Fancy Posters

RNA-seq Data Analysis With R-bioconductor: An Exercise in Reproducable Research

Pietà Schofield
Data Analysis Group
Computational Biology
College of Life Sciences
University of Dundee

Introduction *knitr*

This poster is a demonstration of reproducible research. The poster was created as a 300 line R-noweb script. The script contains all the words that appear on the poster and also all the code required to process the raw data and to produce the figures that appear on the poster. R-noweb is an augmented form of LaTeX. Similarly `Rmarkdown` scripts can be written to produce HTML and MS Word documents. They are compiled by the R `knitr` package

```
setwd(posterPath)
logknit <- knit("rnaseq_poster.rnw")
loglatex <- system("pdflatex rnaseq_poster.tex",intern=T)
system("open rnaseq_poster.pdf")
```

For completeness this is the code that is used to compile the poster *NB although this code is included in the script it is not excuted at compile time as this would create and infinitely recursive process*

Quality Checking *seqTools*

If you have Illumina data generally it is in the form of FASTQ files. The first step is to examine the quality of the data, this can be done using the `seqTools` package.
First set up the paths to the raw data and use the `fastqq` function to generate quality metrics for all the files.

```
homePath <- "/Users/pschofield/"
projPath <- paste0(homePath, "Projects/rna_seq/")
fastqPath <- paste0(homePath,"Projects/BS32010_2015/web/data")
reportPath <- paste0(projPath,"bam/")
files <- list.files(fastqPath, pattern="^(S|F).*fq[.]gz", full=TRUE)
names(files) <- sub(".fq.gz","",basename(files))
if(!exists("fq")) fq<-fastqq(files,k=6,probeLabel=names(files))
```

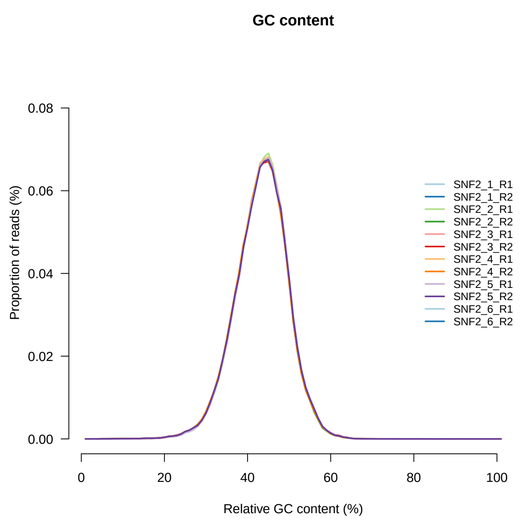
then visualise some of the quality metrics on all these files, here we show just a couple though it would be best to look at the metrics for each file. For example:

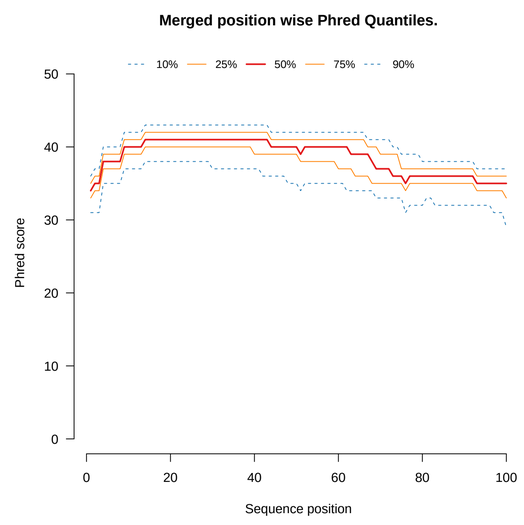
View the gc content distribution of all the reads

```
plotGCcontent(fq)
```

View the phred quality distribution for each base by position

```
plotMergedPhredQuant(fq)
```





Read Alignment *Rsubread*

The reads must be aligned to a reference genome. We can use the bioconductor package `Rsubread` to make a reference index from a FASTA file available from the ENSEMBL website and align the reads in the FASTQ files. The `subjunc` function is used to do the alignment as it copes with reads containing splice junctions.

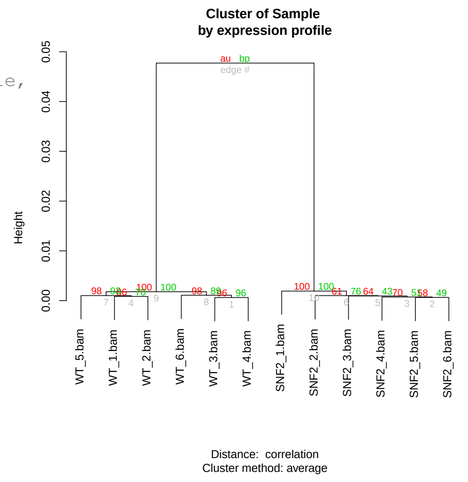
```
ref <- paste0(projPath,"annot/Scerevisiae_R64_1_1_dna.fa")
indexname <- paste0(projPath,"annot/sc_R64")
if(!file.exists(paste0(indexname,".log"))) buildindex(basename=indexname,reference=ref)
```

```
lapply(seq(1,length(files)),function(x){
  subjunc(indexname,files[x],nthread=4,input_format="gzFASTQ",
          output_file=paste0(gsub("[.]fq.gz","",files[x]),"_subjunc.bam"))
})
```

Then the aligned reads assigned to annotated features using the `featureCounts` function

```
bamFiles <- list.files(fastqPath,
                       pattern=".*bam$",full=T)
annoFile <- paste0(projPath,
                   "annot/Scerevisiae_R64_1_1.gtf")
if(!exists("fc")){
  fc <- invisible(featureCounts(bamFiles,annot.ext=annoFile,
                      isGTFAnnotationFile=TRUE,
                      isPairedEnd=FALSE))
}
fcounts <- fc$counts
colnames(fcounts) <- gsub("_subjunc[.]bam","",
                          basename(bamFiles))
fcounts <- fcounts[rowSums(fcounts)>0,]
pvcFC <- pvclust(fcounts)
```

```
## Bootstrap (r = 0.5)... Done.
## Bootstrap (r = 0.6)... Done.
## Bootstrap (r = 0.7)... Done.
## Bootstrap (r = 0.8)... Done.
## Bootstrap (r = 0.9)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.1)... Done.
## Bootstrap (r = 1.2)... Done.
## Bootstrap (r = 1.3)... Done.
## Bootstrap (r = 1.4)... Done.
```



The clustering indicates that generally the replicates cluster by treatment, however one of the snf2 replicates stands out. This might indicate a problem with the replicate, or an unexplain source of biological variation and warrents further investigation.

Differential Expression *edgeR*

In order to perform a differential expression analysis the data files must be distinguished by treatment, or tissue or whatever factor is relevant. This information can be extracted from the file names

```
treatment=relevel(as.factor(ifelse(grepl("^W",colnames(fcounts)),"WT","SNF2")),"WT")
```

There are many differential expression packages available in R-bioconductor, here we will use the `edgeR` package. The data must be transformed into a data structure used by `edgeR`.

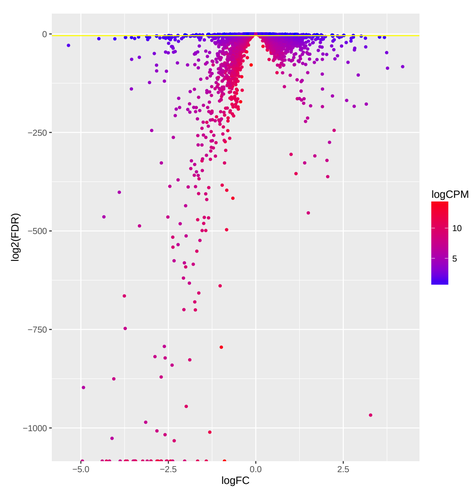
```
require(edgeR, quiet=TRUE)
dge <- DGEList(counts=fcounts[rowSums(fcounts)>0,],
               group=treatment)
```

The data must also be normalized to account for the difference in library sizes and the variance model.

```
dge <- calcNormFactors(dge)
```

The differential expression test can then be performed

```
require(ggplot2,quiet=TRUE)
mm <- model.matrix(~treatment)
dge <- estimateGLMCommonDisp(dge,mm)
dge <- estimateGLMTrendedDisp(dge,mm)
dge <- estimateGLMTagwiseDisp(dge,mm)
fit <- glmFit(dge,mm)
lrt <- glmLRT(fit,coef=2)
tt <- as.data.frame(topTags(lrt,n=10000))
gp <- ggplot(tt, aes(x=logFC,y=log2(FDR)))
gp <- gp + geom_point(size=1, aes(colour = logCPM))
gp <- gp + scale_colour_gradient(high = 2, low = 4 )
gp <- gp + geom_abline(intercept=log2(0.05),
```



We can use the `ggplot2` package to plot the false discovery rate adjusted significance against the Log2 fold-

Significant Genes *biomaRt*

The significantly differentially expressed genes can be extracted from the topTags listing first we will get the annotations for the significant genes using `biomaRt`

```
require(biomaRt)
if(!exists("ttAnno")){
  mart <- useMart("ensembl","scerevisiae_gene_ensembl")
  anno <- getBM(attributes=c("ensembl_gene_id","external_gene_name", "description","entrezgene"),
                filter="ensembl_gene_id",values=rownames(tt)[which(tt$FDR<=0.05)],mart=mart)
  colnames(anno) <- c("GeneId","GeneName","Description","EntrezId")
  ttAnno <- merge(anno,tt[,c("logFC","FDR")],by.x="GeneId",by.y="row.names")
  ttAnno$Description <- paste0("\\parbox{0.5\\textwidth}{",gsub("%"," percent ",ttAnno$Description),"}")
  ttAnno$FDR <- sapply(ttAnno$FDR,function(x) sprintf("%e",x))
}
```

```
downGenes <-  ttAnno$GeneId[which(ttAnno$logFC<=0)]
upGenes <- ttAnno$GeneId[which(ttAnno$logFC>=0)]
```

Expression	Count
Up regulated	1584
Down regulated	1510

Gratifyingly the most significantly differentially expressed gene with also the highest foldchange is YOR290C (SNF2) which is strongly down regulated in the mutant. However the top five most strongly up regulated annotations are all "Dubious ORF"s.

```
upTop <- ttAnno[which(ttAnno$GeneId %in% upGenes),]
kable(head(upTop[rev(order(upTop$logFC)),],5),row.names=FALSE)
```

GeneId	GeneName	Description
YOR376W		\parbox{0.5\textwidth}{Dubious open reading frame; unlikely to encode a functional protein, based on available experime
YPL025C		\parbox{0.5\textwidth}{Dubious open reading frame; unlikely to encode a functional protein, based on available experime
YHR095W		\parbox{0.5\textwidth}{Dubious open reading frame; unlikely to encode a functional protein, based on available experime
YHR125W		\parbox{0.5\textwidth}{Dubious open reading frame; unlikely to encode a functional protein, based on available experime
YPL021W	ECM23	\parbox{0.5\textwidth}{Non-essential protein of unconfirmed function; affects pre-rRNA processing, may act as a negative

similarly for down regulated genes

GeneId	GeneName	Description
tN(GUU)P		\parbox{0.5\textwidth}{Asparagine tRNA (tRNA-Asn), predicted by tRNAscan-SE analysis [Source:SGD;Acc:S0000066
YNR034W-A		\parbox{0.5\textwidth}{Putative protein of unknown function; expression is regulated by Msn2p/Msn4p; YNR034W-A ha
YHR136C	SPL2	\parbox{0.5\textwidth}{Protein with similarity to cyclin-dependent kinase inhibitors; downregulates low-affinity phosphat
YDR545W	YRF1-1	\parbox{0.5\textwidth}{Helicase encoded by the Y' element of subtelomeric regions; highly expressed in the mutants la
YDR545W	YRF1-1	\parbox{0.5\textwidth}{Helicase encoded by the Y' element of subtelomeric regions; highly expressed in the mutants la

Network Analysis *STRINGdb*

There are now many packages in R-bioconductor that can be used to investigate gene set and pathway enrichment of the significant genes. Here we show the example of `STRINGdb`

```
string_db <- invisible(STRINGdb$new(
                    version="9_05",
                    species=4932,
                    score_threshold=0,
                    input_directory=""))
```

```
## WARNING: A new version of the STRING R plugin has been released! We suggest you to install the latest vers
```

```
downMap <- string_db$map(downTop,"GeneId",
                removeUnmappedRows=T)
```

```
## Warning:  we couldn't map to STRING 2% of your identifiers
```

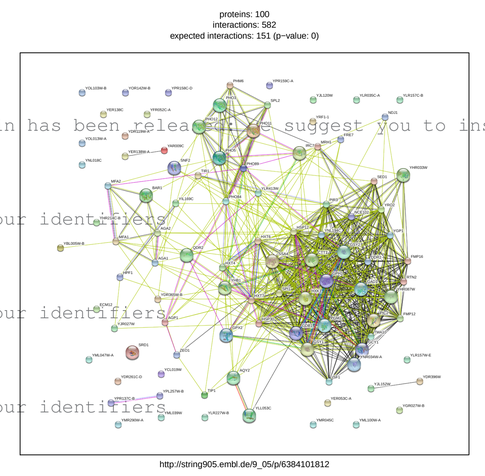
```
upMap <- string_db$map(upTop,"GeneId",
                removeUnmappedRows=T)
```

```
## Warning:  we couldn't map to STRING 0% of your identifiers
```

```
allMap <- string_db$map(ttAnno,"GeneId",
                removeUnmappedRows=T)
```

```
## Warning:  we couldn't map to STRING 1% of your identifiers
```

```
hits <- head(downMap$STRING_id[
            order(downMap$logFC)],100)
```




```
string_db$plot_network(hits)
```

The `STRINGdb` can also be used to look for enriched GO terms and enriched KEGG pathways

```
backgroundV <- allMap$STRING_id[1:2000]
string_db$set_background(backgroundV)
hitsUP <- upMap$STRING_id
hitsDown <- downMap$STRING_id
upGOBP <- string_db$get_enrichment(
            hitsUP,
            category = "Process",
            methodMT = "fdr",
            iea = TRUE )
upKEGG <- string_db$get_enrichment(
            hitsUP,
            category = "KEGG",
            methodMT = "fdr",
            iea = TRUE )
```

```
colnames(upGOBP) <- gsub("_"," ",colnames(upGOBP))
kable(head(upGOBP,4))
```

term id	proteins	hits	pvalue	pvalue fdr	term description
GO:0008150	1243	683	0	0	biological_process
GO:0009987	1181	632	0	0	cellular process
GO:0044763	831	464	0	0	single-organism cellular process
GO:0044699	796	452	0	0	single-organism process

```
colnames(upKEGG) <- gsub("_"," ",colnames(upKEGG))
kable(head(upKEGG,4))
```

term id	proteins	hits	pvalue	pvalue fdr	term description
sce04111	56	43	0	0	Cell cycle - yeast
sce04113	51	38	0	0	Meiosis - yeast
sce03030	12	12	0	0	DNA replication
sce04120	18	14	0	0	Ubiquitin mediated proteolysis

Pathway Analysis *pathview*

Then using the `pathview` package it is possible to visualise the second top hit in the enriched pathway, sce:04111

```
gene.data <- as.numeric(ttAnno$logFC)
names(gene.data) <- ttAnno$GeneId
pathID <- gsub("sce","",head(upKEGG[1,1]))
pathview(gene.data=gene.data,
         gene.idtype="kegg",
         pathway.id=pathID,
         species="sce",
         out.suffix="kegg",
         kegg.native=T,
         same.layer=T)
```