Normalization Example using spike-in control as a standard

Andrew Holding

8/7/2017

Introduction

This is a normalisation using linear regression to CTCF of MCF7 cells chip treameant etc.

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

To keep file size down these are provided as a Rdata File rather than as raw counts.

```
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)
```

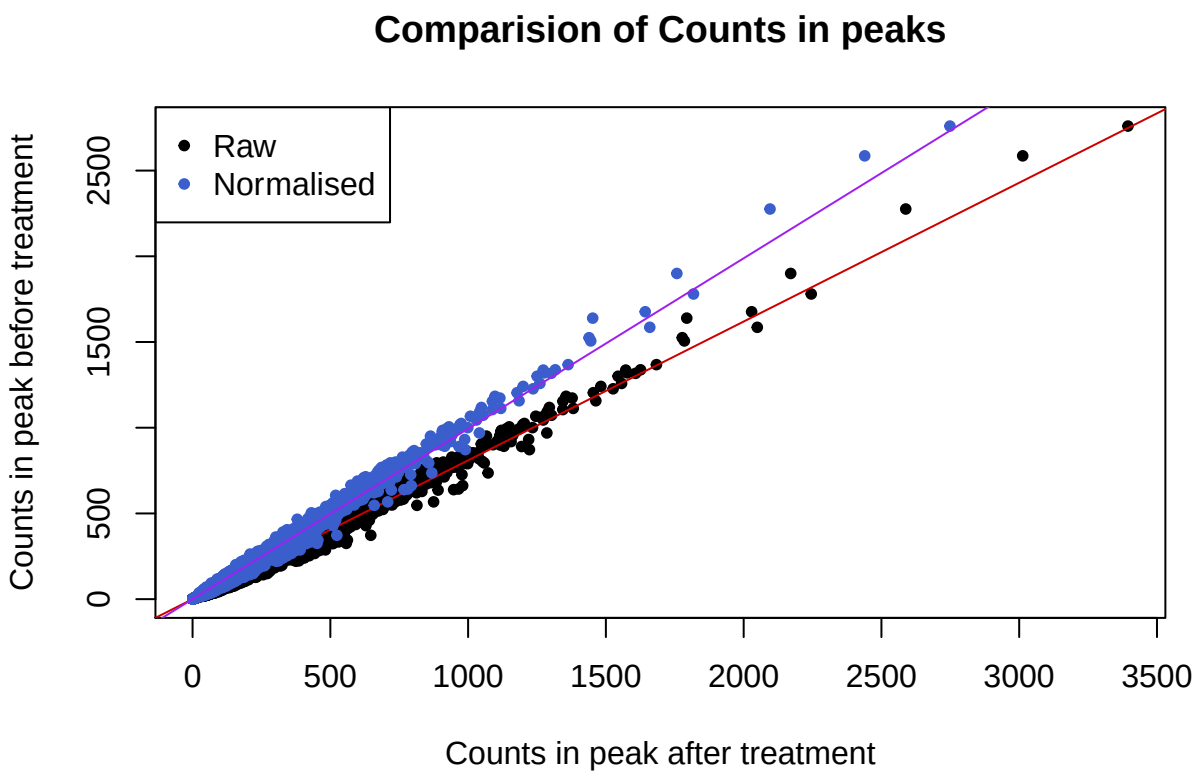
```
## Extract Peak set from DiffBind
jg.experimentPeakset <- jg.dbaGetPeakset(dbaExperiment)
jg.controlPeakset    <- jg.dbaGetPeakset(dbaControl)
```

```
#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)
```

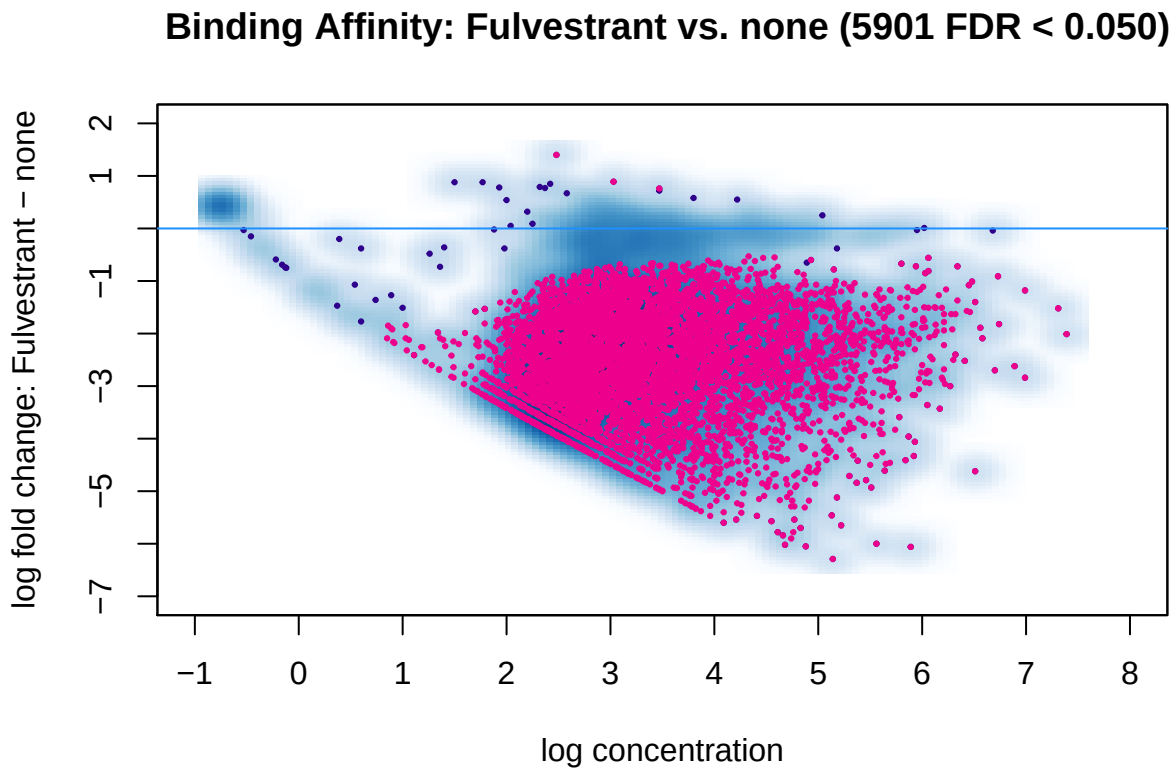
```
#Apply coefficent and control factor
jg.experimentPeaksetNormalised<-jg.applyNormalisation(jg.experimentPeakset,
                                                      jg.coefficient,
                                                      jg.correctionFactor,
                                      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