Analysis of ChIP-seq data from

Sulfolobus acidocaldarius DSM 639

1 Get Ready

For this exercise we will use Bowtie to align the reads. Bowtie is available as a module on Abel, so we just have to load it. Bowtie is for readlengths <~50 and Bowtie2 is for readlengths >~50. Our reads are ~50 basepairs long, so each one of them would be ok.

```
module load bowtie
```

You can find the data for these exercises in /work/projects/chip/data, and the programs in /work/projects/chip/programs. Make your own chip directory inside the workflows directory, cd into it and copy the two directories there. To make the programs runnable, use chmod

```
cp –r /work/projects/norbis/workflow/chip/data .
cp –r /work/projects/norbis/workflow/chip/programs .

chmod a+rx programs/*
```

2 Alignment

You now have the reference genome sequence as a fasta file in the data directory, and can cd into the data directory and make a Bowtie index like this:

```
bowtie-build -f data/saci.fasta data/saci
```

The alignment can be parallelized with the –p option.

```
bowtie -p 7 -S -v 2 -m 1 data/saci data/N1_CHIP.fastq > N1_CHIP.sam

bowtie -p 7 -S -v 2 -m 1 data/saci data/N1_INPUT.fastq > N1_INPUT.sam

bowtie -p 7 -S -v 2 -m 1 data/saci data/T1_CHIP.fastq > T1_CHIP.sam

bowtie -p 7 -S -v 2 -m 1 data/saci data/T1_INPUT.fastq > T1_INPUT.sam
```

Option	Explanation
-p 7	Launch 7 parallel search threads
-S	Output in SAM format

-v 2	No more than 2 mismatches
-m 1	Suppress all alignments for a read if more than 1 reportable alignments exist

Write down the number of aligned reads, and compute the size normalization factor. We will use these numbers later.

	# of reads	# of aligned reads	Size normalization factors #of aligned reads/1000000
N1 CHIP			
N1 INPUT			
T1 CHIP			
T1 INPUT			

3 Making bigwig files:

Resulting SAM files can be very big, so for some purposes, like uploading the files to view them in a browser, the bigwig format may be useful. Description of many kinds of file formats can be found here:

https://genome.ucsc.edu/FAQ/FAQformat.html

Making bigwig files from the alignment files requires several steps that are described below. Some of the steps require samtools and bedtools, so these have to be loaded first.

```
module load samtools
module load bedtools
```

Getting BAM file from SAM file:

```
samtools view N1_CHIP.sam -bS > N1_CHIP.bam
```

samtools view T1_CHIP.sam -bS > T1_CHIP.bam

samtools view N1_INPUT.sam -bS > N1_INPUT.bam

samtools view T1_INPUT.sam -bS > T1_INPUT.bam

Option	Explanation
-b	Output is BAM format
-S	Input in SAM format

Sorting the BAM file with samtools:

The next step (genomeCoverageBed) requires a sorted file as input. This is done with samtools sort.

samtools sort N1_INPUT.bam N1_INPUT_sorted

samtools sort T1_INPUT.bam T1_INPUT_sorted

samtools sort T1_CHIP.bam T1_CHIP_sorted

samtools sort N1_CHIP.bam N1_CHIP_sorted

Getting bedGraph files with genomeCoverageBed:

At this step, a bedGraph file is made from the sorted BAM.

genomeCoverageBed -ibam N1_CHIP_sorted.bam -bg -g ../sacisizes.txt > N1_CHIP.bedGraph

genomeCoverageBed -ibam T1_CHIP_sorted.bam -bg -g ../sacisizes.txt > T1_CHIP.bedGraph

genomeCoverageBed -ibam T1_INPUT_sorted.bam -bg -g ../sacisizes.txt > T1_INPUT.bedGraph

genomeCoverageBed -ibam N1_INPUT_sorted.bam -bg -g ../sacisizes.txt > N1_INPUT.bedGraph

Option	Explanation
-ibam	Input file in BAM format
-bg	Report bedGraph depth
-g	File with chromosome sizes

Sorting bedGraph files based on column 1, then column 2:

The next step (bedGraphToBigWig) needs sorted input. This is done with unix sort. The file is first sorted by column1, then by column 2 (n – numerically).

```
sort -k1,1 -k2,2n N1_INPUT.bedGraph > N1_INPUT_sorted.bedGraph
sort -k1,1 -k2,2n T1_INPUT.bedGraph > T1_INPUT_sorted.bedGraph
sort -k1,1 -k2,2n T1_CHIP.bedGraph > T1_CHIP_sorted.bedGraph
sort -k1,1 -k2,2n N1_CHIP.bedGraph > N1_CHIP_sorted.bedGraph
```

Making bigWig files from bedGraph:

```
programs/bedGraphToBigWig -unc N1_INPUT_sorted.bedGraph data/sacisizes.txt N1_INPUT_bigWig.bwig
programs/bedGraphToBigWig -unc T1_INPUT_sorted.bedGraph data/sacisizes.txt T1_INPUT_bigWig.bwig
programs/bedGraphToBigWig -unc T1_CHIP_sorted.bedGraph data/sacisizes.txt T1_CHIP_bigWig.bwig
programs/bedGraphToBigWig -unc N1_CHIP_sorted.bedGraph data/sacisizes.txt N1_CHIP_bigWig.bwig
```

Option	Explanation
-unc	Not using compression

The program needs the size of each chromosome as input, these are listed in the file sacisizes.txt. This file is in the data folder.

4 Peak calling:

There are a lot of peak callers out there. We will use MACS 14 to do the peak calling. This is available as a module on Abel.

```
module load macs
```

The commands for calling the peaks are as follows:

```
macs14 -t N1_CHIP.sam -c N1_INPUT.sam  -n N1 -g 2.23e6 -f SAM
macs14 -t T1_CHIP.sam -c T1_INPUT.sam  -n T1 -g 2.23e6 -f SAM
```

Option	Explanation
-t	Treatment file

-c	Control file
-n	Experiment name
-g	Effective genome size
-f	Format

Find the number of peaks for each sample and write them down

Samples	# of peaks
N1	
T1	

Check this link if you want an overview of command line options and a description of the output files from MACS:

http://liulab.dfci.harvard.edu/MACS/00README.html

Finding read counts for each peak

Getting one file with all the peaks and sorting it:

```
cat N1_peaks.bed T1_peaks.bed > N1_T1_peaks.bed
sort -k1,1 -k2,2n N1_T1_peaks.bed > N1_T1_sorted_peaks.bed
```

Running bedtools merge for peaks found in N1 and T1 to have the complete set for comparison:

```
bedtools merge -i N1_T1_sorted_peaks.bed > N1_T1_merged_peaks.bed
```

Running intersectBed to find all reads that intersect with the peaks:

```
intersectBed -bed -wb -abam N1_CHIP_sorted.bam -b N1_T1_merged_peaks.bed > N1_CHIP_intersect.txt
intersectBed -bed -wb -abam T1_CHIP_sorted.bam -b N1_T1_merged_peaks.bed > T1_CHIP_intersect.txt
intersectBed -bed -wb -abam N1_INPUT_sorted.bam -b N1_T1_merged_peaks.bed > N1_INPUT_intersect.txt
intersectBed -bed -wb -abam T1_INPUT_sorted.bam -b N1_T1_merged_peaks.bed > T1_INPUT_intersect.txt
```

Using cut to get out the columns for chr, start and end for the peaks. Sorting and counting them to find the number of reads that overlap them:

```
cut -f13,14,15 N1_CHIP_intersect.txt | sort|uniq -c > N1_CHIP_ib_count.txt
cut - f13,14,15 T1_CHIP_intersect.txt | sort|uniq -c > T1_CHIP_ib_count.txt
cut - f13,14,15 T1_INPUT_intersect.txt | sort|uniq -c > T1_INPUT_ib_count.txt
cut - f13,14,15 N1_INPUT_intersect.txt | sort|uniq -c > N1_INPUT_ib_count.txt
```

Using awk to make a format R reads properly, with no spaces between chr, start and end:

```
awk '{print $2"_"$3"_"$4"\t"$1}' N1_CHIP_ib_count.txt > N1_CHIP_countR.txt
awk '{print $2"_"$3"_"$4"\t"$1}' T1_CHIP_ib_count.txt > T1_CHIP_countR.txt
awk '{print $2"_"$3"_"$4"\t"$1}' T1_INPUT_ib_count.txt > T1_INPUT_countR.txt
awk '{print $2"_"$3"_"$4"\t"$1}' N1_INPUT_ib_count.txt > N1_INPUT_countR.txt
```

Normalization, comparing N1 and T1 and making plots in R:

#Reading in the files

```
N1cCHIP<-read.table("N1_CHIP_countR.txt", sep="\t")
T1cCHIP<-read.table("T1_CHIP_countR.txt", sep="\t")
T1cINPUT<-read.table("T1_INPUT_countR.txt", sep="\t")
N1cINPUT<-read.table("N1_INPUT_countR.txt", sep="\t")
```

#Setting the column names

```
colnames(N1cCHIP)=c("peak", "N1cCHIPcount")
colnames(T1cCHIP)=c("peak", "T1cCHIPcount")
colnames(T1cINPUT)=c("peak", "T1cINPUTcount")
colnames(N1cINPUT)=c("peak", "N1cINPUTcount")
```

#Merging the four count tables into one, and setting missing values (no reads) to 0.

```
readcounts <- merge(N1cCHIP, N1cINPUT, by="peak", ALL=T)
readcounts <- merge(readcounts,T1cCHIP, by="peak", ALL=T)
readcounts <- merge(readcounts,T1cINPUT, by="peak", ALL=T)
readcounts[is.na(readcounts)]=0
```

#Adding four columns with size normalization counts (calculation of normalization factor is shown in #the Alignment section).

```
readcounts$N1cCHIPsizenorm<-readcounts$N1cCHIPcount/1.21
readcounts$N1cINPUTsizenorm<-readcounts$N1cINPUTcount/1.02
readcounts$T1cCHIPsizenorm<-readcounts$T1cCHIPcount/1.12
readcounts$T1cINPUTsizenorm<-readcounts$T1cINPUTcount/1.34
```

#Importing the package preprocessCore (used for quantile normalization)

```
library(preprocessCore)
```

#Getting a table with the CHIP counts for normalization

```
fornorm<-cbind(readcounts$N1cCHIPcount, readcounts$T1cCHIPcount)
```

#Normalizing

```
normcounts<-normalize.quantiles(as.matrix(fornorm), copy=TRUE)
```

#Adding two columns with the normalized counts in the readcount table

```
readcounts$N1cCHIPnorm<-normcounts[,1]
readcounts$T1cCHIPnorm<-normcounts[,2]
```

#Adding columns with the logfold change values for each of the three CHIP-counts

```
readcounts$logfoldChangeRaw <- log2( (readcounts$N1cCHIPcount+1)/(readcounts$T1cCHIPcount+1) )
readcounts$logfoldChangeSnorm <- log2( (readcounts$N1cCHIPsizenorm+1)/(readcounts$T1cCHIPsizenorm+1) )
readcounts$logfoldChangeQnorm<- log2( (readcounts$N1cCHIPnorm+1)/(readcounts$T1cCHIPnorm+1) )
```

#Sorting the readcount table based on logfoldChange for the quantile normalized counts and write it #to file

```
ordered_readcounts <- readcounts[order(readcounts$logfoldChangeQnorm),]

write.table(ordered_readcounts, "ordered_readcounts.txt", sep="\t", quote=FALSE, row.names=F)
```

#Plot CHIP vs INPUT for N1 using size normalized counts

```
pdf("readcount_plot_N1c.pdf")
plot(log2(readcounts$N1cCHIPsizenorm+1), log2(readcounts$N1cINPUTsizenorm+1), main="Size normalized readcounts N1c", xlab="CHIP", ylab="INPUT", pch=16, xlim=c(1,12), ylim=c(1,12))
abline(0,1,col="red")
dev.off()
```

#Plot CHIP vs INPUT for T1 using size normalized counts

```
pdf("readcount_plot_T1c.pdf")
plot(log2(readcounts$T1cCHIPsizenorm+1), log2(readcounts$T1cINPUTsizenorm+1), main="Size normalized readcounts T1c", xlab="CHIP", ylab="INPUT", pch=16, xlim=c(1,12), ylim=c(1,12))
abline(0,1,col="red")
dev.off()
```

#Plot the quantile normalized counts of N1c vs T1c

```
pdf("readcount_plot_CHIP_N1c_T1c.pdf")
plot(log2(readcounts$N1cCHIPnorm+1), log2(readcounts$T1cCHIPnorm+1), main="N1c vs T1c", xlab="N1c log2(normalized counts)", ylab="T1c log2(normalised counts)", pch=16, xlim=c(1,13), ylim=c(1,13))
abline(0,1,col="red")
dev.off()
```

#Plot histograms of logfoldChange for raw count, size normalized count and quantile normalized #counts

```
pdf("histogram_logfoldChange_quantile_normalized_counts.pdf")
hist(readcounts$logfoldChangeQnorm, breaks=60, col="yellow", main="log2( (N1cCHIPnorm+1)/(T1cCHIPnorm+1) )")
abline(v=0,  col="red", lwd=2)
dev.off()

pdf("histogram_logfoldChange_size_normalized_counts.pdf")
hist(readcounts$logfoldChangeSnorm, breaks=60, col="yellow", main="log2( (N1cCHIPsizenorm+1)/(T1cCHIPsizenorm+1) )")    abline(v=0,  col="red", lwd=2)
dev.off()

pdf("histogram_logfoldChange_raw_counts.pdf")
hist(readcounts$logfoldChangeRaw, breaks=60, col="yellow", main="log2( (N1cCHIPcounts+1)/(T1cCHIPcounts+1) )")
abline(v=0,  col="red", lwd=2)
dev.off()
```