Contents

4 5 4

```
#########################################################
#  10/24/2008
# RW Citek, C. Tatham, R.K. Maloney, A. Garrido, and J.A. Jeddeloh
# Copyright (C) 2005 Orion Genomics
# All rights reserved.
# Redistribution and use in source and binary forms are freely
# permitted provided that the above copyright notice and
# attribution and date of work and this paragraph are duplicated
# in all such forms and that neither this software nor software
# based in whole or in part on this software is sold for profit.
# THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
# IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
# WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR
# PURPOSE.
#########################################################

The purpose of this code is to generate a data table containing the cytosine methylation
occupancy information from bisulfite treated patient samples.    The approach should
work for the analysis of pooled PCR products derrvied from ultra deep sequenncing or
from shot gun library approaches derrived from complexity reduced samples.

The idea was to generate a large table that would eassily support querries allowing robust
statistical analysis of the methylation information per molecule as well as per CpG position.
The approach was utilized successfully to analyze PCR products amplified using site specific
primers that contained not only the 454 A and B primer information, but also patient specific
 nucleotide tags.   A full decription of the work is in press in the peer reviewed journal
Genome Research.

'''Massively Parallel Bisulphite Pyrosequencing Reveals the Molecular Complexity of
Breast Cancer Associated Cytosine Methylation Patterns Obtained from Tissue and Serum DNA'''

Yulia Korshunova1, Rebecca K. Maloney1, Nathan Lakey1, Robert W. Citek1, Blaire Bacher1,
Arief Budiman1, Jared M. Ordway1, W. Richard McCombie2, Jorge Leon1,
 Jeffrey A. Jeddeloh1+* and John D. McPherson3

1 Orion Genomics, LLC, 4041 Forest Park Ave, St. Louis. MO 63108;
2   Cold Spring Harbor Laboratory, Cold Spring Harbor, New York 11723;
3 Department of Molecular and Human Genetics, Baylor College of Medicine, One Baylor Plaza, Houston, TX 77030
* Current Address:
Roche NimbleGen, 500 S. Rosa, Madison, WI 53719
jjeddeloh[AT]nimblegen.com
```

The general idea is to create a working enviornment, generate the BLAST libraries necessary to use MethylMapper (http://www.sourceforge.net) for the analysis of the amplicons. The source files necessary are kept in a directory called base.files and linked into where they are necessary for use. Since each sequencing round resulted in modifications to the code, we chose to put each analysis from a sequencing run into a direcotry called round.00X where X is the interation of the analysis.

```
export TMPDIR=/extra/tmp     # needed for extra temp space for sort
runnum= #put what you want to name your experiment here
```

The working environment initially should look like this:

```
${runnum}
|-- base.files
`-- round.001
```

Commands:

```
cd ${runnum}
mkdir base.files
mkdir round.001
tree ${runnum}
```

The base.files folder holds the amplicon information and tar file. The round.001 (or round.002, round.003, etc.) folder is the working folder. For the base.files, they are usually created by hand or downloaded, e.g. from Wash-U. They can also be copied from previous runs, e.g run.001:

```
cp -r ../run.001/base.files/* base.files
ls -la base.files
```

1. Move to the working directory:

```
cd round.001
```

1. And run this script to initialize the environment:

```
set -x
clear

rm -r *
mkdir blastdb
ln -s ../base.files/* .
ls -la --color
```

For previouslly designed Bis{TY} amplicons, copy from your favorite folder:

```
cp -r ../../run.002/amp ../
```

amplicon

```
set -x

# genomic feature loci and TSS loci
# at Orion we had these gennomic segments in a database of ~1kb elements
# we like to maipulate sequence information by converting between tab delimited sequence files (.seq) and
# files in the fasta format (.fasta)
# this section will grab the elements from a databse, quick check them and get them ready for digital
# bisulfite conversion

{ arraydb.sql.pl -f feature.loci "
     select FEATURENAME,SEQUENCE1060_CHUNK
     from tblfeature_map b
     where ASSEMBLY_VERSION=2 and b.FEATURENAME in "

  grep -i ghsr \
/nfs/orion2/projects/MethylScope/Bisulfite/Analysis/breast.c3.2006-05-05/transcription.start.sites/tss.seq
} > loci.seq
# verify
cut -c1-100 loci.seq
wc -l loci.seq

tab2fasta.pl loci.seq > loci.fasta
# verify
grep -c '^>' loci.fasta

#See MethylMapper code

# bisulfite T convert for analysis
bisulfite_convert.pl loci.fasta > loci.bisT.fasta
fasta2tab.pl loci.bisT.fasta > loci.bisT.seq
length loci.bisT.fasta > loci.bisT.length
# verify
wc -l loci.bisT.*

# bisulfite Y convert as a conversion control
bisulfite_convert.pl -y loci.fasta > loci.bisY.fasta
fasta2tab.pl loci.bisY.fasta > loci.bisY.seq
length loci.bisY.fasta > loci.bisY.length
# verify
wc -l loci.bisY.*

# create blastdb of loci.bisT
formatdb -i loci.bisT.fasta -o T -p F -n blastdb/loci.bisT
# verify
fastacmd -I -d blastdb/loci.bisT

# create blastdb of loci.bisY
formatdb -i loci.bisY.fasta -o T -p F -n blastdb/loci.bisY
# verify
fastacmd -I -d blastdb/loci.bisY

#We will want to find the amplicon primer sequence in each read so we need to
#prepare the converted primers.  We will also use the position of the primers
#to generate the amplicons using a script that grabs sequence betweenn two
#coordinates, we call this script nab. The amplicons are used as a BLAST assembly
#scaffold (i.e. template with no-mistach).

# bisT primer sequence
perl -F'\t' -lane 'print join("_",@F[0..2]) ."\t$F[4]"' loci+primer.raw |
 tail +2 > primers.bisT.seq
tab2fasta.pl primers.bisT.seq  > primers.bisT.fasta
length primers.bisT.fasta > primers.bisT.length
# verify
wc -l primers.bisT.length
head primers.bisT.length

# create blastdb of bisT primers
formatdb -i primers.bisT.fasta -o T -p F -n blastdb/primers.bisT
# verify
fastacmd -I -d blastdb/primers.bisT

# find location of bisT primers in bisT loci
#May have to adjust the E-value ... its the threshold filter

blastall -p blastn -a 4 -i primers.bisT.fasta -d blastdb/loci.bisT -F F -W 15 -m 8 -e 1e-4 \
 > primers.bisT.blast.m8
blast.w.length.pl primers.bisT.blast.m8 primers.bisT.length loci.bisT.length \
 > primers.bisT.blast.m8.w.length
sed -e 's/revcomp/rc/' primers.bisT.blast.m8.w.length

# verify only 2 primers per amplicon
cut -d"_" -f1,2 primers.bisT.blast.m8.w.length |
 sort |
 uniq -c

# create amplicons coordinates within each 1Kb segment
paste \
 <( cut -f 1,2,9,10 primers.bisT.blast.m8.w.length | grep F$'\t' )  \
 <( cut -f 1,2,9,10 primers.bisT.blast.m8.w.length | grep R$'\t' )  |
 cut -f 1,2,3,7 \
 > amplicon.bisT.coords
# verify
cat amplicon.bisT.coords
wc -l amplicon.bisT.coords

# create primer, amplicon relation
perl -F'\t' -lane 'print "$F[0]\t$F[1]_$F[2]..$F[3]"' amplicon.bisT.coords \
 > primer.amplicon
# verify
head primer.amplicon
wc -l primer.amplicon

# create genomic coordinates

< amplicon.bisT.coords sed -e 's/_2*F//' -e 's/_bs//' \
```

```
 > amplicon.coords
# verify
head amplicon.coords
wc -l amplicon.coords

# Bisulfite conversion generates two non-complimentary DNA strands so one needs to work with both
# since the coordinates are always relative to the 5'end of the strand
#rev is a script that reverses the order of a text string we use tr to replace the complementary base

# create revcomp of genomic loci
cat loci.seq |
 while IFS=$'\t' read def seq ; do
    echo -ne "${def}_revcomp\t"
    echo "$seq" | rev | tr [gcatGCAT] [cgtaCGTA]
done > loci.rc.seq
# verify
head loci.rc.seq

# create genomic amplicons
cat amplicon.coords |
 while IFS=$'\t' read primer locus start end ; do
  grep -h $locus loci.seq loci.rc.seq |
  tab2fasta.pl - |
  nab - $start $end
done |
 fasta2tab.pl - |
 tr -s ' ' '_' > amplicon.seq
# verify
cat amplicon.seq
wc -l amplicon.seq

# create bisY amplicons
cat amplicon.coords |
 while IFS=$'\t' read primer locus start end ; do
    grep $locus loci.bisY.seq |
     tab2fasta.pl - |
     nab - $start $end
done |
 fasta2tab.pl -|
 tr -s ' ' '_' > amplicon.bisY.seq
# verify
cat amplicon.bisY.seq
wc -l amplicon.bisY.seq

# create bisT amplicons
cat amplicon.coords |
 while IFS=$'\t' read primer locus start end ; do
    grep $locus loci.bisT.seq |
     tab2fasta.pl - |
     nab - $start $end
done |
 fasta2tab.pl -|
 tr -s ' ' '_' > amplicon.bisT.seq
echo -e 'fail\t' >>amplicon.bisT.seq
# verify
cat amplicon.bisT.seq
wc -l amplicon.bisT.seq

# verify bisT amplicons
# using amplicon as query, should see 2 hits per amplicon, unless there's overlap of the PCR products

tab2fasta.pl amplicon.bisT.seq |
 blastall -p blastn -a 4 -i stdin -d blastdb/primers.bisT -F F -m 8 -W 7 -e 1e-4 |
 cut -f1 |
 sort |
 uniq -c

# modify bisY amplicons
cat amplicon.bisY.seq |
 while IFS=$'\t' read def seq ; do
  echo -ne "$def\t"
  echo "ccccr${seq}rrccccccccc" | tr '[NY]' '[CN]'
done > amplicon.bisY.mod.seq

# verify
head amplicon.bisY.mod.seq

# create blastdb with modded bisY amplicons
tab2fasta.pl amplicon.bisY.mod.seq |
 formatdb -i stdin -o T -p F -n blastdb/ampY

# blast modded bisY amplicons to  modded amplicons db
# ... checking for NGs at 60x positions since these may result in bad calls b/c of m1 report line wrap
# issues

tab2fasta.pl amplicon.bisY.mod.seq |
 blastall -p blastn -a 4 -i stdin -d blastdb/ampY -F F -m 1 -W 7 -e 1e-2 |
 grep -i ^query |
 more "+/ n|cccr|rccc"

# adjust amplicon.bisY.mod.seq by hand,
# 1) adding Cs to the front to adjust for ' n'
# 2) adding Cs to the front to clamp front
# 3) adding Cs to the back to clamp back
# ... and repeat
# vi amplicon.bisY.mod.seq

# modify bisT amplicons
cat amplicon.bisT.seq |
 while IFS=$'\t' read def seq ; do
  echo -ne "$def\t"
  echo "ccccr${seq}rrccccccccc"
done > amplicon.bisT.mod.seq
# verify
head amplicon.bisT.mod.seq
wc -l amplicon.bisT.mod.seq

# create blastdb with modded bisT amplicons
tab2fasta.pl amplicon.bisT.mod.seq |
 formatdb -i stdin -o T -p F -n blastdb/ampT
```

```
# blast modded bisT amplicons to  modded amplicons db
# ... checking for NGs at 60x positions
tab2fasta.pl  amplicon.bisT.mod.seq  |
 blastall -p blastn -a 4 -i stdin -d blastdb/ampT -F F -m 1 -W 7 -e 1e-2 |
 grep -i ^query |
 more "+/ n|cccr|rccc"

# adjust amplicon.bisT.mod.seq by hand, adding/removing Cs from the front
# ... and repeat
#  vi  amplicon.bisT.mod.seq


# create amplicon, locus table -- weak, need better way
paste  amplicon.bisT.seq  amplicon.bisT.seq   |
 cut -f 1,3 |
 sed -e 's/_[^_]*_[^_]*$//' -e 's/_revcomp$//' \
 >  amplicon-bisT.locus
cat  amplicon-bisT.locus
wc -l  amplicon-bisT.locus

# create tag, tag5 table
## list of tags for each patient that links each 5bp tag to a single patient

cut -f4  sample.id.t-n.tag   |
 tail +2 |
 sort > tags
cat tags
wc -l tags

##May not need this...
## extend any 4-base tags to 5-base tag, i.e. CGAC{A,C,G,T}
{  grep -Ev '.{5}' tags |
    while read seq ; do
      for i in A C G T N ; do
        echo ${seq}$i
      done
    done
    grep -E '.{5}' tags
} | sort > tag5
cat tag5
wc -l tag5

# create tag, tag5 mapping - for those tags that are fewer than 5 bp
# manually trim from 5bp to 4bp
diff -y tags tag5 |
 tr -d '[|<>\]' |
 tr -s '[ \t]' '\t' \
 > tag.tag5
#  vi tag.tag5
```

To link in previously designed amplicon information:

```
ln -snf ../amp/* .
```

reads

NAME YOUR READS FOR THE EXPERIMENT BY CHANGING THIS VAR

```
read_name=[YOUR  NAMING  CONVENTION]


set -x

## When working with 454 data the reads are contained within .fna files
#one can get the reads by (try to do this so that the file is in base.files)
cat *.fna > reads.fasta

#We got our data from the WU Genome Center through collaboration as a tar file
#here we are getting the data

# get all.fasta from Wash-U
ls  -la  ~/mnt/vader/shared/Info_Systems/temp/454/orion

# gunzip
#tar -Otzvf file name will help you find the fasta (BIGG)
tar -Oxzvf orion_101306.tar.gz All* \
 > reads.fasta
head  reads.fasta
grep -c '^>' reads.fasta

#We now want to make the 454 read name more amenable to blast table parsing (i.e. small)
#We will create a relationship table that links our new name (Variable above) with the 454 name

# make tab delimited sequence file (ideal for joining)
fasta2tab.pl  reads.fasta  \
 > reads.seq
head  reads.seq
wc -l  reads.seq


# create seqid, 454id, sequence table
# seq[#] is for run of experiment, change to fit exact expt
cat -n reads.seq   |
  perl -F'\t' -lane 'printf("seq'$read_name'-%07d\t%s\t%s\n", @F)' \
  > seqid.454id.seq
head  seqid.454id.seq
wc -l  seqid.454id.seq

##First level of Q/C is to find the patient tags...keep reads with tags
# create seqid, tag table, look at frequencies of tags and total # of tags found
perl -F'\t' -lane 'print join("\t", $F[0],substr($F[2],0,5))' seqid.454id.seq \
```

```
 > seqid.tag
head  seqid.tag
wc  -l  seqid.tag
cut  -f2  seqid.tag  |  sort|uniq  -c|sort  -rn  >  seqid.tag.histo_raw
wc  -l  seqid.tag.histo_raw


# create patient info with tag5
join  -t$'\t'  -1  4  \
 <(  sort  -t$'\t'  -k  4,4  sample.id.t-n.tag  )  \
 <(  sort  tag.tag5   )  |
 sort  -k  2,2n  \
 >  tag.sample.id.t-n.tag5
cat  tag.sample.id.t-n.tag5
wc  -l  tag.sample.id.t-n.tag5


# create patient info with read
join  -t$'\t'  -1  5  -2  2  \
 <(  sort  -k  5,5  tag.sample.id.t-n.tag5)  \
 <(  sort  -k  2,2  seqid.tag)  \
 >  tag5.tag.sample.id.t-n.seqid
head  tag5.tag.sample.id.t-n.seqid
wc  -l  tag5.tag.sample.id.t-n.seqid


# create seqid, amplicon table this will take the reads with tags and find the amplicons in them

## create blastdb of forward primers
grep  F  primers.bisT.seq  |
 tab2fasta.pl  -  |
 formatdb  -i  stdin  -o  T  -p  F  -n  blastdb/f-primers
fastacmd  -I  -d  blastdb/f-primers

## pull out tag+primer

<  seqid.454id.seq  \
 perl  -F'\t'  -lane  'print  join("\t",  $F[0],substr($F[2],0,34))'  \
 >  seqid.tag+primer
head  seqid.tag+primer
wc  -l  seqid.tag+primer  seqid.454id.seq


## determine blast params by blasting tag+primer against forward primers -v 5 -b 5
tab2fasta.pl  seqid.tag+primer  |
 blastall  -a  4  -p  blastn  -i  stdin  -d  blastdb/f-primers  -F  F  -W  7  -v  1  -b  1  -m  8  \
 >  primer.bin.blast

## bin best e-value less than 0.001 and alignment start <=8
<  primer.bin.blast  \
 perl  -F'\t'  -lane  'print unless $prev eq $F[0] or $F[10] > 0.001 or $F[6] > 8 ; $prev=$F[0] ' \
 >  primer.bin.top-hit
 wc  -l  primer.bin.top-hit

## bin the results by primer
## i.e. histogram of reads per primer
cut  -f2  primer.bin.top-hit  |
 sort  |
 uniq  -c  |
 sed  -e  's/_2*F$//'  \
 >  primer.bin.top-hit.histo
cat  primer.bin.top-hit.histo
wc  -l  primer.bin.top-hit.histo


## create passed seqid, amplicon table
join  -t$'\t'  -1  2  \
 <(cut  -f1,2  primer.bin.top-hit  |  sort  -k  2,2)  \
 <(sort  primer.amplicon  )  |
 cut  -f2,3  |
 sort  >  seqid.amplicon
head  seqid.amplicon
wc  -l  seqid.amplicon


# create list of seqids that have sample tags
#
join  -t$'\t'  -2  2   \
 <(sort  tag5  )   \
 <(sort  -t$'\t'  -k  2,2  seqid.tag  )  |
 perl  -F'\t'  -lane  'print  join("\t",  @F[1,0])'  |
 sort  \
 >  seqid.tag-passed

## should keep tags to do group stats... want to make sure that all patients are represented randomly
join  -t$'\t'  -2  2  \
 <(sort  tag5  )  \
 <(sort  -t$'\t'  -k  2,2  seqid.tag  )  |
 cut  -f2  |
 sort  >  seqid-passedTags
head  seqid-passedTags
wc  -l  seqid-passedTags

# create seqid, sequence table for seqids with passedTags
join  -t$'\t'  seqid-passedTags  seqid.454id.seq  |
  cut  -f1,3  >  seqid-passedTags.seq
head  seqid-passedTags.seq
wc  -l  seqid-passedTags.seq


#Ok..we have the reads with tags and the reads with tags and aplicon sequence
#now we want to identify the reads with poor Bisulifte conversion
#these have C's remaining outside of CG motifs (works only in mammals...NOT PLANTS See MethylMapper).
#Sometimes the BisY MethylMapper control does not work because there is too much mismatch
#in templates to get a decent blast hit back to the BisY mutagenized template.   This happens
#in stretches of high CG, high C or CW homopolymeric/simple sequence.   In cases like this another control is better.
#Instead do the BisT conversion to the template and blast the reads to it, then
#look for unconverted Cs using a new script blast.m1.bisQC-norm.pl   This new script is now available
#at  http://sourceforge.net/projects/methylmapper/
```

```
# BisY filter -for the work in the manuscript we used BisY control
## create blastdb with modded bisY amplicons and seqid,sequence table (ampY-read db)
cat amplicon.bisY.mod.seq seqid-passedTags.seq |
 tab2fasta.pl - |
 formatdb -i stdin -o T -p F -n blastdb/ampY-read
fastacmd -I -d blastdb/ampY-read

## blast modded bisY amplicons to ampY-read db
## -b and -v options are necessary with blastall since the default is to report only 250 lines, make numbers big to get all t
tab2fasta.pl amplicon.bisY.mod.seq \
 > amplicon.bisY.mod.seq.fasta
blastall -a 4 -p blastn -i amplicon.bisY.mod.seq.fasta -d blastdb/ampY-read -F F -q -2 -r 3 -W 7 -m 1 -b 1000000 -v 1000000 -
 cleanBlast.pl - \

> blast.bisY.report.m1
head blast.bisY.report.m1
wc -l blast.bisY.report.m1

## normalize m1 bisY blast
blast.m1.normalize.pl blast.bisY.report.m1 2> /dev/null |
 perl -F'\t' -lane 'print unless $F[0] eq $F[1]' \
 > blast.bisY.report.m1.norm
head blast.bisY.report.m1.norm
wc -l blast.bisY.report.m1.norm
cut -f2 blast.bisY.report.m1.norm | sort | uniq | wc -l

## filter out reads that don't have a read-amplicon pair
join -t$'\t' \
 <( < seqid.amplicon tr '\t' '-' | sort ) \
 <( < blast.bisY.report.m1.norm \
     perl -F'\t' -lane 'print join("\t", $F[1] ."-".$F[0], @F)' | sort) \
> blast.bisY.report.m1.norm.read-amp
wc -l blast.bisY.report.m1.norm.read-amp
head   blast.bisY.report.m1.norm.read-amp
cut -f3 blast.bisY.report.m1.norm.read-amp | sort | uniq | wc -l

## reads with poor bis-conversion
< blast.bisY.report.m1.norm.read-amp \
 perl -F'\t' -lane 'print $F[2] if $F[5] eq "c"' |
 sort |
 uniq -c \
 > seqid.poor.bisY
head seqid.poor.bisY
wc -l seqid.poor.bisY
tr -s ' ' '\t' < seqid.poor.bisY | cut -f2 | sort -n | uniq -c > seqid.poor.bisY.hist

## Should get association of tag with poor bisY because each sample is independently mutagenized
## this means that one wants to look at rate of read loss in each patient so that exclusion
## of bad samples can be considered objectively

join -t$'\t' \
 <( perl -F'\t' -lane 'print join("\t",@F[5,0])' tag5.tag.sample.id.t-n.seqid |sort ) \
 <( perl -lane 'print join("\t",@F[1,0])' seqid.poor.bisY |sort )\
 > seqid.tag5.poor.bisY
wc -l seqid.tag5.poor.bisY
cut -f2 seqid.tag5.poor.bisY|sort|uniq -c >seqid.tag5.poor.bisY.histo
wc -l seqid.tag5.poor.bisY.histo


## create list of seqids sans poor bis-conversions
join -v 1 \
 <( cut -f1 seqid-passedTags.seq | sort) \
 <( awk '{print $2}' seqid.poor.bisY | sort) \
 > seqid.sans-bisY
head seqid.sans-bisY
wc -l seqid.sans-bisY

## create seqid,read sans bisY (these are the most interesting reads that we want to analyze)
join -t$'\t' seqid.sans-bisY <( sort seqid-passedTags.seq ) \
 > seqid-pt2.seq
head seqid-pt2.seq
wc -l seqid-pt2.seq



# Creating blast database .. seed the reads with the C-tailed amplicons
# this will ensure that the top hit for every amplicon is the C-tailed amplicon
# this is a nice blast control and helps you understand what the best E-value
# may be for a read

# BisT
# create blastdb with modded amplicons and seqid,sequence table (ampT-read db)
cat amplicon.bisT.mod.seq seqid-pt2.seq |
 tab2fasta.pl - |
 formatdb -i stdin -o T -p F -n blastdb/ampT-read
fastacmd -I -d blastdb/ampT-read


# Next when we use blast to align we want to remove multi-hsp reads.   This is because
# they likely are the result of bad sequencing ... so we want to not trust them. The way
# we approached this was to create an association between a read and only one amplicon and
# then we only use the top blast hit of a read to that one amplicon.   If a read hits an amplicon
# two times we want to ignore the data from the read.

## blast bisT amps to ampT-reads
tab2fasta.pl amplicon.bisT.mod.seq > amplicon.bisT.mod.fasta
 blastall -a 4 -p blastn -i amplicon.bisT.mod.fasta -d blastdb/ampT-read -F F -m 8 -W 7 -e 1e-8 -b 1000000 -v 1000000 \
 > blast.bisT.report.m8
head blast.bisT.report.m8
wc -l blast.bisT.report.m8

## generate list of single-hsp seqids
join -t$'\t' \
 <(perl -F'\t' -lane 'print join("_", @F)' seqid.amplicon | sort ) \
 <(perl -F'\t' -lane 'print join("\t", join("_", @F[1,0]), @F)' blast.bisT.report.m8 \
 | sort ) |
 tee seq-amp.blast.bisT.report.m8 |
 cut -f1 |
 sort |
 uniq -c |
```

```
 sort -n |
 grep '^ *1 ' |
 cut -c9- \
 > seqid-amp.single-hsp.a
head  seqid-amp.single-hsp.a
wc  -l  seqid-amp.single-hsp.a
head  seq-amp.blast.bisT.report.m8
wc  -l  seq-amp.blast.bisT.report.m8

# restrict reads that blast too far into amplicons since these are typically homopolymer anchored
join -t$'\t' \
 <(sort  seqid-amp.single-hsp.a ) \
 <(sort  seq-amp.blast.bisT.report.m8)|
 perl -F'\t' -lane 'print $F[0] unless $F[11] > 0.00000001 or $F[9] > 10;'\
 > seqid-amp.single-hsp
head  seqid-amp.single-hsp
wc  -l  seqid-amp.single-hsp

join -t$'\t' \
 <(sort  seqid-amp.single-hsp)\
 <(perl -F'\t' -lane 'print join("\t","$F[1]_$F[0]", @F)' blast.bisT.report.m8 |sort) |
 cut -f2- \
 > blast.bisT.report.m8.restricted
head  blast.bisT.report.m8.restricted
wc  -l  blast.bisT.report.m8.restricted

# you now have the really good reads in the blast report.
# We now need to get the mismatch info (i.e. methylation) from an m1 blast report.

# MethylMapper part

# blast modded amplicons to ampT-read db
tab2fasta.pl  amplicon.bisT.mod.seq  >  amplicon.bisT.mod.fasta
 blastall -a 4 -p blastn -i amplicon.bisT.mod.fasta -d blastdb/ampT-read -F F -m 1 -W 7 -e 1e-8 -b 1000000 -v 1000000 |cleanB
 > blast.bisT.report.m1
head  blast.bisT.report.m1
wc  -l  blast.bisT.report.m1

# normalize m1 bisT blast
blast.m1.normalize.pl  blast.bisT.report.m1   2> /dev/null  |
perl -F'\t' -lane 'print unless $F[0] eq $F[1]' \

>  blast.bisT.report.m1.norm
head  blast.bisT.report.m1.norm
wc  -l  blast.bisT.report.m1.norm
cut -f2 blast.bisT.report.m1.norm | sort | uniq | wc -l

# filter for reads that have single hsps and have a read-amplicon pair
join -t$'\t' \
   seqid-amp.single-hsp \
   <( < blast.bisT.report.m1.norm \
      perl -F'\t' -lane 'print join("\t", "$F[1]_$F[0]", @F)' | sort) \
>  blast.bisT.report.m1.norm.read-amp
wc  -l  blast.bisT.report.m1.norm.read-amp
head   blast.bisT.report.m1.norm.read-amp
cut -f3 blast.bisT.report.m1.norm.read-amp | sort | uniq | wc -l

# put all the pieces together into one big table
# we chose a table b/c we could use pivot tables in excel to get data views with cell format colors
# reflecting the methylated (1) or unmethylated (0) values
# The other advantage of a table is that allows SAS/R to query the data easily.

# create normalized table: amp, read, position, base, U/M
cut  -f2,3,4,6,7  blast.bisT.report.m1.norm.read-amp  |
 sort  -t$'\t'  -k  2,2  >  amp.read.pos.base.um
cut -f2 amp.read.pos.base.um | sort | uniq | wc -l

# create pre-big table
join -t$'\t' -2  2 seqid.tag  amp.read.pos.base.um \
 >  seqid.tag5.amp.pos.base.um
wc  -l  seqid.tag5.amp.pos.base.um
numer=$(cut -f1 seqid.tag5.amp.pos.base.um | uniq | wc -l)
echo "$numer/274908*100" = $(echo "$numer/274908*100" | bc -l)

#use e-value to filter out bad reads... here's an example of differet e-value effects upon fractionn
#of reads in a 1/4 plate run on an FLX...

# @ bisT blast 1e-8 : 61656/72923*100 = 84.549456275797759280
# @ bisT blast 1e-15 : 60539/72923*100 = 83.01770360517257929500
# @ bisT blast 1e-20 : 59206/72923*100 = 81.18974809045157220600
# @ bisT blast 1e-40 : 43236/72923*100 = 59.28993595984805891100

# create pre2-big table by joining in sample info
join -t$'\t' -1 5 -2 2 \
 <(sort -t$'\t' -k 5,5 tag.sample.id.t-n.tag5 ) \
 <( sort -t$'\t' -k 2,2 seqid.tag5.amp.pos.base.um ) \
 >  tag5.tag.sample.id.t-n.seqid.amp.pos.base.um
head  tag5.tag.sample.id.t-n.seqid.amp.pos.base.um
wc  -l  tag5.tag.sample.id.t-n.seqid.amp.pos.base.um
wc  -l  seqid.tag5.amp.pos.base.um

# create pre3-big table by joining in locus info
join -t$'\t' -1 1 -2 7 \
 <(sort -t$'\t' -k 1,1 amplicon-bisT.locus ) \
 <( sort -t$'\t' -k 7,7 tag5.tag.sample.id.t-n.seqid.amp.pos.base.um ) \
 >  amp.locus.tag5.tag.sample.id.t-n.seqid.pos.base.um
head  amp.locus.tag5.tag.sample.id.t-n.seqid.pos.base.um
wc  -l  amp.locus.tag5.tag.sample.id.t-n.seqid.pos.base.um

# create big table by rearranging fields and sorting... Now the stats folks have a data table
{
echo  -e  "tag\tsample\tpatient\tT-N\tlocus\tamplicon\tread\tposition\tbase\tmeth-unmeth"

<  amp.locus.tag5.tag.sample.id.t-n.seqid.pos.base.um  \
 perl -F'\t' -lane 'print join("\t",  @F[3,4,5,6,1,0,7,8,9,10])' |
 sort -t$'\t' -k 2,2n -k 6,6 -k 8,8n
} > big.table.tsv
head  big.table.tsv
wc  -l  big.table.tsv
```

```
#Simple checks of the data by command line querry...

#tag representation in big table across multiple lines
cut -f1 big.table.tsv | sort| uniq -c > big.table.tag.histo
head big.table.tag.histo
wc -l big.table.tag.histo

#tag representation in big table by read
cut -f1,7 big.table.tsv > big.table.tag.read
wc -l big.table.tag.read
cut -f1,2 big.table.tag.read| sort -k2|uniq -c >big.table.tsv.tag.read.collapse
wc -l big.table.tsv.tag.read.collapse
  <big.table.tsv.tag.read.collapse tr -s ' ' '\t' > big.table.tsv.tag.read.collapse.tab
cut -f3 big.table.tsv.tag.read.collapse.tab|sort| uniq -c >big.table.tag.abundace
cat big.table.tag.abundace

#locus representation in big table by read
cut -f5,7 big.table.tsv > big.table.loc.read
wc -l big.table.loc.read
cut -f1,2 big.table.loc.read| sort -k2|uniq -c >big.table.loc.read.collapse
wc -l big.table.loc.read.collapse
<big.table.loc.read.collapse tr -s ' ' '\t' > big.table.loc.read.collapse.tab
cut -f3 big.table.loc.read.collapse.tab|sort| uniq -c >big.table.loc.abundace
cat big.table.loc.abundace # will be 1 too high because of header

#amplicon representation in big table by read
cut -f6,7 big.table.tsv > big.table.amp.read
wc -l big.table.amp.read
cut -f1,2 big.table.amp.read| sort -k2|uniq -c >big.table.amp.read.collapse
wc -l big.table.amp.read.collapse
  <big.table.amp.read.collapse tr -s ' ' '\t' > big.table.amp.read.collapse.tab
cut -f3 big.table.amp.read.collapse.tab|sort| uniq -c >big.table.amp.abundace
cat big.table.amp.abundace
wc -l big.table.amp.abundace #will be one too high b/c of header

# create relative and absolute methylation postion table
CountBisulfite.pl blast.bisT.report.m1 |
 perl -lane '
    if (/Position/) { $foo=$F[0] ;next; } ;
    $F[0] =~ s/NG// ;
    print join("\t", $foo, @F[0,1])' \
 > amp.meth-rel_pos.meth-abs_pos
```

qc 2

```
# Seq Quality Table
#                 Patient          Primer  BS Conversion                      Alignment information
# Seq ID          Seq L   Tag ID  Amplicon ID      Bis Y Test       Bis T test       Amplicon Length Alignment Length         A_Sta
#                 0 = no tag found         0 = no primer found     0 = fail         0 = fail
#                 (1-23) = tag ID (1-14) = primer ID       1 = pass         1 = pass
#

# orig fields
      1  Seq ID
      2  Seq L
      3  Tag ID (patient)
      4  Amplicon ID
      7  Amplicon Length
      5  Bis Y Test
      6  Bis T test
      8  Alignment Length
      9  A_Start
     10  A_End
     11  R_Start
     12  R_End

# latest fields
      1  Seq ID
      2  Seq L
      3  Sample ID (patient)
      4  Patient ID
      5  Amplicon ID
      6  Amplicon Length
      7  Bis Y Test
      8  Bis T test
      9  Alignment Length
     10  A_Start
     11  A_End
     12  R_Start
     13  R_End
     14  Amplicon ID (old)
     15  pct_id
     16  mismatches
     17  gaps
     18  E-value
     18  bit_score

# all reads seqid
cut -f1 seqid.454id.seq > seqid
wc -l seqid


# seqid, length table
cut -f1,3 seqid.454id.seq |
 tab2fasta.pl - |
 length - \
 > seqid.length
tail seqid.length
wc -l seqid.454id.seq


# seqid, patient table
##  those that passed
join -t$'\t' \
 <(sort seqid-passedTags) \
```

```
 <(sort seqid.tag) \
 > seqid.tag5.passed
head seqid.tag5.passed
wc -l seqid.tag seqid-passedTags seqid.tag5.passed

## those that didn't pass
join -t$'\t' -v 1 <(sort seqid.tag) <(sort seqid-passedTags) |
 sed -re 's/\t.*$/\tNNNNN/' \
 > seqid.tag5.passed.not
head seqid.tag5.passed.not
wc -l seqid.tag seqid-passedTags seqid.tag5.passed.not

## join passed and not passed with patient number and patient ID
join -t$'\t' -1 5 -2 2 \
 <(sort -k 5,5 tag.sample.id.t-n.tag5 ) \
 <( sort -t$'\t' -k 2,2 seqid.tag5.passed seqid.tag5.passed.not) |
 perl -F'\t' -lane 'print join("\t", @F[5,2,3])' |
 sort \
 > seqid.sample
head seqid.sample
wc -l seqid.tag seqid.sample


# seqid, amplicon ID, length
## combine those with and without an amplicon
{
join -t$'\t' -o 1.1,2.1 -1 2 -2 2 \
 <( sort -k 2,2 seqid.amplicon ) \
 <( sort -k 2,2 amplicon.new-old )
join -t$'\t' -v 1 \
 <( cut -f1 seqid.454id.seq | sort ) \
 <( cut -f1 seqid.amplicon | sort ) |
 sed -e 's/$/\t0/'
} | sort \
 > seqid.ampid
head seqid.ampid
wc -l seqid.ampid
cut -f1 seqid.ampid | sort | uniq | wc -l

join -t$'\t' \
 <(perl -F'\t' -lane 'print join("\t", $F[1], @F)' seqid.ampid | sort ) \
 <(paste \
  <( sort -k 2,2 amplicon.new-old) \
  <(sort amplicon.bisT.seq | tab2fasta.pl - | length - ) |
  cut -f 1,2,4 | sort) |
 cut -f,2,3,5 |
 sort \
 > seqid.ampid.length
head seqid.ampid.length
wc -l seqid.ampid.length

# seqid, bisY pass/fail
{ sed -e 's/$/\t1/' seqid.sans-bisY
  join -t$'\t' -v 1 \
   <( cut -f1 seqid.454id.seq | sort ) \
   <( sort seqid.sans-bisY ) |
  sed -e 's/$/\t0/'
} | sort \
 > seqid.bisY
head seqid.bisY
wc -l seqid.bisY

## generate list
{ cut -f2 blast.bisT.report.m8.restricted |
  sed -e 's/$/\t1/'
  join -t$'\t' -v 1 \
   <( cut -f1 seqid.454id.seq | sort ) \
   <( cut -f2 blast.bisT.report.m8.restricted | sort ) |
  sed -e 's/$/\t0/'
} | sort \
 > seqid.bisT
head seqid.bisT
wc -l seqid.bisT

# seqid, blast report fields

< blast.bisT.report.m8.restricted \
 perl -F'\t' -lane 'print join("\t", @F[1,3,6..9,0,2,4,5,10,11])' \
 > seqid.blast
wc -l seqid.blast
head seqid.blast

#########
# joining all the fields together

sort seqid |
 join -t$'\t' -a 1 - <( sort seqid.length ) |
 join -t$'\t' -a 1 - <( sort seqid.sample ) |
 join -t$'\t' -a 1 - <( sort seqid.ampid.length) |
 join -t$'\t' -a 1 - <( sort seqid.bisY) |
 join -t$'\t' -a 1 - <( sort seqid.bisT) |
 join -t$'\t' -a 1 - <( sort seqid.blast) \
 > seqid.length.sample.ampid.amplength.bisY.bisT.blast
head seqid.length.sample.ampid.amplength.bisY.bisT.blast
wc -l seqid.length.sample.ampid.amplength.bisY.bisT.blast seqid
\cp seqid.length.sample.ampid.amplength.bisY.bisT.blast big.QC.table.tsv
```