Custom scripts:

I: Script to determine methylation calls using Bismark (custom script for protocol step 11.6):

#!/usr/bin/perl

# a script that calls base methylation from bismark output

# default at least 10 read coverage

# default at least 20 quality score per C

# examples:

# methylationCall\_fromBismark.pl --sample sample\_id --output /path/to/output sortedInput.txt >> log.txt

# methylationCall\_fromBismark.pl --minqual 20 --mincov 0 --sample samplename sortedBismarkAlignments.txt

use warnings;

use strict;

use Getopt::Long;

# variables

my $minqual = 20; # the minimum quality score to in the output

my $mincov = 0; # the minimum coverage to appear in the output

my $offset = 33; # this is no longer necessary, bismark outputs phred33 scores regardless of input scores

my $sample = 'sample';

my $output = './';

my $outFile = '';

my $outchh = '';

my $outchg = '';

my $lnctr = '';

GetOptions(

'sample=s' => \$sample,

'minqual=i' => \$minqual,

'mincov=i' => \$mincov,

'offset=i' => \$offset,

'output=s' => \$output

);

my $input\_file = $ARGV[0];

if( @ARGV < 1 ){

&usage();

}

# example line from bismark

# D7ZQJ5M1\_0444:1:1:1318:1996#0/1 + chr14 49602985 49603034 CNGAGAGAGAGTTCGTTTTATGTTTAGTGAGAAATGGTGTTAGGGTTTTT CGGAGAGAGAGCCCGTCCTATGCTTAGTGAGAAATGGTGTCAGGGTCTCCAA Z..........hxZ..hh....h.................x.....h.hh CT CT ]BZPPYZZUY`]^b\_deeceVZcacb`^S\_BBBBBBBBBBBBBBBBBBBB

# Single-end output format (tab-separated):

#

# (1) <seq-ID>

# (2) <read alignment strand>

# (3) <chromosome>

# (4) <start position>

# (5) <end position>

# (6) <observed bisulfite sequence>

# (7) <equivalent genomic sequence>

# (8) <methylation call>

# (9) <read conversion

# (10) <genome conversion>

# (11) <read quality score (Phred33)>

# bismark output codes are:

# z unmeth CpG

# Z meth CpG

# x unmeth CHG

# X meth CHG

# h unmeth CHH

# H meth CHH

print "methyl call params: minqual:$minqual mincov:$mincov offset:$offset sample:$sample output:$output\n\n";

#open CpG output file and print the header

$outFile = $output . "/" . "methylcall.CpG." . $sample . ".mincov" . $mincov . ".txt";

open( my $out, ">$outFile" );

print $out join("\t",qw(chrBase chr base strand coverage freqC freqT)),"\n";

#open CHH output file and print the header

$outchh = $output . "/" . "methylcall.CHH." . $sample . ".mincov" . $mincov . ".txt";

open( my $CHHout, ">$outchh" );

print $CHHout join("\t",qw(chrBase chr base strand coverage freqC freqT nextBase)),"\n";

#open CHG output file and print the header

$outchg = $output . "/" . "methylcall.CHG." . $sample . ".mincov" . $mincov . ".txt";

open( my $CHGout, ">$outchg" );

print $CHGout join("\t",qw(chrBase chr base strand coverage freqC freqT nextBase)),"\n";

print "outfile : $outFile\n";

print "outchh : $outchh\n";

print "outchg : $outchg\n\n";

#read-in file to count C bases

my %CGmethHash = ();

my %CHHmethHash = ();

my %CHGmethHash = ();

my $lastPos = -1;

my $lastChrom = "null";

$lnctr = 0;

open ( my $fh, $input\_file );

while( <$fh> )

{

$lnctr++;

if($\_=~/Bismark/){ next; } # step over the header line

chomp;

my @cols = split(/\t/,$\_);

my $strand = $cols[1];

my $chr = $cols[2];

my $start = $cols[3];

my $end = $cols[4];

my @gbases = split("",$cols[6]); # get the genomic bases

my @mcalls = split("",$cols[7]); # get the bismark methylation calls

my @quals = split("",$cols[10]); # get the quality scores

#processes hashes if start-LastPos>100

if( ( $start-$lastPos > 100 && $lastPos != -1 ) || ($chr ne $lastChrom && $lastChrom ne "null" ))

{

processCGmethHash (\%CGmethHash, $out, $mincov);

processCHmethHash (\%CHHmethHash, $CHHout, $mincov);

processCHmethHash (\%CHGmethHash, $CHGout, $mincov);

%CGmethHash = ();

%CHHmethHash = ();

%CHGmethHash = ();

}

# iterate over the mapped sequence

for( my $i=0;$i< @quals; $i++)

{

# if ( (ord($quals[$i])-64) < $minqual || ( $gbases[$i] ne "C") ) { next; }

if ( (ord($quals[$i])-$offset) < $minqual || ( $gbases[$i] ne "C") ) {

next;

}

if( ( $gbases[$i] eq "C" && $i==(scalar @quals)) && $gbases[$i-1].$gbases[$i].$gbases[$i+1].$gbases[$i+2] eq "CCGG" ) {

next;

} #if last base is a C and it is a part of CCGG motif, don't call for meth

my $key; # initilaize the hash key

if($strand eq "+") {

$key = join("|",("F",$chr,$start+$i,$gbases[$i+1]));

} else {

$key = join("|",("R",$chr,$end-$i,$gbases[$i+1]));

}

if($gbases[$i+1] eq "G") # if is CpG

{

unless(exists $CGmethHash{$key}) {

$CGmethHash{$key}=[0,0,0];

}

if( $mcalls[$i] eq "Z" ) {

$CGmethHash{$key}->[0]++; # update Cs

}

elsif( $mcalls[$i] eq "z") {

$CGmethHash{$key}->[1]++; # update Ts

}

else {

$CGmethHash{$key}->[2]++; # update other bases

}

}

else # if genomic base is non-CpG

{

if ($gbases[$i+2] eq "G") { # CHG

# process X|x

# x unmeth CHG

# X meth CHG

unless(exists $CHGmethHash{$key}) {

$CHGmethHash{$key}=[0,0,0];

}

if( $mcalls[$i] eq "X" ) {

$CHGmethHash{$key}->[0]++;

}

elsif( $mcalls[$i] eq "x" ) {

$CHGmethHash{$key}->[1]++;

}

else {

$CHGmethHash{$key}->[2]++;

}

}

if ($gbases[$i+2] =~ /[ACT]/) { # CHH

# process H|h

# h unmeth CHH

# H meth CHH

unless(exists $CHHmethHash{$key}) {

$CHHmethHash{$key}=[0,0,0];

}

if( $mcalls[$i] eq "H" ) {

$CHHmethHash{$key}->[0]++;

}

elsif( $mcalls[$i] eq "h" ) {

$CHHmethHash{$key}->[1]++;

}

else {

$CHHmethHash{$key}->[2]++;

}

}

}

}

$lastPos = $end;

$lastChrom = $chr;

if( $lnctr % 100000 == 0 ) {

warn "methylation calling processed $lnctr sequences so far\n";

}

}

# capture anything left in hashes

processCGmethHash (\%CGmethHash, $out, $mincov);

processCHmethHash (\%CHHmethHash, $CHHout, $mincov);

processCHmethHash (\%CHGmethHash, $CHGout, $mincov);

close $fh;

close $out;

close $CHGout;

close $CHHout;

exit 0;

### SUBROUTINES ###################

sub usage

{

print STDERR <<EndOfUsage;

Usage:

perl $0 [options] input\_file

Examples:

methylationCall\_fromBismark.pl --sample sample\_id --output /path/to/output sortedInput.txt >> log.txt

methylationCall\_fromBismark.pl --minqual 20 --mincov 0 --sample samplename sortedBismarkAlignments.txt

Options:

--minqual : minimum quality score cutoff

default 20

--mincov : minimum read coverage

default 0

--sample : sample id

default 'sample'

this is used to identify the output file

--offset : FASTQ qualities are ASCII chars equal to the Phred quality plus 33 or 64

default '33'

--output : path\_to\_output

default current working directory

EndOfUsage

exit;

}

# process a given CG methlation hash

# writes the filter passing CGs to output file

sub processCGmethHash

{

my ($CGmethHash,$out, $mincov) = (@\_);

foreach my $key (keys %{$CGmethHash})

{

my($strand, $chr, $loc) = split(/\|/, $key);

my $noCs = $CGmethHash->{$key}->[0];

my $noTs = $CGmethHash->{$key}->[1];

my $noOs = $CGmethHash->{$key}->[2];

my $Cperc = sprintf("%.3f", 100\*$noCs/($noTs+$noCs+$noOs) );

my $Tperc = sprintf("%.3f", 100\*$noTs/($noTs+$noCs+$noOs) );

if(($noTs+$noCs)/($noTs+$noCs+$noOs) > 0.9 && ($noTs+$noCs+$noOs) >= $mincov )

{

print $out join("\t",($chr.".".$loc,$chr,$loc,$strand,$noCs+$noTs+$noOs,$Cperc,$Tperc) ),"\n";

}

}

return 1;

}

# process a given non CG methlation hash

# writes the filter passing Cs to a hash, that hash will be used to calculate conversion rate later on

sub processCHmethHash

{

my ($CHmethHash, $out, $mincov) = (@\_);

foreach my $key (keys %{$CHmethHash})

{

my ($strand, $chr, $loc, $nextBase) = split(/\|/, $key);

my $noCs = $CHmethHash->{$key}->[0];

my $noTs = $CHmethHash->{$key}->[1];

my $noOs = $CHmethHash->{$key}->[2];

my $Cperc = sprintf("%.3f", 100\*$noCs/($noTs+$noCs+$noOs) );

my $Tperc = sprintf("%.3f", 100\*$noTs/($noTs+$noCs+$noOs) );

if (($noTs+$noCs)/($noTs+$noCs+$noOs) > 0.95 && ($noTs+$noCs+$noOs) >= $mincov ){

print $out join("\t",($chr.".".$loc,$chr,$loc,$strand,$noCs+$noTs+$noOs,$Cperc,$Tperc,$nextBase) ),"\n";

}

}

return 1;

}

# get the median value of a given array

# array of numbers

sub median

{

my $array\_ref = shift;

my @array = ();

my $median = "";

@array = sort { $a <=> $b } @$array\_ref;

if( (@array % 2) == 1 ) {

$median = $array[ @array/2 ];

} else {

$median = ( $array[ @array/2 - 1 ] + $array[ @array/2 ] ) / 2;

}

return $median;

}

II: Script to compute bisulfite conversion rate (custom script for protocol step 11.7):

#!/usr/bin/perl

# filter for a given coverage and compute conversion rates

# example:

# conversionRate.pl -mincov <int> -cpg <cpgfile> -chg <chgfile> -chh <chhfile>

# conversionRate.pl -mincov <int> -cpg methylcall.CpG.txt -chg methylcall.CHG.txt -chh methylcall.CHH.txt

use warnings;

use strict;

use Getopt::Long;

use File::Basename;

# variables

my $verbose = '';

my $sample = '';

my $outdir = '';

my $mincov = 0;

my $cpgfile = '';

my $chgfile = '';

my $chhfile = '';

my $outfile = '';

my $header = '';

my $line = '';

my $numcpg = 0;

my $meancpg = 0;

my $numchg = 0;

my $numchh = 0;

my $tmp = '';

my $results = '';

my $totalC = 0;

my $avgConvRate = 0;

my $medianConvRate = 0;

my $totalCF = 0;

my $avgConvRateF = 0;

my $medianConvRateF = 0;

my $totalCR = 0;

my $avgConvRateR = 0;

my $medianConvRateR = 0;

my @cpg = ();

my %chg = ();

my %chh = ();

my @tmp = ();

GetOptions(

'h|help' => sub { usage();

exit 0;

},

'mincov=i' => \$mincov,

'sample=s' => \$sample,

'cpgfile=s' => \$cpgfile,

'chgfile=s' => \$chgfile,

'chhfile=s' => \$chhfile,

'v|verbose' => \$verbose

);

unless( $chgfile && $chhfile ) {

usage();

unless( $chgfile ) {

print "no chg file given\n";

}

unless( $chhfile ) {

print "no chh file given\n";

}

print "both chg and chh files are necesary for computing conversion rates\n\n";

exit 1;

}

# set mincov

unless( $mincov ) {

$mincov = 0;

print "minimum coverage not given, setting to 0 - no filtering will be done\n" if $verbose ;

}

# parse outdir from input files

if( $cpgfile ) {

($tmp, $outdir) = fileparse($cpgfile);

} elsif ( $chgfile ) {

($tmp, $outdir) = fileparse($chgfile);

} else {

$outdir = "./";

}

# get sample name from input file

# works for typical methylcall output names

# like methylcall.CHG.Sample\_abc123.mincov0.txt

unless( $sample ) {

@tmp = split("\\.", $tmp);

$sample = $tmp[-3];

}

if( $verbose ) {

print "$0\n";

print "mincov : $mincov\n";

print "sample : $sample\n";

print "outdir : $outdir\n";

print "cpg file : $cpgfile\n";

print "chg file : $chgfile\n";

print "chh file : $chhfile\n\n";

}

if( ! $cpgfile ) {

print "no cpg file given\n" if $verbose;

} else {

# get cpg data

open( F, "<$cpgfile") or die "cannot find cpg file [$cpgfile]\n\n";

$header = <F>;

if( $mincov > 0 ) {

$outfile = $outdir."cpg.".$sample.".mincov".$mincov.".txt";

open( G, ">$outfile") or die "cannot open outfile [$outfile]\n\n";

print G $header;

}

while( $line = <F> ) {

@tmp = split("\t", $line);

if( $tmp[4] >= $mincov ) {

print G $line if $mincov > 0;

push( @cpg, $tmp[4]);

}

}

if( $mincov > 0 ) {

close(G);

print "filtered output in file: $outfile\n" if $verbose;

}

close(F);

$numcpg = @cpg;

if( $numcpg > 0 ) {

$meancpg = sprintf("%.2f", mean(@cpg));

}

$results .= "minimum coverage : ". $mincov ."X\n";

$results .= "number of CpG's covered : $numcpg\n";

$results .= "mean CpG coverage : $meancpg \n\n";

}

if( $chgfile && $chhfile ) {

# get chg data

print "getting chg data\n" if $verbose;

open( F, "<$chgfile") or die "cannot find chg file [$chgfile]\n$!\n";

$header = <F>;

if( $mincov > 0 ) {

$outfile = $outdir."chg.".$sample.".mincov".$mincov.".txt";

open( G, ">$outfile") or die "cannot open outfile [$outfile]\n\n";

print G $header;

}

@{$chg{"F"}} = ();

@{$chg{"R"}} = ();

while( $line = <F> ) {

@tmp = split("\t", $line);

if( $tmp[4] >= $mincov ) {

print G $line if $mincov > 0;

if( $tmp[3] eq "F" ) {

push( @{$chg{"F"}}, $tmp[6] );

}

if( $tmp[3] eq "R" ) {

push( @{$chg{"R"}}, $tmp[6] );

}

}

}

if( $mincov > 0 ) {

close(G);

print "filtered output in file: $outfile\n" if $verbose;

}

close(F);

# get chh data

print "getting chh data\n" if $verbose;

open( F, "<$chhfile") or die "cannot find chh file [$chhfile]\n$!\n";

$header = <F>;

if( $mincov > 0 ) {

$outfile = $outdir."chh.".$sample.".mincov".$mincov.".txt";

open( G, ">$outfile") or die "cannot open outfile [$outfile]\n\n";

print G $header;

}

@{$chh{"F"}} = ();

@{$chh{"R"}} = ();

while( $line = <F> ) {

@tmp = split("\t", $line);

if( $tmp[4] >= $mincov ) {

print G $line if $mincov > 0;

if( $tmp[3] eq "F" ) {

push( @{$chh{"F"}}, $tmp[6] );

}

if( $tmp[3] eq "R" ) {

push( @{$chh{"R"}}, $tmp[6] );

}

}

}

if( $mincov > 0 ) {

close(G);

print "filtered output in file: $outfile\n" if $verbose;

}

close(F);

print "computing conversion rates\n" if $verbose;

$totalCF = @{$chg{"F"}} + @{$chh{"F"}};

$totalCR = @{$chg{"R"}} + @{$chh{"R"}};

$totalC = $totalCF + $totalCR;

$avgConvRateF = sprintf( "%.4f", mean( @{$chg{"F"}}, @{$chh{"F"}} ));

if( ! $avgConvRateF ) { $avgConvRateF = 0; }

$medianConvRateF = sprintf( "%.4f", median( @{$chg{"F"}}, @{$chh{"F"}} ));

if( ! $medianConvRateF ) { $medianConvRateF = 0; }

$avgConvRateR = sprintf( "%.4f", mean( @{$chg{"R"}}, @{$chh{"R"}} ));

if( ! $avgConvRateR ) { $avgConvRateR = 0; }

$medianConvRateR = sprintf( "%.4f", median( @{$chg{"R"}}, @{$chh{"R"}} ));

if( ! $medianConvRateR ) { $medianConvRateR = 0; }

$avgConvRate = sprintf( "%.4f", mean( @{$chg{"F"}}, @{$chh{"F"}}, @{$chg{"R"}}, @{$chh{"R"}} ));

if( ! $avgConvRate ) { $avgConvRate = 0; }

$medianConvRate = sprintf( "%.4f", median( @{$chg{"F"}}, @{$chh{"F"}}, @{$chg{"R"}}, @{$chh{"R"}} ));

if( ! $medianConvRate ) { $medianConvRate = 0; }

$results .= "Conversion Rates for sample ($sample) ";

$results .= $mincov."X minimum coverage\n\n";

$results .= "total otherC considered (>95% C+T) : $totalC\n";

$results .= "average conversion rate : $avgConvRate\n";

$results .= "median conversion rate : $medianConvRate\n\n";

$results .= "total otherC considered (Forward) (>95% C+T) : $totalCF\n";

$results .= "average conversion rate (Forward) : $avgConvRateF\n";

$results .= "median conversion rate (Forward) : $medianConvRateF\n\n";

$results .= "total otherC considered (Reverse) (>95% C+T) : $totalCR\n";

$results .= "average conversion rate (Reverse) : $avgConvRateR\n";

$results .= "median conversion rate (Reverse) : $medianConvRateR\n\n";

}

$outfile = $outdir."conversionRates.".$sample.".mincov".$mincov.".txt";

if( $verbose ) {

# print results to screen

print $results;

print "conversion rates saved to file [$outfile]\n";

}

# print results to file

open( OUTPUT, ">$outfile") or die "cannot open outfile [$outfile]\n\n";

print OUTPUT $results;

close(OUTPUT);

print "done...\n" if $verbose;

exit 0;

### SUBROUTINES ###################

sub usage

{

print STDERR <<EndOfUsage;

Computes the conversion rates for ERRBS results.

If a minimum coverage is given, the files will be filtered for the given value.

Usage:

perl $0 -mincov [mincov] -cpg [cpg file] -chg [chg file] -chh [chh file]

Examples:

conversionRate.pl -mincov 10 -cpg methylcall.CpG.sample.mincov0.txt -chg methylcall.CHG.sample.mincov0.txt -chh methylcall.CHH.sample.mincov0.txt

- filters all the files for coverage >= 10x and compute the conversion rates

conversionRate.pl -cpg methylcall.CpG.sample.mincov0.txt

- compute the number of cpg's covered at 0x coverage and the mean cpg coverage

conversionRate.pl -mincov 10 -cpg methylcall.CpG.sample.mincov0.txt

- filters the cpg file for 10x coverage and computes the number of cpg's covered and the mean cpg coverage

conversionRate.pl -chg methylcall.CHG.sample.mincov0.txt -chh methylcall.CHH.sample.mincov0.txt

- computes the conversion rates for 0x coverage

Options:

-chg optional

mandatory with -chh. necessary to compute conversion rates.

-chh optional

mandatory with -chg necessary to compute conversion rates.

-cpg optional

neccessary to compute mean cpg coverage and number of cpg's covered.

-h, -help print this help text.

-mincov default: 0

the minimum coverage to consider.

-sample default: sample

a name to include in the result files. an attempt will be made to parse

the name from the input files.

-v, -verbose increase verbosity.

EndOfUsage

}

# from pm Statistics::Lite

sub sum

{

return unless @\_;

return $\_[0] unless @\_ > 1;

my $sum;

foreach(@\_) { $sum+= $\_; }

return $sum;

}

sub mean

{

return unless @\_;

return $\_[0] unless @\_ > 1;

return sum(@\_)/scalar(@\_);

}

sub median

{

return unless @\_;

return $\_[0] unless @\_ > 1;

@\_= sort{$a<=>$b}@\_;

return $\_[$#\_/2] if @\_&1;

my $mid= @\_/2;

return ($\_[$mid-1]+$\_[$mid])/2;

}