Differential Expression Analysis (Ballgown)

Ballgown is an R/Bioconductor package for flexible, isoform-level differential expression analysis of RNA-seq data [9]. Ballgown's data structures make it easy to use table-based packages like limma (Smyth (2005)), limma Voom (Law et al. (2014)), DESeq (Anders & Huber (2010)), DEXSeq (Anders et al. (2012)), or edgeR (Robinson et al. (2010)) for differential expression analysis.

Ballgown requires three pre-processing steps:

1. MeRIP-Seq reads should be aligned to a reference genome. (**HISAT2**)
2. A transcriptome should be assembled, or a reference transcriptome should be downloaded. (**StringTie**)
3. Expression for the features (transcript, exon, and intron junctions) in the transcriptome should be estimated in a Ballgown readable format. (**StringTie**)

Differential Expression Analysis

An example of the working directory:

```
stringtie_homo/
    SRR5978827/
        e2t.ctab
        e_data.ctab
        i2t.ctab
        i_data.ctab
        t_data.ctab
    SRR5978828/
        e2t.ctab
        e_data.ctab
        i2t.ctab
        i_data.ctab
        t_data.ctab
    ...
```

Create an R script `load_bg.R` for loading ballgown object:

```
library(ballgown)
bg <- ballgown(dataDir="/path/to/stringtie_homo", samplePattern='SRR',
meas='all')
save(bg, file='bg.rda')
```

Then run this script in terminal

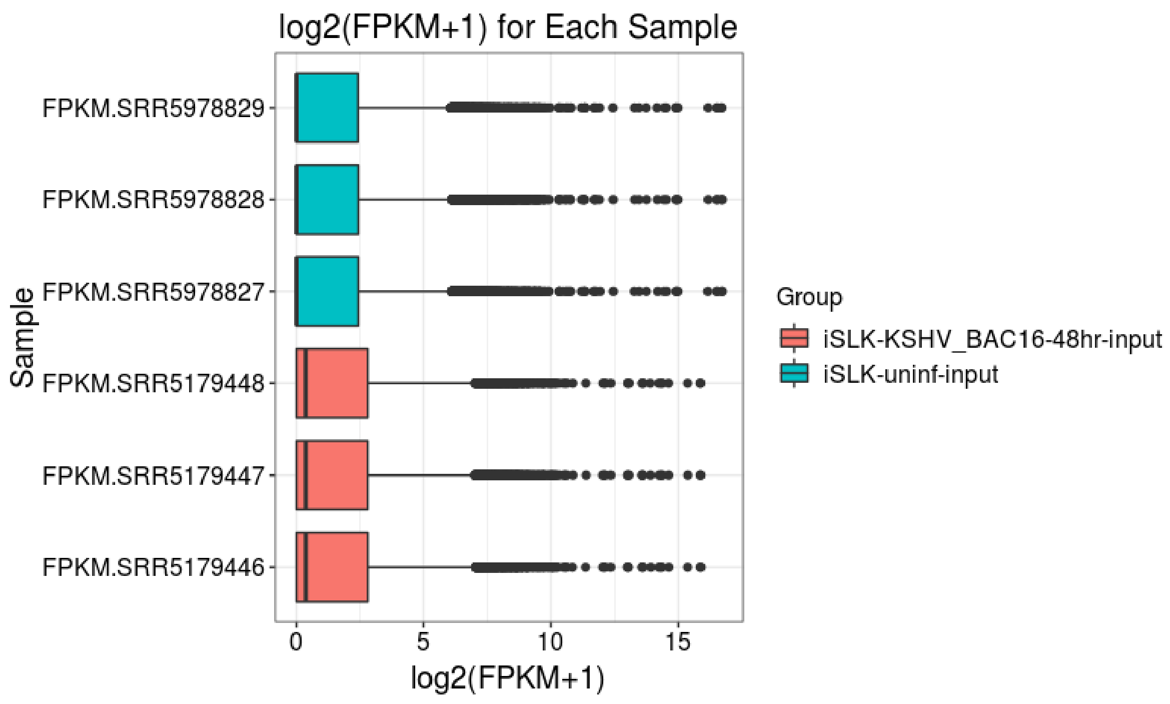
```
$ R CMD BATCH load_bg.R
```

Differential expression analysis with Ballgown:

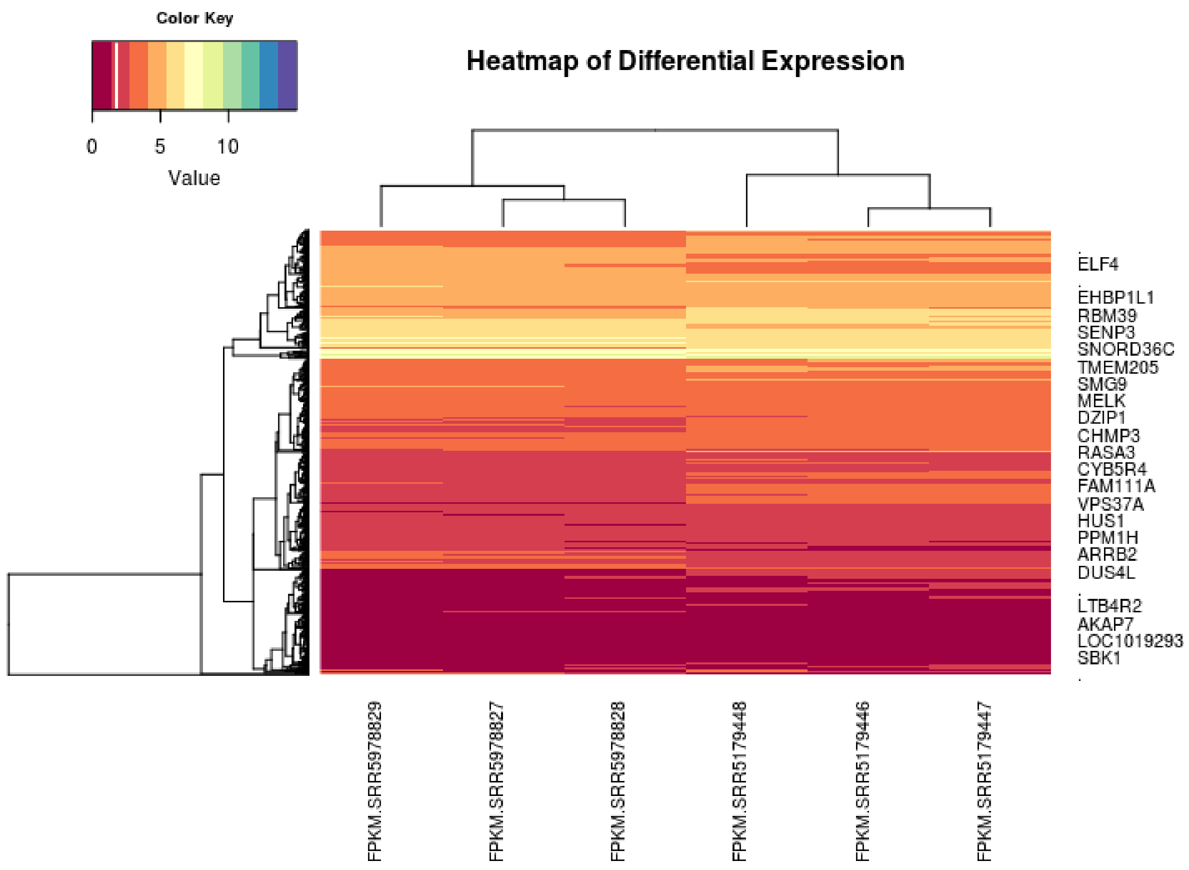
```
# Set directory
setwd("/path/to/ballgown/")
```

```r
# Load R packages
library(ballgown)
library(genefilter)

# Load data
bg = ballgown(dataDir="/path/to/stringtie_homo", samplePattern='SRR',
meas='all')
# Save data for backup
save(bg, file='bg.rda')

# Load all attributes and gene names
bg_table = texpr(bg, 'all')
bg_gene_names = unique(bg_table[, 9:10])

# Get gene expression data frame
gene_expression = as.data.frame(gexpr(bg))

# Add pData specifying groups
group = c(rep("iSLK-KSHV_BAC16-48hr-input", 3), rep("iSLK-uninf-input", 3))
pData(bg) = data.frame(id=sampleNames(bg), group=group)

# Perform differential expression (DE) analysis with no filtering
results_transcripts = stattest(bg, feature="transcript", covariate="group",
getFC=TRUE, meas="FPKM")
results_genes = stattest(bg, feature="gene", covariate="group", getFC=TRUE,
meas="FPKM")
results_genes = merge(results_genes, bg_gene_names, by.x=c("id"),
by.y=c("gene_id"))

# Filter low-abundance genes.
bg_filt = subset(bg,"rowVars(texpr(bg)) > 1", genomesubset=TRUE)

# Load all attributes including gene name
bg_filt_table = texpr(bg_filt , 'all')
bg_filt_gene_names = unique(bg_filt_table[, 9:10])

# Perform DE analysis now using the filtered data
results_transcripts = stattest(bg_filt, feature="transcript", covariate="group",
getFC=TRUE, meas="FPKM")
results_genes = stattest(bg_filt, feature="gene", covariate="group", getFC=TRUE,
meas="FPKM")
results_genes = merge(results_genes, bg_filt_gene_names, by.x=c("id"),
by.y=c("gene_id"))

# Identify the significant genes with p-value < 0.05
sig_transcripts = subset(results_transcripts, results_transcripts$pval < 0.05)
sig_genes = subset(results_genes, results_genes$pval < 0.05)

# Visualization1: view the range of values and general distribution of FPKM
values
FPKM_dist = function(gene_expression) {
  log_fpkm = log2(gene_expression+1)
  fpkm_long = gather(data.frame(log_fpkm), "Sample", "log2(FPKM+1)")
  index_table = pData(bg)
  index_table$id = paste("FPKM.", index_table$id, sep='')
  colnames(index_table) = c("Sample", "Group")
  fpkm_long = left_join(fpkm_long, index_table, by = "Sample")
  p = ggplot(fpkm_long, aes(x=Sample, y=`log2(FPKM+1)`, fill = Group)) +
    geom_boxplot() + coord_flip() +
    labs(title = "log2(FPKM+1) for Each Sample") +
    theme_bw() +
    theme(plot.title = element_text( size=18, vjust=0.5, hjust=0.5),
          axis.title.x = element_text(size=15, vjust=0.5),
          axis.title.y = element_text( size=15, vjust=0.5),
          axis.text.x = element_text(size=12, colour="black"),
          axis.text.y = element_text(size=12, colour="black"),
          legend.text = element_text(size=12, colour="black"),
          legend.title = element_text(size=12, colour="black"),
          plot.caption = element_text(size=12, colour="gray75", face="italic",
hjust = 1, vjust = 1))
  return(p)
}
p1 = FPKM_dist(gene_expression)

# Visualization2: heat map of differential expression
heatmap = function(gene_expression, results_genes){
  library(RColorBrewer)
  Colors=brewer.pal(11,"Spectral")
  results_genes[,"de"] = log2(results_genes[,"fc"])
  sigpi = which(results_genes[,"pval"]<0.05)
  topn = order(abs(results_genes[sigpi,"fc"]), decreasing=TRUE)[1:25]
  topn = order(results_genes[sigpi,"qval"])[1:25]
  sigp = results_genes[sigpi,]
  sigde = which(abs(sigp[,"de"]) >= 2)
  sig_tn_de = sigp[sigde,]
  mydist=function(c) {dist(c,method="euclidian")}
  myclust=function(c) {hclust(c,method="average")}
  main_title="Heatmap of Differential Expression"
  par(cex.main=0.8)
  sig_genes_de=sig_tn_de[,"id"]
  sig_gene_names_de=sig_tn_de[,"gene_name"]
  data=log2(as.matrix(gene_expression[as.vector(sig_genes_de),])+1)
  p = heatmap.2(data,
               hclustfun=myclust,
               distfun=mydist,
               na.rm = TRUE,
               scale="none",
               dendrogram="both",
               margins=c(7,5),
               Rowv=TRUE, Colv=TRUE,
               symbreaks=FALSE, key=TRUE, symkey=FALSE,
               density.info="none", trace="none",
               main=main_title,
               cexRow=0.8, cexCol=0.8,
               labRow=sig_gene_names_de,
               # col=rev(heat.colors(75)),
               col=Colors)
  return(p)
}
p2 = heatmap(gene_expression, results_genes)
```

A barplot shows the range and general distribution of FPKM values:



A heat map shows the significant differential expressed genes among all samples:



Note that `pData` should hold a data frame of phenotype information for the samples in the experiment, and be added during ballgown object construction. It can also be added later.

Also note that you can remove all transcripts with a low variance across the samples before differential analysis.

Results

DE results for dataset GSE93676:

```
# To see the first few lines of significant genes and transcripts:
head(sig_transcripts)
#       feature  id           fc        pval      qval
# 4  transcript  40 1.237362e-02 0.047948681 0.7342077
# 13 transcript  68 3.643648e-13 0.041800485 0.7342077
# 35 transcript 133 2.283918e+17 0.009168513 0.7342077
# 51 transcript 167 6.634653e-10 0.034473953 0.7342077
# 70 transcript 244 2.241706e-16 0.048610761 0.7342077
# 76 transcript 276 2.637682e-18 0.002536443 0.7342077

head(sig_genes)
#              id feature           fc       pval      qval gene_name
# 1     100128071    gene 1.445238e-59 0.04327419 0.7012609   FAM229A
# 16       119710    gene 1.378979e-51 0.01122029 0.6171722  C11orf74
# 57         6289    gene 1.718444e-71 0.01229523 0.6171722      SAA2
# 74  MSTRG.10000    gene 1.942633e+58 0.01965612 0.6250552     NGLY1
# 97  MSTRG.10028    gene 5.764237e+05 0.04138674 0.6962953   PDCD6IP
# 141 MSTRG.10096    gene 1.998718e-66 0.02531592 0.6422143     PTH1R
```

DE results for dataset GSE47217:

```
head(sig_transcripts)
#        feature   id        fc        pval      qval
# 57  transcript  479 0.9339480 0.036290263 0.7220780
# 66  transcript  522 0.4784699 0.002714966 0.5205885
# 86  transcript  685 0.8718279 0.008432731 0.5794108
# 107 transcript  847 0.8360476 0.031839607 0.7176861
# 117 transcript  908 1.0784123 0.004445226 0.5428939
# 137 transcript 1051 0.4439950 0.036100181 0.7220780

head(sig_genes)
#             id feature        fc       pval      qval gene_name
# 1   MSTRG.1000    gene 1.0657124 0.04960470 0.5469039  Ppp1r15b
# 36 MSTRG.10096    gene 0.9003980 0.03641091 0.5417405     Lpin2
# 55 MSTRG.10153    gene 0.9036117 0.02278910 0.5227679     Birc6
# 56 MSTRG.10153    gene 0.9036117 0.02278910 0.5227679         .
# 85  MSTRG.1024    gene 1.2618979 0.03277910 0.5417405  Tmem183a
# 94  MSTRG.1027    gene 0.9601584 0.04558797 0.5458907   Adipor1
```