n";

print STAT "Modified reads rate(\%):\t",100\*($MReadNum1+$MReadNum2)/$OReadNum,"\n";

print STAT "Original Q20 bases rate(\%):\t",sprintf ("%.2f",100\*$Oq20Base/($OReadNum1\*$readLen1 + $OReadNum2\*$readLen2)),"\n";

print STAT "Modified Q20 bases rate(\%):\t",sprintf ("%.2f",100\*$Mq20Base/($MReadNum1\*$readLen1 + $MReadNum2\*$readLen2)),"\n";

print STAT "Original GC rate(\%):\t",sprintf("%.2f",100\*$or\_GC/($OReadNum1\*$readLen1 + $OReadNum2\*$readLen2)),"\n";

print STAT "Modified GC rate(\%):\t",sprintf("%.2f",100\*$cl\_GC/($MReadNum1\*$readLen1 + $MReadNum2\*$readLen2)),"\n";

close STAT;

print "Remove adapter.. END TIME: ",`date`;

my $length = @quality;

open QA,">$out\_dir/quality.list" or die $!;

for my $l(0..$length-1){

my $equa = sprintf("%.2f",$quality[$l]/($MReadNum1+$MReadNum2));

my $num = $l +1;

print QA "$num\t$equa\n";

}

close QA;

my $max = sprintf("%.f",($readLen1+$readLen2)/2);

if($max % 10 != 0){

$max = 10\*(int($max/10)+1);

}

my $Rline =<<Rline;

pdf(file="$out\_dir/quality.pdf",w=8,h=6)

rt <- read.table("$out\_dir/quality.list")

opar <- par()

x <- rt\$V1[1:$max]

y <- rt\$V2[1:$max]

par(mar=c(4.5, 4.5, 2.5, 2.5))

plot(x,y,col="red",type='l', lwd=2, bty="l",xaxt="n",yaxt="n", xlab="", ylab="", ylim=c(0,40),xlim=c(0,$max))

xpos <- seq(0,$max,by=10)

ypos <- seq(0,40,by=10)

axis(side=1, xpos, tcl=0.2, labels=FALSE)

axis(side=2, ypos, tcl=0.2, labels=FALSE)

mtext("Distribution along reads",side=1, line=2, at=median(xpos), cex=1.5 )

mtext("Quality",side=2, line=3, at=median(ypos), cex=1.5)

mtext(xpos, side=1, las=1, at=xpos, line=0.3, cex=1.4)

mtext(ypos, side=2, las=1, at=ypos, line=0.3, cex=1.4)

par(opar)

dev.off()

Rline

open Rout,">$out\_dir/quality.R" or die $!;

print Rout $Rline;

close Rout;

my $R;

if($out\_dir =~ /ifshk/){

$R = "/opt/blc/genome/biosoft/R-212/bin/R CMD BATCH";

}else{

$R = "/opt/blc/genome/biosoft/R/bin/R CMD BATCH"

}

chdir $out\_dir;

system("$R $out\_dir/quality.R");

system("convert $out\_dir/quality.pdf $out\_dir/quality.png");

system("convert $out\_dir/quality.pdf $out\_dir/quality.jpg");

system("rm $out\_dir/quality.Rout $out\_dir/quality.R $out\_dir/.RData");

#---------------------------------------------------------------------------------------------------

sub ReadFiles{

# read adapter file

my($adaFile,$readNum1,$fqFile,$readNum2,$readL,$hash)=@\_;

my %hTemp;

if($adaFile =~ /\.gz$/){

open ADP,"gzip -dc $adaFile|" or die $!;

}else{

open(ADP,$adaFile) || die "$!";

}

# #reads\_id reads\_len reads\_start reads\_end adapter\_id adapter\_len adapter\_start adapter\_end align\_le n mismatch gap

# A20BD5ABXX:5:1:1719:1978#ATCACGAT/1 90 44 76 iPE-3+ 33 0 32 33 0 0

<ADP>;

while(<ADP>){

chomp;

my $ID= (split /\t/)[0];

$$readL = (split /\t/)[1];

$ID =~ s/\/[12]$//;

#$$hash{$ID}++;

#$$readNum1++;

$hTemp{$ID}++;

}

close ADP;

# read fq file

if($fqFile =~ /\.gz$/){

open FQ,"gzip -dc $fqFile|" or die $!;

}else{

open(FQ,$fqFile) || die "$!";

}

while(my $ID=<FQ>){

my($seq,$t,$qual);

my ($seqL,$lowQualNum);

$lowQualNum=0;

chomp $ID; $ID =~ s/^@//; $ID =~ s/\/[12]$//;

if(exists $hTemp{$ID}){

$$hash{$ID}++;

$$readNum1++;

}

$seq=<FQ>; chomp $seq;

$t=<FQ>;

$qual=<FQ>; chomp $qual;

$seqL=length $qual;

$lowQualNum = $qual =~ s/[$STR]//g;

my @q\_line = split //,$qual;

for my $q\_line(@q\_line){

if(((ord $q\_line)-64) >= 20){

$Oq20Base ++;

}

}

if($lowQualNum/$seqL>$aQual[1]){ # >50%

$$hash{$ID}++;

$$readNum2++;

}

my @seq = split //,$seq;

for my $sq(@seq){

if($sq =~ /[GC]/){

$or\_GC ++;

}

}

}

close FQ;

%hTemp=();

}

#--------------------------------------------------------------------------------------------

sub rmReads{

my($fqFile, $outFile, $readNum1, $readNum2, $hash) = @\_;

if($fqFile =~ /\.gz$/){

open FQ,"gzip -dc $fqFile|" or die $!;

}else{

open FQ,$fqFile || die $!;

}

if($outFile =~ /\.gz$/){

open OUT,"|gzip > $outFile" or die $!;

}else{

open OUT,"> $outFile" or die $!;;

}

while(my $ID=<FQ>){

$$readNum1++;

my($ID\_t,$seq,$t,$qual);

$ID\_t = $ID;

chomp $ID\_t; $ID\_t =~ s/^@//; $ID\_t =~ s/\/[12]$//;

$seq=<FQ>; #chomp $seq;

$t=<FQ>;

$qual=<FQ>;

next if(exists $$hash{$ID\_t});

$ID =~ s/\#(\S+)\//\#\//;

print OUT "$ID$seq$t$qual";

$$readNum2++;

chomp $qual;

my @Qua = split //,$qual;

for my $i(0..@Qua-1){

$quality[$i] += (ord $Qua[$i]) - 64;

if(((ord $Qua[$i]) - 64) >= 20){

$Mq20Base ++;

}

}

my @seq = split //,$seq;

for my $sq(@seq){

if($sq =~ /[GC]/){

$cl\_GC ++;

}

}

}

close OUT;

close FQ;

}