Normalization Example using spike-in control as a standard

Andrew Holding

8/7/2017

Introduction

This is a normalisation using H2av binding within an external spike-in of droshilia chromatin to provide a control for the analysis of ER binding in MCF7 cells after treatment with Fulvestrant.

This example builds on the previous two and therefore they should be read first as many details of key functions have not been repeated.

Preprocessing

Unlike Example 001 & 002 the use of a xenogenic spike in requires more pre-processing. The scripts for this process are provided below. Genomes were downloaded from http://support.illumina.com/sequencing/sequencing_software/igenome.html.

The first step of our preprossing pipeline requires the generation of a combined Human/Drosophila genome. We used the following script to generated this. The human and drosophila genomes downloaded from the link provided above are contained in a subdirectory "genomes" relative to where the script was run. These example scripts are not evalutated in this markdown due to the time it takes to process the data.

```
# Merge the genomes

human=../Homo_sapiens/UCSC/hg19/Sequence/WholeGenomeFasta/genome.fa
droso=../Drosophila_melanogaster/UCSC/dm3/Sequence/WholeGenomeFasta/genome.fa


mkdir genomes/dmhs
cd ./genomes/dmhs
sed "s/^>/>hs_/" $human > tmp
sed "s/^>/>dm_/" $droso > tmp2
cat tmp tmp2 > dmhs.fa
rm tmp tmp2


# Merge Annotations
human=../Homo_sapiens/UCSC/hg19/Annotation/Archives/archive-current/Genes/genes.gtf
droso=../Drosophila_melanogaster/UCSC/dm3/Annotation/Archives/archive-current/Genes/genes.gtf

sed "s/^/hs_/" $human > tmp
sed "s/^/dm_/" $droso > tmp2

cat tmp tmp2 > dmhs.gtf
rm tmp tmp2
```

To allow use to use the merged genome we first indexed it using bowtie2. For our system we specified 32 threads, this will depend on the number of processors you have available.

```
### Index the merged genome
cd ./genomes/dmhs

bowtie2-build --threads 32 dmhs.fa dmhs
```

Once indexed, we could the aligned to the merge genome. This script assumes that the FastQ files are in a directory ./SLX-8047_dhms from where the scripts are run.

```
#Align, Sort and Build indeces

mkdir ./SLX-8047_dmhs
cd ./SLX-8047_dmhs

genome=../genomes/dmhs/dmhs
for fq in ../SLX-8047/*fq.gz
do
  root=`basename $fq`
  bowtie2 -p 32 -x $genome -U $fq > tmp.sam \
  && samtools view -Sbh tmp.sam > tmp.bam \
  && samtools sort tmp.bam ${root}  \
  && samtools index ${root}.bam
done
rm tmp.sam tmp.bam
```

Blacklisting requires a merged blacklist. This script downloads the blacklist and merges it, then applies this to the alignment.

```
mkdir blacklists
cd blacklists
# Human hg19/GRCh37
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDacMapabilityC

# Drosophila dm3
wget http://www.broadinstitute.org/~anshul/projects/fly/blacklist/dm3-blacklist.bed.gz --no-check-certi
gunzip dm3-blacklist.bed.gz

### Generate a merged Drosophila/Human blacklist
droso=./dm3-blacklist.bed
homo=./hg19-blacklist.bed

sed "s/^/hs_/" $homo |cut -f1-3 > tmp
sed "s/^/dm_/" $droso > tmp2
cat tmp tmp2 > dmhs-blacklist.bed
rm tmp tmp2

cd ..

bl=../blacklists/dmhs-blacklist.bed


cd ./SLX-8047_dmhs

mkdir ./blacklist_filtered
for f in *.bam
do
```

```
    echo $f
    bedtools intersect -v -abam $f -b $bl  > blacklist_filtered/$f
done

### Re-index

cd ./blacklist_filtered
for f in *.bam
do
  samtools index $f
done
```

Peak calling of the reads aligned to the merged genome is then similar to that of a typical experiment.

```
### Run macs on the blacklisted data
mkdir ./SLX-8047_dmhs/peaks
cd ./SLX-8047_dmhs/peaks
control=../blacklist_filtered/SLX-8047.D705_D507.C81G5ANXX.s_1.r_1.fq.gz.bam

for bam in ../blacklist_filtered/*bam
do
  root=`basename $bam .bam`
  macs2 callpeak -t $bam -c $control -f BAM -n $root -g hs
done
```

After aligning the data can calling peaks we can then split the data back into Drosphila and Human.

```
# The BAMs and BEDs should be split into Drosophila (for normalization) and into Human (for experimenta



### BAMs, split in two
cd ./SLX-8047_dmhs/blacklist_filtered

mkdir human
mkdir drosophila

for bam in *bam
do
  samtools view -h $bam | grep -v "dm_chr" | sed s/hs_chr/chr/g | samtools view -bS - > human/$bam
  samtools view -h $bam | grep -v "hs_chr" | sed s/dm_chr/chr/g | samtools view -bS - > drosophila/$bam
done

# Index
cd ./human
echo `pwd`

for bam in *bam
do
  samtools index $bam
done


cd ../drosophila
echo `pwd`
```

```
for bam in *bam
do
  samtools index $bam
done


### Peaks BEDs and XLSs, split in two
cd ../../peaks

echo `pwd`

mkdir human
mkdir drosophila

for bed in *peaks.bed
do
  grep -v "dm_chr" $bed | sed s/hs_chr/chr/g > human/$bed
  grep -v "hs_chr" $bed | sed s/dm_chr/chr/g > drosophila/$bed
done

for xls in *peaks.xls
do
  grep -v "dm_chr" $xls | sed s/hs_chr/chr/g > human/$xls
  grep -v "hs_chr" $xls | sed s/dm_chr/chr/g > drosophila/$xls
done

for narrow in *narrowPeak
do
  grep -v "dm_chr" $narrow | sed s/hs_chr/chr/g > human/$narrow
  grep -v "hs_chr" $narrow | sed s/dm_chr/chr/g > drosophila/$narrow
done
```

With the peak files and the seperate alignments extracted we are can then write samplesheets. For the normalization we need one for the drosphila aligned reads from each sample and the respective H2av peak file, and one for human reads from each sample and their repsective ER peak files. This forms the bais of the following analysis.

Load convience functions

These functions facilitate the normalisation of data.

```
source('../package/brundle.R')
```

Apply settings

```
dbaSummits                <- 200
jg.controlMinOverlap      <- 5
jg.controlSampleSheet     <- "samplesheet/samplesheet_SLX8047_dm.csv"
jg.experimentSampleSheet  <- "samplesheet/samplesheet_SLX8047_hs.csv"
jg.treatedCondition       =  "Fulvestrant"
jg.untreatedCondition     =  "none"
```

Load control and experimental DiffBind object

As before to keep file size down data are provided as in Rdata format rather than as raw BAM files.

```
filename<-"Rdata/example_003_SLX-8047_dba_HsDm.rda"
if(!file.exists(filename)){
  dbaExperiment <- jg.getDba(jg.experimentSampleSheet, bRemoveDuplicates=TRUE)
  dbaControl    <- jg.getDba(jg.controlSampleSheet, bRemoveDuplicates=TRUE)
  save(dbaExperiment,dbaControl,file=filename)
} else {
  load(filename)
}
```

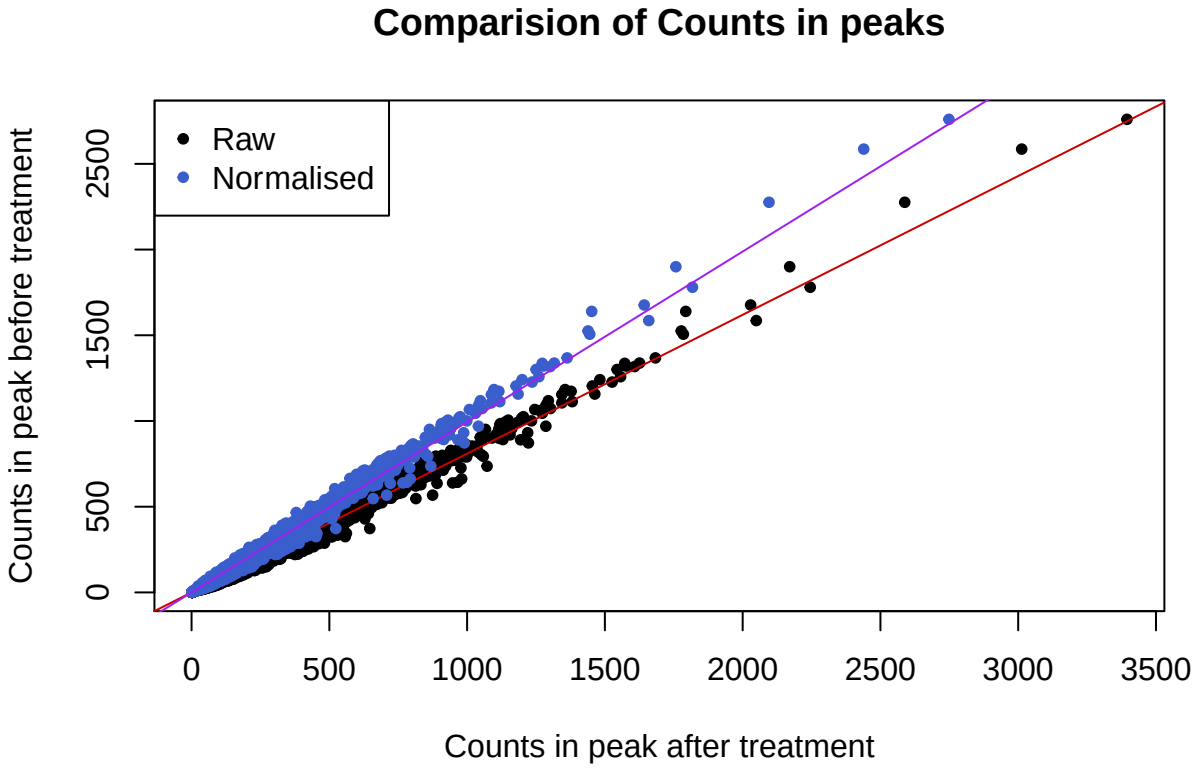
```
#Load Sample Ids from control sample sheet.
jg.sampleIds <- jg.getSampleIds(jg.controlSampleSheet)
```

```
# Extract Peak set from DiffBind
jg.experimentPeakset <- jg.dbaGetPeakset(dbaExperiment)
jg.controlPeakset    <- jg.dbaGetPeakset(dbaControl)
```

```
#Get counts for each condition
jg.controlCountsTreated<-jg.getControlCounts(jg.controlPeakset,
                                             jg.controlSampleSheet,
                                             jg.treatedCondition)
jg.controlCountsUntreated<-jg.getControlCounts(jg.controlPeakset,
                                               jg.controlSampleSheet,
                                               jg.untreatedCondition)
```

```
#Get sample names for conditions
jg.untreatedNames <- names(jg.controlCountsUntreated)
jg.treatedNames   <- names(jg.controlCountsTreated)
```

```
##Plot showing normalization calculation (Optional)
  jg.plotNormalization(jg.controlCountsTreated,
                       jg.controlCountsUntreated)
```




```
## rowMeans(jg.controlCountsTreated)
##                           0.8096594
```

```
##Get Normalization Coefficient
jg.coefficient<-jg.getNormalizationCoefficient(jg.controlCountsTreated,
                                               jg.controlCountsUntreated)
jg.correctionFactor<-jg.getCorrectionFactor(jg.experimentSampleSheet,
                                            jg.treatedNames,
                                            jg.untreatedNames)
```

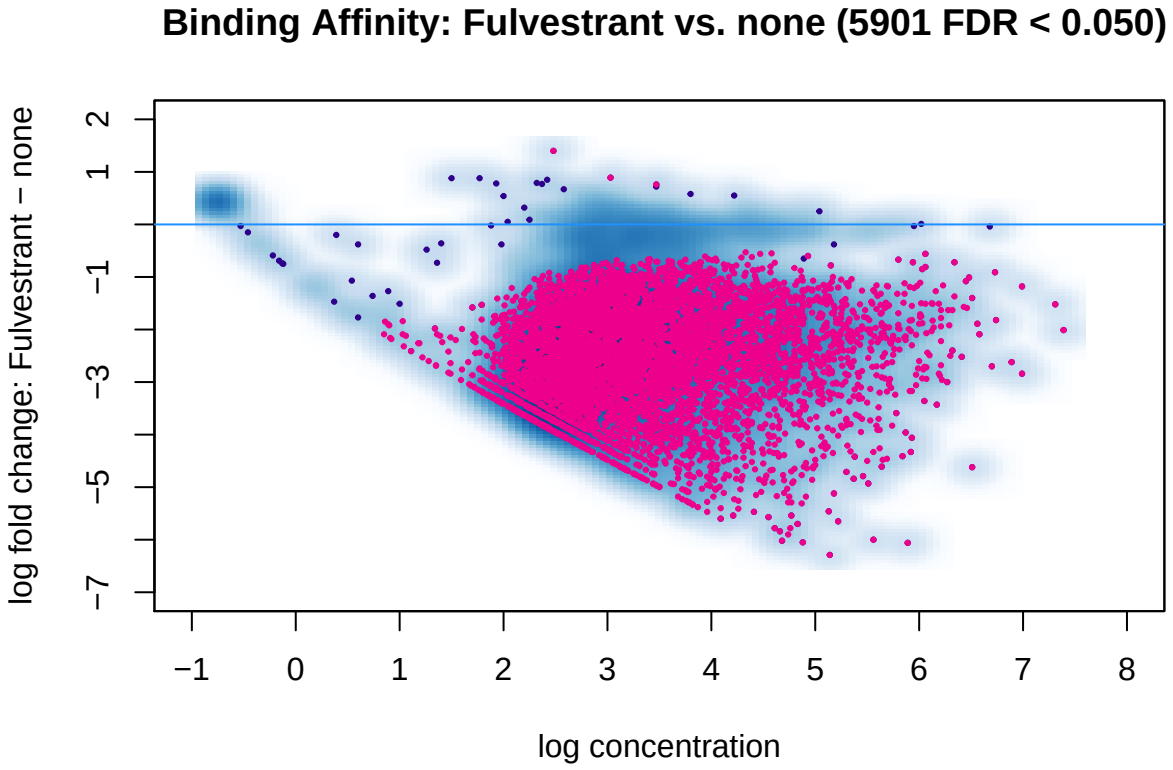
```
#Apply coefficent and control factor
jg.experimentPeaksetNormalised<-jg.applyNormalisation(jg.experimentPeakset,
                                                      jg.coefficient,
                                                      jg.correctionFactor,
                                                      jg.treatedNames)
```

```
#Return values to Diffbind and plot normalised result.
jg.dba <- DiffBind:::pv.resetCounts(dbaExperiment,
                                    jg.experimentPeaksetNormalised)
```

```
jg.dba_analysis<-dba.analyze(jg.dba)
```

```
## converting counts to integer mode
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
```

```
dba.plotMA(jg.dba_analysis)
```



Save results

```
write.csv(dba.report(jg.dba_analysis),file="results/Example_003.csv")
```