## Set up libraries and functions

~~~
```{r session setup, include=FALSE}
library(rtracklayer)
library(tidyr)
library(sleuth)
library(tximport)
library(dplyr)
library(ggplot2)
library(biomaRt)
library(reactome.db)
library(goseq)
library(GO.db)
library(org.Ce.eg.db)
library(topGO)
library(Rgraphviz)
library(EnhancedVolcano)
library(pheatmap)
library(clusterProfiler)
library(svglite)
library(ggplot2)
library(enrichplot)
library(ggupset)
library(cowplot)
library(pathview)
library(gage)
options(stringsAsFactors = FALSE)
```
~~~

# Analysis of RNA-seq data from a neurexin-deficient worm

## Introduction

The RNA-seq data in this project originates from sequencing done at the Genomics Centre of QMUL. The samples came from Prof. Filipe Cabreiro's lab (then UCL, now Imperial) from normal and neurexin-deficient worms - the work was mostly carried out by Peter Cooke (then Phd student). Other members in Filipe's lab had helped out at the time with supervision and RNA extraction. The sequencing centre did not achieve good ribosomal depletion as it transpired that they did not have the right kit for *C.elegans*. Hence, only medium/high expressed genes are seen in these data and ribosomal RNA reads must be removed before analysing the data further.

Aine did a differential gene expression analysis based on counting reads using HTseq count and doing differential expresion with DESEq2 and limma. However, if I remember well, she missed out ~ half the reads due to a problem in one of her shell scripts (when pooling together the lanes, I think she pulled only two out of the four available).

It is now recognised that gene expression analysis should be based on aggregating the results of transcript-based analysis. Below is a list of links that describe how this should be done:

- Differential analysis of count data (by Pachter)
https://www.youtube.com/watch?v=ucPBBTjH5EE
The pipeline here focuses on the kallisto/sleuth pipeline.
- A good argument for using alignment-independent methods for transcript quantification:
https://cgatoxford.wordpress.com/2016/08/17/why-you-should-stop-using-featurecounts-htseq-or-cufflinks2-and-start-using-kallisto-salmon-or-sailfish/
(in there also reference to paper that uses transcript quantification and then summarises to gene level and finally also recommendations from Deseq2 developers on getting counts from kallisto/salmon and importing them to DESeq2)
- Kallisto + sleuth documentation: https://pachterlab.github.io/kallisto/ and https://pachterlab.github.io/sleuth/
- To assist with enrichment calculations / pathway analysis, these might be useful (was planning to explore them but ran out of time):
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6607905/ and
http://bioconductor.org/packages/release/BiocViews.html#**___GeneSetEnrichment**
- Two more potentially interesting papers:
https://link.springer.com/article/10.1007/s12035-011-8213-1 and
https://www.genetics.org/content/213/4/1415.abstract
### The data
The data comprise 64 FASTQ files in total. There are 8 samples, each split into 4 lanes and each being sequenced in paired-end mode, hence there are: 8 x 4 x 2= 64 files.All data is from reversely-stranded libraries i.e. read 2 is in the same direction as the transcript on the genome.
- 4 normal samples, names starting with "N" (biological replicates)
- 4 neurexin-deficient samples, names starting with "SG" (biological replicates).
All samples originated from 1-day old adult worms.

Below is an outline of the steps to analyse the RNA-seq data from Aine's
project, starting with raw fastq data and ending with mapped reads to the C.
elegans transcriptome using *kallisto*.
All code was run on the departmental server thoth. Where the variable
$PROJECT is used below it refers to the following directory:
/d/in16/u/ubcg71a/research/filipe/ .
## Data quality analysis with FastQC/multiQC
Initial fastqc data quality analysis on raw data was carried out by both Aine
and repeated by Krzysztof at some point. Results can be found here:
/d/in13/u/kszkop01/worm_neurexin/fastqc/
Had I run it myself I would have used my bash script *run_fastqc.sh* for
running fastQC on multiple files:
```{bash, eval = FALSE}
#!/bin/bash
# Runs FASTQC on all fastq.gz files found in given directory
# Run as:
# ./run_fastqc dir_of_fastq_files
timestamp=`date "+%Y%m%d-%H%M%S"`
logfile="run_$timestamp.log"
exec > $logfile 2>&1  #all output will be logged to logfile
FASTQC_DIR="/s/software/fastqc/v0.11.8/FastQC/fastqc"
echo "Running FASTQC using executable: $FASTQC_DIR"
for fastq_file in $1/*.fastq.gz;
do
  echo "Fastq file: $fastq_file"
  $FASTQC_DIR $fastq_file -o .
done
```
I summarised K's FASTQC results using multiQC:
```{bash, eval = FALSE}
module use -s /s/mm/modules
module load python/v2
multiqc /d/in13/u/kszkop01/worm_neurexin/fastqc/
```
Results are under: `$PROJECT/multiqc_on_raw/`
All samples have sequences of a single length (76 bp).
Diagnostic plots from FASTQC summarised with multiQC (in all these plots,
files with R1 are coloured red, files with R2 are blue):
![Alt text](multiqc/fastqc_per_base_sequence_quality_plot.png)
![Alt text](multiqc/fastqc_sequence_duplication_levels_plot.png)
![Alt text](multiqc/fastqc_per_base_n_content_plot.png)
![Alt text](multiqc/fastqc_per_sequence_gc_content_plot.png)
![Alt text](multiqc/fastqc_per_sequence_quality_scores_plot.png)
![Alt text](multiqc/fastqc_overrepresented_sequencesi_plot.png)
#### Observations from initial quality analysis

~~~
- Quality scores are generally very high even for the last few positions of
the read; in general R1 (shown in red in the plots above) slightly better
than R2 (shown in blue)
- There are high levels of duplication (most likely rRNA)
- There are some Ns mostly at the beginning of the reads
- GC content shows strange distribution with differences between R1 and R2
(I've seen this before in datasets; maybe bias from the bias due to adapters
still present?). GC of C. elegans is supposed to have mean base composition
of 36% GC (with the rRNA cistrons being closer to 51%; ref: PMID: 4858229).
Our plots show higher GC content in all cases but there is a peak where rRNA
should be.
- There is evidence of adapters present in the 3' end
I saved the statistics table from multiqc and read it in here to summarise:
```{r mutliqc_initial_stats}
t.stats <- read.table(file="multiqc/stats.txt", header=TRUE, sep="\t")
#use tidyr's separate to split the first column
t.stats <- separate(t.stats, Sample.Name, into=c("sample", "sample_num",
"lane", "read", "nothing"), sep="_", remove=FALSE, convert=FALSE)
summary(t.stats$M.Seqs)
#print statistics per group
t.stats %>% group_by(sample) %>% summarize(mean=mean(M.Seqs))
t.stats %>% group_by(lane) %>% summarize(mean=mean(M.Seqs))
t.stats %>% group_by(sample) %>% summarize(mean=mean(X..Dups))
t.stats %>% group_by(lane) %>% summarize(mean=mean(X..Dups))
t.stats %>% group_by(sample) %>% summarize(mean=mean(X..GC))
t.stats %>% group_by(lane) %>% summarize(mean=mean(X..GC))
```
## Read pre-processing with fastp
I trimmed (adapters and poly(A) tails) and quality-filtered the reads using
the program *fastp* :
Raw fastq data is under:
**$PROJECT/data/Data/170519_NS500784_0235_AHT2JYBGX2/Demulti/Data/Intensities/BaseCalls/GC-IN-6725/**
To remove poly(A) tails and adapters, I ran the script under
**$PROJECT/fastp**:
```{bash, eval = FALSE}
nohup ./run_fastp.sh
../data/Data/170519_NS500784_0235_AHT2JYBGX2/Demulti/Data/Intensities/BaseCalls/GC-IN-6725/ >& run_fastp.out &
```
Below is the run_fastp.sh script:
```{bash, eval = FALSE}
#!/bin/bash
# Runs fastp in PE mode for all .fastq.gz files in a directory
# Run as:
# ./run_fastp.sh directory_of_samples
~~~

~~~~
timestamp=`date "+%Y%m%d-%H%M%S"`
logfile="run_$timestamp.log"
exec > $logfile 2>&1  #all output will be logged to logfile
FASTP_EXEC="/d/in7/s/fastp/fastp"
DIR=$1
ADAPTER_FILE="./adapters_all.fa"
echo "Running fastp using executable: $FASTP_EXEC"
for file in `ls $DIR/*_R1_*.fastq.gz`;
do
  sample=${file/$DIR\//}
  sample=${sample/_R1_001.fastq.gz/}
  echo "Sample= $sample"
  $FASTP_EXEC -i  "$DIR/$sample"_R1_001.fastq.gz  \
              -I  "$DIR/$sample"_R2_001.fastq.gz \
         -o "$sample"_R1_001_trimmed.fastq.gz \
         -O "$sample"_R2_001_trimmed.fastq.gz \
         --adapter_fasta $ADAPTER_FILE \
         -l 30 --trim_poly_x \
         -h "$sample"_fastp.html -j "$sample"_fastp.json
done
```
NOTE: fastp by default uses the following quality filtering options:
- 40% of bases are allowed to be at phred quality Q<15 (unqualified)
- reads with at least 5 Ns are discarded
- average quality of the read is not checked by default
## Data quality re-analysis with FastQC/multiQC following fastp
Re-ran fastQC with the files in the fastp directory:
`./run_fastp.sh ../fastp`
Results in the directory: `$PROJECT/fastqc_after_fastp`
Then re-ran multiqc and moved the .html file locally to save the plots shown
below:
![Alt text](multiqc_after_fastp/fastqc_per_base_sequence_quality_plot.png)
![Alt text](multiqc_after_fastp/fastqc_sequence_duplication_levels_plot.png)
![Alt text](multiqc_after_fastp/fastqc_per_base_n_content_plot.png)
![Alt text](multiqc_after_fastp/fastqc_per_sequence_gc_content_plot.png)
![Alt text](multiqc_after_fastp/fastqc_per_sequence_quality_scores_plot.png)
![Alt text](multiqc_after_fastp/fastqc_overrepresented_sequencesi_plot.png)
Results are under `$PROJECT/fastqc_after_fastp/multiqc`
#### Observations from  re-analysis of quality
- There are still Ns mostly at the beginning of the reads but these are only
in small percentage of reads and only in the first 1-3 bases so will leave
them in.
- GC content shows the same strange distribution hence this has nothing to do
with adapters and either to do with biases in the library creation or, more
likely, the rRNA that is still present.
- Adapters have been successfully removed.
~~~~

~~~
## Merging data from the 4 lanes to have one file per sample
Seeing that there were no strong lane effects, I merged all 4 lanes for each
sample (this could have been done following mapping but there really was no
evidence here to suggest a lane effect). The new files are in the
**$PROJECT/fastp_merged** directory and the script run_fastq_merger.sh is
given below:
```{bash, eval = FALSE}
#!/bin/bash
# Merges all fastq files for the same sample and same read orientation into
single fastqc
# Run as:
# ./run_fastq_merger.sh directory_of_fastq_files sample_name sample_number
timestamp=`date "+%Y%m%d-%H%M%S"`
logfile="run_$timestamp.log"
exec > $logfile 2>&1  #all output will be logged to logfile
MERGED_DIR="/d/in16/u/ubcg71a/research/filipe/fastp_merged/"
cd $1
sample=$2
samplenum=$3
echo $(pwd)
echo Merging for sample $sample
cat "$sample"*"$samplenum"*R1_001*fastq.gz >
$MERGED_DIR/"$sample"_"$samplenum"_R1.fastq.gz
cat "$sample"*"$samplenum"*R2_001*fastq.gz >
$MERGED_DIR/"$sample"_"$samplenum"_R2.fastq.gz
```
```{bash, eval = FALSE}
./run_fastq_merger.sh ../fastp/ N1 S5
./run_fastq_merger.sh ../fastp/ N2 S6
./run_fastq_merger.sh ../fastp/ N3 S7
./run_fastq_merger.sh ../fastp/ N4 S8
./run_fastq_merger.sh ../fastp/ SG1 S1
./run_fastq_merger.sh ../fastp/ SG2 S2
./run_fastq_merger.sh ../fastp/ SG3 S3
./run_fastq_merger.sh ../fastp/ SG4 S4
```
## Mapping processed reads to the genome with STAR
Before carrying out the analysis with kallisto/sleuth, we will map the reads
using **STAR** and carry out some exploratory analysis of the data.
Aine's work showed a huge number of reads mapping to rRNA.Here, we will allow
them to map and exclude them later. During the kallisto run, we will remove
them from the transcriptome before mapping to save hassle.
First, we need to create an index of the genome with STAR:
```{bash, eval=FALSE}
module load star/v2.7
~~~

~~~
STAR --runThreadN 8 --runMode genomeGenerate --genomeDir . --genomeFastaFiles
../Caenorhabditis_elegans.WBcel235.dna.toplevel.fa --sjdbGTFfile
../Caenorhabditis_elegans.WBcel235.99.gtf --genomeSAindexNbases 12
#(note that parameter genomeSAindexNbases must be scaled down to 12 from the
default 14 because the genome
#is smaller than the human genome)
#The genome index is currently under:
/d/in16/u/ubcg71a/research/genomes/c.elegans/ensembl_99/STAR_INDEX/
#Once the genome index is created we can map the files using a bash script:
nohup ./run_star.sh ../fastp/ list_files > & run_star.out &
#where list_files contains a list of all samples
#N1_S5
#N2_S6
#N4_S8
#N3_S7
#SG1_S1
#SG2_S2
#SG3_S3
#SG4_S4
```
The script run_star.sh is given below:
```{bash, eval=FALSE}
#!/bin/bash
# Runs STAR mapping on all samples in a given directory
# Samples must end in .fastq.gz and are given as a list in a file (second
argument)
# Run as:
# run_star.sh directory_of_fastq_files list_of_samples
timestamp=`date "+%Y%m%d-%H%M%S"`
logfile="run_$timestamp.log"
exec > $logfile 2>&1  #all output will be logged to logfile
dir=$1
shift
#set location of executables
STAR_EXEC="/s/software/STAR/bin/Linux_x86_64/STAR"
numProc=5   #number of processors/threads to be used
genomeIndex="/d/in16/u/ubcg71a/research/genomes/c.elegans/ensembl_99/STAR_IND
EX/" #directory for genome index files
for sample in `cat $1`;
do
   echo "Running STAR on sample $sample (paired-end reads) ..."
   inputFile1="$dir$sample"_R1.fastq.gz
   inputFile2="$dir$sample"_R2.fastq.gz
   STAR --runThreadN $numProc \
        --runMode alignReads \
        --outSAMtype BAM SortedByCoordinate \
~~~

~~~
        --readFilesCommand zcat \
        --genomeDir $genomeIndex \
        --outFileNamePrefix "$sample"_ \
        --readFilesIn $inputFile1 $inputFile2 \
        --outReadsUnmapped Fastx \
        --quantMode GeneCounts
done
echo "All done!"
```
STAR produces separate files with the unmapped reads (which we can use, for
example, to map to circular RNA transcripts later) and it also produces gene
counts. Ideally one should rely on transcript counts for differential gene
expression; here we will use gene counts for the exploratory part of the
analysis only.
I summarised the STAR output by running multiqc again. Results are under
`$PROJECT/star_mapping/multiqc/`
![Alt text](multiqc_after_star/alignment_result.png)
![Alt text](multiqc_after_star/star_alignment_plot.png)
![Alt text](multiqc_after_star/star_gene_counts.png)
## Exploratory data analysis with STAR gene counts
I start by defining my own version of DESeq2's plotPCA() function so that I
can define which PCs I will be plotting
```{r myPlotPCA definition}
#define my own pcaplot to allow other components besides PC1 and PC2 to be
plotted
#the function itself is copied from DESeq2
myPlotPCA <- function (x, intgroup = "condition", pc1 = 1, pc2 =2, ntop =
500, returnData = FALSE)
{
  rv <- genefilter::rowVars(assay(x)) #if using with SummarizedExperiment
objects
  select <- order(rv, decreasing = TRUE)[seq_len(min(ntop,
                                                     length(rv)))]
  pca <- prcomp(t(assay(x)[select, ]))
  percentVar <- pca$sdev^2/sum(pca$sdev^2)
  if (!all(intgroup %in% names(colData(x)))) {
     stop("the argument 'intgroup' should specify columns of colData(dds)")
  }
  intgroup.df <- as.data.frame(colData(x)[, intgroup, drop = FALSE])
  group <- if (length(intgroup) > 1) {
            factor(apply(intgroup.df, 1, paste, collapse = " : "))
        }
        else {
            colData(x)[[intgroup]]
        }
~~~

~~~
  d <- data.frame(PC1 = pca$x[, pc1], PC2 = pca$x[, pc2], group = group,
                  intgroup.df, names = colnames(x))
  if (returnData) {
    attr(d, "percentVar") <- percentVar[c(pc1,pc2)]
    return(d)
  }
  ggplot(data = d,
         aes_string(x = "PC1", y = "PC2",  color = "group"))+
    geom_point(size=3) +
    xlab(paste0("PC",pc1,": ", round(percentVar[pc1] * 100), "% variance")) +
    ylab(paste0("PC",pc2,": ", round(percentVar[pc2] * 100), "% variance"))
}
```
## Using rtracklayer and the gtf file to obtain gene names
```{r gene_info_from_gtf}
gtf <- rtracklayer::import('genomes/Caenorhabditis_elegans.WBcel235.99.gtf')
gtf_df=as.data.frame(gtf)
t2g <- data.frame(target_id = gtf_df$transcript_id,
                  ens_gene = gtf_df$gene_id,
                  ext_gene = gtf_df$gene_name)
t2g <- t2g %>% drop_na()
t2g <- dplyr::distinct(t2g)
```
The gene counts are in the .tab files of STAR. I moved those locally and use
code from:
http://biostars.org/p/241602
to read in counts so they can be used with DESeq2.
```{r EDA with STAR gene counts and DESeq2, eval=TRUE}
library(ggplot2)
library(DESeq2)
#First, read in the gene counts from STAR
number<- 4 #reverse-stranded library so counts should be in column 4
ff <- list.files( path = "./star_mapping", pattern =
"*ReadsPerGene.out.tab$", full.names = TRUE )
counts.files <- lapply( ff, read.table, skip = 4 )
starcounts <- as.data.frame( sapply( counts.files, function(x) x[ , number ]
) )
ff <- gsub( "[_]ReadsPerGene[.]out[.]tab", "", ff )
ff <- gsub( "[.]/star_mapping/", "", ff )
colnames(starcounts) <- ff
row.names(starcounts) <- counts.files[[1]]$V1
rm(counts.files)
coldata <- read.table("metadata.txt", header=T)
rownames(coldata) <- coldata[,1]
coldata
#Check row and column names match in the same order
~~~

~~~~
rownames(coldata)  == colnames(starcounts)
dds = DESeqDataSetFromMatrix(countData = starcounts,
                             colData = coldata,
                             design = ~ condition )
dds
#pre-filter to remove genes with too few counts
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
dds
#check quickly the distribution of the other counts
quantile(rowSums(counts(dds)))
#and in more detail
quantile(rowSums(counts(dds)), probs=seq(0.9,1,0.005))
vsd <- vst(dds, blind=TRUE)
#note that by default only the top 500 genes in terms of variance are used
for this PCA plot
plotPCA(vsd, intgroup=c("condition"))
```
This initial PCA looks promising with good separation of the samples.
However, the rRNAs remain in the counts as shown above and they could be
skewing the results.
We need to remove all counts corresponding to rRNA transcript_biotype reads
(there are 23) - the genome/transcriptome files are from Ensembl and can be
found in this directory: **/d/in16/u/ubcg71a/research/genomes/c.elegans**
```{bash, eval = FALSE}
 zcat Caenorhabditis_elegans.WBcel235.cdna_and_ncrna.fa.gz | grep
biotype:rRNA > list_of_biotype_rRNA_transcripts.txt

#keep gene names only
awk '{print $4}' list_of_biotype_rRNA_transcripts.txt | sed 's/gene://' | sed
's/\..//' > list_biotype_rRNA_genes.txt
```
We will now move this list of rRNA genes locally and use it to filter out the
unwanted counts.
```{r EDA with STAR gene counts and DESEq2 no rRNAs, eval=TRUE}
library("pheatmap")
#clean up
rm(ff, coldata, starcounts, dds, vsd)
number<- 4 #reverse-stranded library so counts should be in column 4
ff <- list.files( path = "./star_mapping", pattern =
"*ReadsPerGene.out.tab$", full.names = TRUE )
counts.files <- lapply( ff, read.table, skip = 4 )
starcounts <- as.data.frame( sapply( counts.files, function(x) x[ , number ]
) )
ff <- gsub( "[_]ReadsPerGene[.]out[.]tab", "", ff )
ff <- gsub( "[.]/star_mapping/", "", ff )
~~~~

```
colnames(starcounts) <- ff
row.names(starcounts) <- counts.files[[1]]$V1
rm(counts.files)
coldata <- read.table("metadata.txt", header=T)
rownames(coldata) <- coldata[,1]
coldata
#Check row and column names match in the same order
rownames(coldata)  == colnames(starcounts)
#now remove the rRNAs from the counts table
rRNAgenes <- read.table(file="list_biotype_rRNA_genes.txt", header=F)
rRNAgenes
starcounts.norRNA <- starcounts[!row.names(starcounts)%in%rRNAgenes[,1],]
dim(starcounts)
dim(starcounts.norRNA)
dim(rRNAgenes)
rm(starcounts) #clean up
dds = DESeqDataSetFromMatrix(countData = starcounts.norRNA,
                             colData = coldata,
                             design = ~ condition )
dds
#pre-filter to remove genes with too few counts
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
dds
#check quickly the distribution of the other counts
quantile(rowSums(counts(dds)))
#and in more detail
quantile(rowSums(counts(dds)), probs=seq(0.9,1,0.005))
#show the genes that still have very large counts
i<- rowSums(counts(dds)) >= 100000
assay(dds)[i,]
#Check correlation between raw gene expression vectors for all samples
(without log transformation)
pairs(assay(dds))
#and following log transformation
pairs(log(assay(dds)+0.5)) #add 0.5 as pseudocount
cor.mat <- cor(assay(dds))
cor.mat
pheatmap(cor.mat, legend=T)
#This all looks good - Ns correlate better with each other than with SGs and
vice versa.
#Transform the counts
vsd <- vst(dds, blind=TRUE)
plotPCA(vsd, intgroup=c("condition"))
#As expected the PCA is not changed much because we only removed a few genes
#try rlog
```

~~~~
rld <- rlog(dds, blind = TRUE)
plotPCA(rld, intgroup=c("condition"))
#we can improve the plot a bit by highlighting the replicate number in
different shape
pcaData <- plotPCA(vsd, intgroup=c("condition", "sample"), returnData=TRUE)
pcaData$replicate <- sub(pattern= ".*(.)_.*",
                         replacement="\\1",
                         perl=TRUE, fixed=FALSE,
                         x=pcaData$sample)
pcaData
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=condition, shape=replicate)) +
    geom_point(size=3) +
    xlab(paste0("PC1: ",percentVar[1],"% variance")) +
    ylab(paste0("PC2: ",percentVar[2],"% variance")) +
    coord_fixed() +
    ggtitle(label="STAR counts, transformed with vsd")
pcaData <- myPlotPCA(vsd, pc1= 2, pc2= 3, intgroup=c("condition", "sample"),
returnData=TRUE)
pcaData$replicate <- sub(pattern= ".*(.)_.*",
                         replacement="\\1",
                         perl=TRUE, fixed=FALSE,
                         x=pcaData$sample)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=condition, shape=replicate)) +
    geom_point(size=3) +
    xlab(paste0("PC2: ",percentVar[1],"% variance")) +
    ylab(paste0("PC3: ",percentVar[2],"% variance")) +
    coord_fixed()
#Check correlation between transformed gene expression vectors for all
samples
pairs(assay(vsd))
cor.mat <- cor(assay(vsd))
cor.mat
pheatmap(cor.mat, legend=T)
```
Next, we carry out differential gene expression based on the gene counts from
STAR
mapping. Note that this approach is now pretty much deprecated and replaced
by transcript-based analysis summarised at the gene level so we won't really
analyse the results here but add them here for completion.
```{r Old-style differential gene expression with DESeq2}
#BiocManager::install("apeglm")
library(apeglm) #to use in the lfcShrink function
dds <- DESeq(dds) #estimates size factors and dispersions and fits the model
res <- results(dds)
~~~~

~~~
res
#next we want to shrink the LFC to help visualisation and ranking of genes
resultsNames(dds)
resLFC <- lfcShrink(dds, coef="condition_neurexin_vs_control", type="apeglm")
resLFC
#MA plot
plotMA(resLFC, ylim=c(-2,2))
#contrast with the same plot but for the unshrunken log fold changes
plotMA(res, ylim=c(-2,2))
#plot top few genes
resLFCOrdered <- resLFC[order(resLFC$pvalue),]
resLFCOrdered[1:10,]
rm(resLFC)
for (i in 1:10) {
 plotCounts(dds, gene=rownames(resLFCOrdered)[i], intgroup="condition")
}
```
## Further exploration of the data using the pcaExplorer package in R (just
trying out - DO NOT INCLUDE)
```{r EDA with pcaExplorer, eval =FALSE}
#devtools::install_github("rstudio/d3heatmap") #doesn't install otherwise
#BiocManager::install("pcaExplorer")
library(pcaExplorer)
#need a mapping from Ensembl ids to gene names; will use a t2g subset; row
names must map gene ids
t2g.sub <- t2g[,c("ens_gene", "ext_gene")]
t2g.sub<- dplyr::distinct(t2g.sub)
rownames(t2g.sub) <- t2g.sub[,"ens_gene"]
colnames(t2g.sub) <- c("ens_gene", "gene_name")
#pcaExplorer(dds = dds, dst = vsd, annotation= t2g.sub)
#Conclusion: Good for quick exploration but the output downloaded plot for
loadings did not agree with the one shown online so I'm not sure how buggy it
is....(on another trial it worked...)
#Will explore loadings outside this...
```
![Alt text](results/pcaExplorer/pcae_hiload.pdf)
## Further exploration of the data using the PCAtools package from
Bioconductor
```{r EDA with PCAtools}
#BiocManager::install('PCAtools')
library(PCAtools)
p<- pca(assay(vsd), metadata=coldata)
screeplot(p)
biplot(p, colby="condition")
pairsplot(p, colby= "condition")
plotloadings(p)
~~~

loadings(p)[which.max(abs(loadings(p)$PC1)),]
loadings(p)[which.max(abs(loadings(p)$PC2)),]
~~~
Observations from PCA loadings plot:
- Largest contribution to PC1 is from gene WBGene00008862 (F15D4.5). There is very little annotation for this gene, except that it plays some regulatory role and interacts with deps-1, a gene localized to P granules. HHblits hits almost exclusively uncharacterised proteins but the C-terminal seems to map to CCHC-type domains (which are zinc fingers). It also hits some Pro-Pol polyproteins across a much larger part of the protein sequence (again at the C-terminus)
- Large negative loadings for PC1:
  WBGene00016627 (C44B7.5), an orthologue of human ferric chelate reductase 1. Human orthologues of this gene are implicated in early infantile epileptic encephalopathy 37.
  WBGene00021468 (epg-2), involved in macroautophagy and negative regulation of autophagosome assmebly.
- Largest (negative) loading for PC2:
  WBGene00010965 (ctc-2), cytochrome c oxidase activity, affected by daf-2
# Differential expression analysis based on isoform quantification from *kallisto*
## Dealing with ribosomal RNAs
Aine's work showed a huge number of reads mapping to rRNA.
I've decided it's probably best to remove them BEFORE proceeding with anything else so we don't have to worry about them later.
I saw that many people use bwa or bowtie2 for this but rRNAs of eukaryotes can also have introns so would need a spliced aligner.
Other solutions; BBsplit from BBmap can split reads from different references but does it based on k-mers. SortMeRNA also removes reads that map to
a set of sequences (used mostly with prokaryotes and metagenomes..would need to check if it works well with eukaryotes).
I decided in the end that the easiest solution might be to rebuild the index that kallisto uses and re-run kallisto so that the rRNAs are not part
of the reference transcriptome, allowing the reads to remain unmapped (hopefully!).
We need to remove all sequences with  rRNA transcript_biotype reads (there are 23) - the genome/transcriptome files are from Ensembl and can be found in this directory: **/d/in16/u/ubcg71a/research/genomes/c.elegans**
```{bash, eval = FALSE}
 zcat Caenorhabditis_elegans.WBcel235.cdna_and_ncrna.fa.gz | grep biotype:rRNA > list_of_biotype_rRNA_transcripts.txt
~~~
Found a one-liner online to do this easily (basically, stitches the lines together separated by "\t", greps and then
separates again with newlines:
```{bash, eval = FALSE}
```

~~~
zcat Caenorhabditis_elegans.WBcel235.cdna_and_ncrna.fa.gz | awk '{ if
((NR>1)&&($0~/^>/)) { printf("\n%s", $0); } else if (NR==1) { printf("%s",
$0); } else { printf("\t%s", $0); } }' | grep -vFf pattern.txt - | tr "\t"
"\n" > Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.fa
```
Checking:
```{bash, eval = FALSE}
grep ">" Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.fa | wc
61428
zcat Caenorhabditis_elegans.WBcel235.cdna_and_ncrna.fa.gz | grep ">" | wc
61451
wc list_of_biotype_rRNA_transcripts.txt
23
```
Then zip the final .fa file:
```{bash, eval = FALSE}
gzip -9 Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.fa
```
## Indexing the transcriptome with kallisto
(all transcriptome files are under:
**/d/in16/u/ubcg71a/research/genomes/c.elegans** See README for how they were
obtained):
```{bash, eval = FALSE}
/d/in7/s/kallisto/kallisto_linux-v0.46.1/kallisto index -i
Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.index
Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.fa.gz
```
## Pseudo-mapping with *kallisto*
This blog: http://genomespot.blogspot.com/2015/08/how-accurate-is-
kallisto.html
suggests that the difference between trimming before kallisto or not trimming
does not make a huge difference overall.
So I did not run Trimmomatic on top of fastp as I would have done if we were
mapping with STAR or similar.
Running first without pseudoalignment (which takes a lot more time and space)
but with bootstrapping (this is run in the directory
**$PROJECT/kallisto_with_ncRNA_norRNA/**):
```{bash, eval = FALSE}
nohup ./run_kallisto.sh ../fastp_merged/ N1_S5 N2_S6 N3_S7 N4_S8 SG1_S1
SG2_S2 SG3_S3 SG4_S4 >& run_kallisto.out &
```
I got initially rather low mapping rates (about 20%) and was worried I had
the orientation
of the reads wrong. I reversed the orientation and that gave less than 1%
alignment
so that was clearly not the problem. In this forum:
~~~

~~~
https://github.com/pachterlab/kallisto/issues/198
there is a suggestion that the culprit is ncRNAs that are not included in the
transcriptome file, which would make sense since we know that 80% of the
reads
are to rRNA in this case.
I re-did the run with a transcriptome that had ncRNA too. Now, alignment
jumped to
98.2% so the ncRNA missing was definitely the problem here (rRNA really..)
These results were under **kallisto_with_ncRNA**, a directory that I
eventually deleted to save space. The results without ncRNA are now renamed
to **kallisto_cDNA**
NOTE: the abundance estimates for the bootstrap are not written by default to
the
csv file but they can be written out using:
```{bash, eval = FALSE}
/d/in7/s/kallisto/kallisto_linux-v0.46.1/kallisto h5dump -o output
abundance.h5
```
(where output is a directory where the results will be saved)
I will probably keep both versions and run sleuth with both to see whether
there
are any differences that affect the coding genes too.
Finally, I re-did the run with a transcriptome that had ncRNA but no rRNA. I
then
deleted the previous run that had ncRNA, including rRNAs to save space.
So the kallisto_with_ncRNA directory has now been replaced with:
**kallisto_with_ncRNA_norRNA**
Alignment rates: p_unique: 9-10%, p_pseudoaligned: ~20%
I also did one re-run with ncRNA and no bootstrap but asking for pseudo-
alignment to
the genome so that we could get pseudo-bam files for visualisation:
In directory:
**/d/in16/u/ubcg71a/research/filipe/kallisto_with_ncRNA_norRNA/pseudo_bam/**
```{bash, eval =FALSE}
nohup ./run_kallisto_pseudobam.sh ../../fastp_merged/ N1_S5 N2_S6 N3_S7 N4_S8
SG1_S1 SG2_S2 SG3_S3 SG4_S4 >& run_kallisto.out &
```
## Gene-level differential expression using kallisto counts and sleuth
This approach is following the code from:
https://github.com/pachterlab/aggregationDE/blob/master/R/gene_pipeline
and the procedure of aggregating p-values after **transcript** differential
expression has been
carried out to call differentially expressed genes (see paper by Yi et al:
doi: 10.1186/s13059-018-1419-z).
### Setup directories and variables
```{r setup for sleuth and directories}
~~~

~~~
'%!in%' <- function(x,y)!('%in%'(x,y))
setwd("~/Dropbox/work/research/Filipe_neurexin/NEW_PROCESS/")
#sample_id <- dir(file.path(".", "kallisto_cdna"))
sample_id <- dir(file.path(".", "kallisto_with_ncRNA_norRNA"))
sample_id
#kal_dirs <- file.path(".", "kallisto_cdna", sample_id)
kal_dirs <- file.path(".", "kallisto_with_ncRNA_norRNA", sample_id)
```
### Read in metadata
Sleuth requires metadata information corresponding to the samples in the
dataset. In this case, the metadata is very simple and was already created
outside the R script. We simply
need to add to this table the directories where kallisto results are saved.
```{r metadata}
s2c <- read.table("./metadata.txt",  header = TRUE, stringsAsFactors=FALSE)
s2c
s2c <- dplyr::mutate(s2c, path = kal_dirs)
s2c
```
### Using biomaRt to map transcript ids to gene names and descriptions
We will get gene names from biomaRt to add to the transcript IDs we have from
sleuth
```{r biomart}
ens <- useMart("ensembl")
attr = c("ensembl_gene_id",
         "ensembl_transcript_id",
         "description",
         "external_gene_name",
         "entrezgene_id")
celeg <- useMart("ensembl",
                   dataset = "celegans_gene_ensembl")
t2g <- getBM(attributes = attr,
              mart = celeg)
t2g <- dplyr::rename(t2g,
                     target_id = ensembl_transcript_id,
                     ens_gene = ensembl_gene_id,
                     ext_gene = external_gene_name)
```
### Differential expression with sleuth
We first carry out differential expression with sleuth, summarised at the
gene level (using p-value aggregation).
```{r gene_diff_exp_transcript}
#### GENE-LEVEL-ANALYSIS ####
#Following code from:
https://github.com/pachterlab/aggregationDE/blob/master/R/gene_pipeline.R
design <- ~ condition
~~~

```
so <- sleuth_prep(s2c,
                  full_model = design,
                  read_bootstrap_tpm = TRUE,
                  extra_bootstrap_summary=TRUE,
                  target_mapping = t2g,
                  transform_fun_counts = function(x) log2(x + 0.5), #change
the natural log fold change to log2 fold change
                  aggregation_column = 'ens_gene'
                  #filter_target_id=txfilter
                  )
so <- sleuth_fit(so, ~condition, 'full')
#do a Wald test first
so <- sleuth_wt(so, which_beta='conditionneurexin', which_model='full')
sleuth_table_wt <- sleuth_results(so,
                                  test= 'conditionneurexin',
                                  test_type =  'wt',
                                  which_model = 'full',
                                  show_all = TRUE)
dim(sleuth_table_wt)
#how many of these genes have no NAs in q-value? (these are the ones that get
filtered when show_all is FALSE)
dim(filter(sleuth_table_wt, !is.na(qval)))
sleuth_table_wt[1:5,]
sleuth_significant_wt <- dplyr::filter(sleuth_table_wt, qval <= 0.05)
#significant genes: 2328
dim(sleuth_significant_wt)
head(sleuth_significant_wt, 20)
#and then a LRT test
so <- sleuth_fit(so, ~1, 'null')
so <- sleuth_lrt(so, 'null', 'full')
sleuth_table_lrt <- sleuth_results(so, test='null:full', 'lrt')
sleuth_table_lrt[1:5,]
sleuth_significant_lrt <- dplyr::filter(sleuth_table_lrt, qval <= 0.05)
dim(sleuth_significant_lrt)
head(sleuth_significant_lrt, 20)
#Note: The Wald test usually returns more hits than the likelihood ratio test
#check counts for top significant genes in WT
selected_gene <- sleuth_significant_wt$target_id[1]
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene==selected_gene))$target_id[1], units = "est_counts", color_by =
"condition")
selected_gene <- sleuth_significant_wt$target_id[2]
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene==selected_gene))$target_id[1], units = "est_counts", color_by =
"condition")
selected_gene <- sleuth_significant_wt$target_id[3]
```

~~~~
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene==selected_gene))$target_id[1], units = "est_counts", color_by =
"condition")
#save the gene differential expression results so they can be used with web
servers for further analysis
#check the table has no duplicate entries
length(unique(sleuth_table_wt$target_id)) ==
length(sleuth_table_wt$target_id)
write.table(sleuth_significant_wt$target_id,
file="results/sleuth_significant_q005_NEW.txt", quote=FALSE, row.names =
FALSE, col.names = "#target_id")
#avoid using this - pointless
write.table(dplyr::filter(sleuth_table_wt, qval > 0.05)$target_id,
file="results/bg_genes_good.txt", quote=FALSE, row.names = FALSE, col.names =
"#target_id")
#use this instead
write.table(dplyr::filter(sleuth_table_wt, !is.na(qval))$target_id,
file="results/bg_genes_all.txt", quote=FALSE, row.names = FALSE, col.names =
"#target_id")
```
Next, we extract differential expression results from the same sleuth object
but this time we do not aggregate p-values so our target IDs are transcript
IDs.
```{r transcript_diff_expression}
##### TRANSCRIPT-LEVEL ANALYSIS #####
#To do the transcript-level analysis, simply set the pval_aggregate to FALSE
#NOTE that target_id will now be the transcript id rather than the gene id
#We will use the Wald test only from here on
sleuth_table_txn <- sleuth_results(so,
                                   test = 'conditionneurexin',
                                   test_type = 'wt',
                                   which_model = 'full',
                                   show_all = TRUE,
                                   pval_aggregate = FALSE)
dim(sleuth_table_txn)
#how many of these genes have no NAs in q-value? (these are the ones that get
filtered when show_all is FALSE) - 15845
dim(filter(sleuth_table_txn, !is.na(qval)))
sleuth_table_txn[1:5,]
sleuth_significant_txn <- dplyr::filter(sleuth_table_txn, qval <= 0.05)
dim(sleuth_significant_txn) # 2263 transcripts are significant
head(sleuth_significant_txn, 20)
write.table(sleuth_significant_txn$target_id,
file="results/sleuth_significant_TXN_q005.txt", quote=FALSE, row.names =
FALSE, col.names = "#target_id")  #2263 transcripts written
~~~~

```
#the transcript table contains beta values too so one can talk about up and
down regulated transcripts and genes
sleuth_significant_txn.up <- dplyr::filter(sleuth_table_txn, (qval <= 0.05) &
(b>0))  # 905 are upregulated
head(sleuth_significant_txn.up, 20)
#how many genes are upregulated? - 889
length(unique(sleuth_significant_txn.up$ens_gene))
#we will save the gene ids instead of transcript ones
write.table(unique(sleuth_significant_txn.up$ens_gene),
file="results/sleuth_significant_q005_up.txt", quote=FALSE, row.names =
FALSE, col.names = "#ens_gene")
sleuth_significant_txn.down <- dplyr::filter(sleuth_table_txn, (qval <= 0.05)
& (b<0)) # 1358 are downregulated
head(sleuth_significant_txn.down, 20)
#how many genes are downpregulated? - 1334
length(unique(sleuth_significant_txn.down$ens_gene))
write.table(unique(sleuth_significant_txn.down$ens_gene),
file="results/sleuth_significant_q005_down.txt", quote=FALSE, row.names =
FALSE, col.names = "#ens_gene")
######## SUMMARY OF NUMBERS  ##########
#all transcripts and genes included initially in the analysis: 61428
transcripts; 46904 genes
length(sleuth_table_txn$target_id)
length(sleuth_table_wt$target_id)
#note that numbers of genes in the transcript table and number of genes in
the gene table are not exactly equal
#there are 46881 genes in the transcript table
length(unique(sleuth_table_txn$ens_gene))
#and 46904 genes in the gene table
table(sleuth_table_wt$target_id %!in% unique(sleuth_table_txn$ens_gene))
#which genes are missing?
extras <- sleuth_table_wt$target_id[sleuth_table_wt$target_id %!in%
unique(sleuth_table_txn$ens_gene)]
#why? they are entries with NAs (mostly rrns) so would have been filtered
anyway
filter(sleuth_table_wt, sleuth_table_wt$target_id %in%  extras)

#number of transcripts (15845) and genes (11337) with sleuth data in the gene
p-value aggregation table i.e. filtered genes and transcripts
sum(sleuth_table_wt$num_aggregated_transcripts, na.rm=TRUE )
length(sleuth_table_wt[!is.na(sleuth_table_wt$num_aggregated_transcripts),]
$target_id)
#significantly DE genes (aggregated) in the gene results table at q<0.05 -
2328 genes
length(sleuth_significant_wt$target_id)
#significantly DE transcripts in the transcript results table - 2263
```

```
length(sleuth_significant_txn$target_id)
#how many genes correspond to these 2263 transcripts? - 2217
length(unique(sleuth_significant_txn$ens_gene))
########### PROBLEM???
#Need to check this with Pachter's blogs and guidelines as the transcript and
gene results are supposed
#to be fully compatible (but I think it's fine because the top DE genes and
genes corresponding to top DE
#transcripts seem to agree..it's marginal cases that seem to be the problem)
#genes in common:
length(intersect(unique(sleuth_significant_txn$ens_gene),
sleuth_significant_wt$target_id)) # answer is 2107
#which genes are not in the transcript table but are in the gene table as
differentially expressed?
#genes unique to the transcript table:
length(setdiff(unique(sleuth_significant_txn$ens_gene),
sleuth_significant_wt$target_id)) #answer is 110
#genes unique to the gene table:
length(setdiff(sleuth_significant_wt$target_id,
unique(sleuth_significant_txn$ens_gene))) #answer is 221
#compare the very top results - they are identical up to top 50 and start
having more differences as they build up...
length(intersect(sleuth_significant_wt$ext_gene[1:50],
unique(sleuth_significant_txn$ext_gene)[1:50]))
length(intersect(sleuth_significant_wt$ext_gene[1:100],
unique(sleuth_significant_txn$ext_gene)[1:100]))
length(intersect(sleuth_significant_wt$ext_gene[1:500],
unique(sleuth_significant_txn$ext_gene)[1:500]))
###################
#significantly DE transcripts with large fold changes (abs(b)>1) - 480
transcripts
dim(filter(sleuth_significant_txn, abs(b) > 1))
#how many genes correspond to these? - 469 genes
length(filter(sleuth_significant_txn, abs(b) > 1)$ens_gene)
length(unique(filter(sleuth_significant_txn, abs(b) > 1)$ens_gene))
#volcano plot for genes
#Note: there is a problem with volcano and MA plots not setting hte p-value
aggregate to FALSE when reading
#results from the WT test and thus not realising there is a b column
#Error reported is:
#Error in FUN(X[[i]], ...) : object 'b' not found
#To fix this, manually set the pval_aggregate in so before supplying to plots
as suggested here:
#https://github.com/pachterlab/sleuth/issues/233
# so$pval_aggregate <- FALSE
# plot_volcano(obj=so,
        # test = "conditionneurexin",
        # test_type = "wt",
        # which_model="full",
        # sig_level = 0.05,
        # sig_color = "red")
#can also do MA plot
# plot_ma(obj=so,
        # test = "conditionneurexin",
        # test_type = "wt",
        # which_model="full",
        # sig_level = 0.05,
        # sig_color = "red")
# ALTERNATIVELY
#use the EnhancedVolcano package and function to make nicer plots and to also
highlight
#genes easily (the highlight function in plot_volcano from sleuth is not
working for me)
#not used as using cutoffs within the volcano plot options
#select.lab <- head(sleuth_significant_txn, 20)$target_id
#use the same function that plot_volcano() uses to produce the data frame
needed to make the plot
#we could use also the results table from before and remove the NAs but this
is cleaner
toptable <- sleuth_results(so,
                           test="conditionneurexin",
                           test_type="wt",
                           which_model="full",
                           pval_aggregate=FALSE,
                           rename_cols=FALSE,
                           show_all=FALSE)
dim(toptable) #should be the "filtered" transcripts, i.e. 15845
#add a column that the volcano plot is using later
toptable <- dplyr::mutate(toptable, significant = qval < 0.05)
dim(dplyr::filter(toptable, abs(b)>1 & (significant == TRUE))) #this should
be 480 transcripts, as before
#FIGURE fig_volcano_txn
EnhancedVolcano(
     toptable=toptable,
     lab=toptable$target_id,
     x='b',
     y='qval',
     xlab="beta_value",
     selectLab = NULL,
     title="Sleuth results (without p-value aggregation)",
     subtitle="",
     legend=c("NS","beta","P","P & beta"),
```

~~~~
     legendLabels=c("NS","beta","P","P & beta"),
     transcriptLabSize=5.0,
     pLabellingCutoff=10e-6,
     FCcutoff=1.0
     )
#create a table with information about top 10 most significant genes
select.res <- dplyr::filter(toptable, abs(b)>1)[1:10,c(1,2,5,6,7)]
dplyr::filter(t2g, target_id %in% select.res$target_id)
merge(x=select.res, y=dplyr::filter(t2g, target_id %in%
select.res$target_id), by="target_id", sort=FALSE,
no.dups=TRUE)[,c("target_id","ext_gene","entrezgene_id.x","qval","description
")]
#Examine some bootstrapped counts for individual transcripts of the same gene
(note these are log2 transformed by sleuth)
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene=="WBGene00019606"))$target_id[1], units = "est_counts", color_by =
"condition")
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene=="WBGene00019606"))$target_id[2], units = "est_counts", color_by =
"condition")
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene=="WBGene00019606"))$target_id[3], units = "est_counts", color_by =
"condition")
plot_bootstrap(so, target_id=(dplyr::filter(t2g,
ens_gene=="WBGene00019606"))$target_id[4], units = "est_counts", color_by =
"condition")
#use the shiny app when not knitting the document
#sleuth_live(so)
```
**A note on sleuth counts**
The "raw" counts of sleuth correspond more or less to assay(dds) in DESeq2.
The "est_counts" from the normalised data frame (so[["obs_norm""]]) are on a
similar scale to raw counts and correlate well.
The tpm values take into account the length of the genes/transcripts and show
reduced correlation to the est_counts.
Neither est_counts or tpm values are stored as log normalised - a log must be
taken, if one wants to have logged values.
However, when the diff expression results are obtained the mean_obs is
a value obtained as the log (or log2 if the transformation function has been
changed to log2) of the mean of observations across all samples.
As the est_counts are not log-transformed, the PCA plots based on them are
likely dominated simply by the genes with very high read counts.
```{r check_counts}
#pick a gene with large counts
t2g[t2g$target_id == "ZK185.9",]
#DEseq2 counts
~~~~

~~~
assay(dds)["WBGene00220238",]
#log2-transformed
log2(assay(dds)["WBGene00220238",])
#rlog-transformed
assay(rld)["WBGene00220238",]
#raw counts from sleuth
filter(so[["obs_raw"]], so[["obs_raw"]]$target_id=="ZK185.9")$est_counts
#normalised counts from_sleuth
filter(so[["obs_norm"]], so[["obs_norm"]]$target_id=="ZK185.9")$est_counts
#log2 and normalised counts from sleuth
mean(log2(filter(so[["obs_norm"]],
so[["obs_norm"]]$target_id=="ZK185.9")$est_counts))
#mean_obs from sleuth transcript table results
filter(sleuth_table_txn, target_id=="ZK185.9")$mean_obs
#sleuth_matrix results should agree
sleuth_matrix <- sleuth_to_matrix(so, 'obs_norm', 'est_counts')
sleuth_matrix.raw <- sleuth_to_matrix(so, 'obs_raw', 'est_counts')
sleuth_matrix.tpm <- sleuth_to_matrix(so, 'obs_norm', 'tpm')
sleuth_matrix[59889,]
log2(mean(sleuth_matrix[59889,]))
log2(mean(sleuth_matrix.raw[59889,]))
sleuth_matrix.tpm[59889,]
```
### Exploratory Data Analysis with *kallisto/sleuth*
```{r EDA_sleuth}
#Here, we check PCA plots based on kallisto counts
#NOTE: all PCA plots below are done with the transcript counts as
sleuth_to_matrix won't return gene-summarised counts
#first, using sleuth's own plot_pca (this, it seems, uses transcript counts
after all)
#In addition, note that est_counts in sleuth are not log2 normalised
#see:
https://hbctraining.github.io/DGE_workshop_salmon/lessons/09_sleuth.html
plot_pca(so,
            color_by = 'condition',
            text_labels = TRUE)
#and then using the PCAtools package
#need first the counts table (which_units would be scaled_reads_per_base for
genes but it doesn't work for aggregation of pvalues (probably only for the
old gene_mode=true))
#sleuth_matrix <- sleuth_to_matrix(so, 'obs_norm', 'scaled_reads_per_base')
sleuth_matrix <- sleuth_to_matrix(so, 'obs_norm', 'est_counts')
dim(sleuth_matrix) #clearly this is counts of transcripts, not genes
#alternative (this is how plot_pca in sleuth gets the matrix) - THIS WON'T
WORK because sleuth.R does
#not make the spread_abundance_by function available...
~~~

```
#mat <- spread_abundance_by(so$obs_norm_filt, 'est_counts',
#            so$sample_to_covariates$sample)
#will remove using removeVar a % of variables with low variance: we want to
keep about 500 transcripts and we have about 60,000 so we will remove 99% of
all variables
p<- pca(sleuth_matrix, metadata=coldata, removeVar=0.99)
dim(p$loadings) #should give about 600 transcripts
#this is essentially the same PCA as by doing: prcomp(t(sleuth_matrix)),
scale=FALSE, center=TRUE)
screeplot(p)
#the PCA plot is virtually identical to the one drawn by sleuth's plot_pca()
biplot(p, colby="condition", labSize=4.0, widthConnectors=0.2, xlim=c(-25000,
20000), title="Kallisto est_counts")
pairsplot(p, colby= "condition")
plotloadings(p)
#get the top 20 for PC1 and PC2
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20]
#or getting all the info about these transcripts
t2g[t2g$target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20],]
rownames(p$loadings)[order(abs(p$loadings[,2]))][1:20]
t2g[t2g$target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,2]))][1:20],]
#NOTE: the est_counts are not logged in the PCA so we expect the largest
counts to dominate as they have more variance in RNA-seq data
quantile(sleuth_table_txn$mean_obs, na.rm=TRUE)
quantile(sleuth_table_txn$var_obs, na.rm=TRUE)
filter(sleuth_table_txn, target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20])  #counts here are
log2 counts
filter(so[["obs_norm"]], target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20])
#Repeat wiith kallisto TPMs
#note that there is only partial correlation between estimated counts and
TPMs (TPMs take into account the gene length; est_counts seem to be
normalised by sleuth when obs_norm is used)
sleuth_matrix.tpm <- sleuth_to_matrix(so, 'obs_norm', 'tpm')
dim(sleuth_matrix.tpm) #clearly this is counts of transcripts, not genes
cor(sleuth_matrix.tpm[,1], sleuth_matrix[,1])
#will remove using removeVar a % of variables with low variance: we want to
keep about 500 transcripts and we have about 60,000 so we will remove 99% of
all variables
p.tpm<- pca(sleuth_matrix.tpm, metadata=coldata, removeVar=0.99)
dim(p.tpm$loadings) #should give about 600 transcripts
#this is essentially the same PCA as by doing: prcomp(t(sleuth_matrix)),
scale=FALSE, center=TRUE)
```

```
screeplot(p.tpm)
#FIGURE: PCA_tpm_counts
#the PCA plot is virtually identical to the one drawn by sleuth's plot_pca()
biplot(p.tpm, colby="condition", labSize=4.0, widthConnectors=0.2,
title="Kallisto TPM; top 1% varied transcripts")
pairsplot(p.tpm, colby= "condition")
#this command seems to hang for tpm values so  I'm not running it
#plotloadings(p)
#instead we can plot the top 20 for PC1 and PC2
rownames(p.tpm$loadings)[order(abs(p$loadings[,1]))][1:20]
t2g[t2g$target_id %in%
rownames(p.tpm$loadings)[order(abs(p.tpm$loadings[,1]))][1:20],]
rownames(p.tpm$loadings)[order(abs(p.tpm$loadings[,2]))][1:20]
t2g[t2g$target_id %in%
rownames(p.tpm$loadings)[order(abs(p.tpm$loadings[,2]))][1:20],]
quantile(sleuth_table_txn$mean_obs, na.rm=TRUE)
quantile(sleuth_table_txn$var_obs, na.rm=TRUE)
filter(sleuth_table_txn, target_id %in%
rownames(p.tpm$loadings)[order(abs(p.tpm$loadings[,1]))][1:20])
#repeating the plot without removing any transcripts makes no real difference
p.tpm0<- pca(sleuth_matrix.tpm, metadata=coldata, removeVar=0)
dim(p.tpm0$loadings) #should give about 600 transcripts
biplot(p.tpm0, colby="condition", labSize=4.0, widthConnectors=0.2,
title="Kallisto TPM; all transcripts included")
## WHAT HAPPENS IF WE LOG THE VALUES FOR est_counts?
#finally, repeat the PCA but this time using log2-transformed values for the
est_counts
p<- pca(log2(sleuth_matrix+0.5), metadata=coldata, removeVar=0.99)
dim(p$loadings) #should give about 600 transcripts
#this is essentially the same PCA as by doing: prcomp(t(sleuth_matrix)),
scale=FALSE, center=TRUE)
screeplot(p)
#in this case the PC2 separates the samples
biplot(p, colby="condition", labSize=4.0, widthConnectors=0.2,
title="Kallisto log2(est_counts)")
pairsplot(p, colby= "condition")
plotloadings(p)
#get the top 20 for PC1 and PC2
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20]
#or getting all the info about these transcripts
t2g[t2g$target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20],]
rownames(p$loadings)[order(abs(p$loadings[,2]))][1:20]
t2g[t2g$target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,2]))][1:20],]
```

```
#NOTE: when log2 is used with est_counts, it will be the lower counts that
dominate the PCA as they will have the larger variance due to Poisson noise
quantile(sleuth_table_txn$mean_obs, na.rm=TRUE)
quantile(sleuth_table_txn$var_obs, na.rm=TRUE)
filter(sleuth_table_txn, target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20])
#the distribution of est_counts (not logged) for the important loadings in
PC1
quantile(filter(so[["obs_norm"]], target_id %in%
rownames(p$loadings)[order(abs(p$loadings[,1]))][1:20])$est_counts)
#the distribution of est_counts (not_logged) across all transcripts
quantile(so[["obs_norm"]]$est_counts)
#Some additional exploratory analysis using sleuth's own functions
plot_pca(so,
         color_by = 'condition',
         text_labels = TRUE)
#the two types of samples are separated but only just (the SGs cluster
together though)
plot_pca(so,
          color_by = 'condition',
          text_labels = TRUE, units="tpm")
plot_pca(so,
         pc_x=2,
         pc_y=3,
         color_by = 'condition',
         text_labels = TRUE)
plot_pca(so,
         pc_x=2,
         pc_y=8,
         color_by = 'condition',
         text_labels = TRUE)
#how much of the variance is accounted for by each PC?
plot_pc_variance(obj = so, units = "est_counts")
plot_pc_variance(obj = so, units = "tpm")
#note PC8 also separates the samples
plot_loadings(so, pc_input=1)
plot_loadings(so, pc_input=2)
plot_loadings(so, pc_input=3)
plot_loadings(so, pc_input=8)
#note that all influential genes in PCA seem to be very highly expressed
(over 1000 tpm)
plot_group_density(so,
                   use_filtered = FALSE,
                   units = "est_counts",
                   trans = "log",
                   grouping = "condition")
```

```
plot_group_density(so,
                   use_filtered = TRUE,
                   units = "est_counts",
                   trans = "log",
                   grouping = "condition")
plot_sample_heatmap(so)
#check whether there are sex differences
#list of sex determining genes is from:
#https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2619291/pdf/324.pdf
list.sex_genes <- c("xol-1", "sdc-1", "sdc-2",
                    "her-1", "tra-2", "tra-3",
                    "fem-1", "fem-2", "fem-3", "tra-1")
plot_transcript_heatmap(so,
                        transcripts=dplyr::filter(sleuth_table_txn,
                                                  ext_gene %in%
list.sex_genes)$target_id)
#repeat the plot with est_counts instead of tpm
plot_transcript_heatmap(so,
                        transcripts=dplyr::filter(sleuth_table_txn,
                                                  ext_gene %in%
list.sex_genes)$target_id,
                        units="est_counts")
#ZK287.8b.1 seems to separate two of the normal samples but it's a transcript
that
#is not used in differential expression due to filtering - below are original
counts
#note that basic filter in sleuth filters out any transcripts with fewer than
5 estimated counts
#in 47% of the samples (effectively here half of the samples)
so$obs_raw[so$obs_norm$target_id == "ZK287.8b.1",]
```
A note on the variability of PCA plots
We have plotted a number of PCA plots in the code above. Most look similar but there are also some striking differences. What drives most of the differences is that some of the plots have used log-transformed values of counts and others haven't.
The est_counts in the so object's obs_norm and obs_raw data frames are not log normalised (however when they are shown in the results table, the mean observations are indeed logged (in this case transformed by log2 as this is how we built the object)). Hence, when we are using est_counts extracted with sleuth_to_matrix, these counts are not in a log scale. Some are indeed very large and will dominate the PCA plot (they also have very large variance in RNA-seq). If we log these values, then the lower counts will dominate the PCA as they will have the larger variance due to Poisson noise.

## Functional analysis

~~~
Now that we have a list of significant differentially expressed genes (DGEs),
we can  try to gain biological insight by visualising our results in a proper
context. We will use mainly the library `clusterProfiler` for this, which has
a variety of options for viewing the over-represented Gene Ontology (GO)
terms. The layout of this part of the code is largely based on the tutorial
found
[here](https://hbctraining.github.io/DGE_workshop_salmon/lessons/functional_a
nalysis_2019.html).
### Quick GO over-representation analysis using goseq
Below we apply goseq to find overrepresented GO categories among
differentially expressed genes.
We will use the genes from the aggregated p-value sleuth run.
```{r pathway enrichment analysis with goseq}
#collect the results of differential expression
#first the background list of genes (this will all be filtered genes in the
analysis i.e. genes with no NAs in their qvalues)
bg <- dplyr::filter(sleuth_table_wt, !is.na(qval))$ext_gene  #there should be
11337 genes in the background
#then select significant genes with qval <= 0.05
sig.q005 <- (dplyr::filter(sleuth_table_wt, qval <= 0.05))$ext_gene  #2328
genes
t.q005.vector <- as.integer(bg %in% sig.q005)  #vector that specifies which
genes are in the DE set and which not (length 11337)
names(t.q005.vector) <- dplyr::filter(sleuth_table_wt, !is.na(qval))$ext_gene
#sanity check
table(t.q005.vector)
#calculate the object containing the gene names, DE calls and probability
weighting function.
pwf.q005=nullp(t.q005.vector,"ce6","geneSymbol")
subset(pwf.q005, !is.na(pwf.q005$pwf)) -> pwf.q005.noNA
q005.GO.wall.BP=goseq(pwf.q005.noNA,"ce6","geneSymbol", test.cats =
c("GO:BP"))
#q005.GO.wall.BP[(p.adjust(q005.GO.wall.BP$over_represented_pvalue,method="BH
")<.05),c(2,3,6)]
q005.GO.wall.MF=goseq(pwf.q005,"ce6","geneSymbol", test.cats = c("GO:MF"))
#q005.GO.wall.MF[(p.adjust(q005.GO.wall.MF$over_represented_pvalue,method="BH
")<.05),c(2,3,6)]
q005.GO.wall.CC=goseq(pwf.q005,"ce6","geneSymbol", test.cats = c("GO:CC"))
#q005.GO.wall.CC[(p.adjust(q005.GO.wall.CC$over_represented_pvalue,method="BH
")<.05),c(2,3,6)]
#filter for FDR<0.05 and min/max number of genes in each category
q005.GO.wall.BP.filtered <- q005.GO.wall.BP
q005.GO.wall.BP.filtered$adjp <-
p.adjust(q005.GO.wall.BP$over_represented_pvalue,method="BH")
q005.GO.wall.BP.filtered <- dplyr::filter(q005.GO.wall.BP.filtered, adjp <
0.05)
~~~

~~~~
q005.GO.wall.BP.filtered <- dplyr::filter(q005.GO.wall.BP.filtered, (numInCat
< 500) & (numInCat>5))
q005.GO.wall.BP.filtered
q005.GO.wall.MF.filtered <- q005.GO.wall.MF
q005.GO.wall.MF.filtered$adjp <-
p.adjust(q005.GO.wall.MF$over_represented_pvalue,method="BH")
q005.GO.wall.MF.filtered <- dplyr::filter(q005.GO.wall.MF.filtered, adjp <
0.05)
q005.GO.wall.MF.filtered <- dplyr::filter(q005.GO.wall.MF.filtered, (numInCat
<500) & (numInCat>5))
q005.GO.wall.MF.filtered
q005.GO.wall.CC.filtered <- q005.GO.wall.CC
q005.GO.wall.CC.filtered$adjp <-
p.adjust(q005.GO.wall.CC$over_represented_pvalue,method="BH")
q005.GO.wall.CC.filtered <- dplyr::filter(q005.GO.wall.CC.filtered, adjp <
0.05)
q005.GO.wall.CC.filtered <- dplyr::filter(q005.GO.wall.CC.filtered, (numInCat
< 500) & (numInCat>5))
q005.GO.wall.CC.filtered
```
### GO over-representation analysis with clusterProfiler (using gene-level
sleuth results)
Much of this code has been adapted from Diego's analysis.
We apply the various clusterProfiler functions to explore the data.
```{r GO enrichment with clusterProfiler}
# Run GO enrichment analysis and save results
ego <- enrichGO(gene = sleuth_significant_wt$target_id,
                universe = dplyr::filter(sleuth_table_wt,
!is.na(qval))$target_id,
                keyType = "ENSEMBL",
                OrgDb = org.Ce.eg.db,
                ont = "ALL",
                pAdjustMethod = "BH",
                qvalueCutoff = 0.05,
                readable = TRUE)
cluster_summary <- data.frame(ego)
write.csv(cluster_summary, "results/clusterProfiler_ALL_neurexin.csv")
ego <- enrichGO(gene = sleuth_significant_wt$target_id,
                universe = dplyr::filter(sleuth_table_wt,
!is.na(qval))$target_id,
                keyType = "ENSEMBL",
                OrgDb = org.Ce.eg.db,
                ont = "BP",
                pAdjustMethod = "BH",
                qvalueCutoff = 0.05,
                readable = TRUE)
~~~~

~~~
cluster_summary <- data.frame(ego)
write.csv(cluster_summary, "results/clusterProfiler_BP_neurexin.csv")
#enrichment dotplot
enrichment_dotplot <- clusterProfiler::dotplot(ego,
                                               x="GeneRatio",
                                               showCategory = 30)
plot(enrichment_dotplot)
ggsave("results/enrichment_dotplot.svg", plot = enrichment_dotplot, path =
".",
       width = 30, height = 15, units = "cm")
#enrichment mapplot
#this plot visualises gene sets as a network
#(mutually overlapping gene sets tend to cluster together)
clusterProfiler::emapplot(ego,
                          showCategory = 30)
```
### GO over-representation analysis with clusterProfiler (using transcript-
level sleuth results)
An additional plot that ClusterProfiler allows us to do is a category
netplot. These plots
are particularly useful for hypothesis generation in identifying genes
that may be important to several of the most affected processes.
These plots use fold change but this is not available in gene-level analysis
in sleuth.
So instead, we'll use transcript-based results and use beta values as the
fold changes.
```{r netplot with transcript fold changes}
#netplot
# We need to rerun enrichGO with the transcript-based information instead
# This is any way more correct as bias corrections based on length are best
done using
# the individual transcript info rather than some mean length for the gene
# first, create a df without duplicate genes (first row is kept and so most
important transcript should be kept in the df)
df <- sleuth_significant_txn %>% distinct(ens_gene, .keep_all=TRUE)
# check it worked
dim(df)
length(unique(sleuth_significant_txn$ens_gene)) == dim(df)[1]
# now re-run enrichGO using these genes instead
ego.txn <- enrichGO(gene = df$ens_gene,
                universe = unique(dplyr::filter(sleuth_table_txn,
!is.na(qval))$ens_gene),
                keyType = "ENSEMBL",
                OrgDb = org.Ce.eg.db,
                ont = "BP",
                pAdjustMethod = "BH",
~~~

```
                qvalueCutoff = 0.05,
                readable = TRUE)
#repeat quickly the dotplot and emapplot to compare them to the ones produced
from the gene table
#they should be very similar
enrichment_dotplot <- clusterProfiler::dotplot(ego.txn,
                                               x="GeneRatio",
                                               showCategory = 30)
plot(enrichment_dotplot)
ggsave("results/enrichment_dotplot.svg", plot = enrichment_dotplot, path =
".",
       width = 30, height = 15, units = "cm")
#FIGURE fig_emapplot
enrichment_mapplot <- clusterProfiler::emapplot(ego.txn,
                                                showCategory = 30)
plot(enrichment_mapplot)
ggsave("results/enrichment_mapplot.svg", plot = enrichment_mapplot, path =
".")
# A named character vector is needed to colour genes in the net plot
genes_foldchange <- df$b
names(genes_foldchange) <- df$ext_gene
# Netplot
enrichment_netplot <- cnetplot(ego.txn,
                               categorySize = "pvalue",
                               showCategory = 5,
                               foldChange = genes_foldchange,
                               vertex.label.font = 6, drop=TRUE)
plot(enrichment_netplot)
ggsave("results/enrichment_netplot.svg", plot = enrichment_netplot, path =
".",
       width = 40, height = 40, units = "cm")
#remove the gene names for clarity (showing more categories doesn't help much
as they fall on top of each other)
enrichment_netplot <- cnetplot(ego.txn,
          categorySize = "pvalue",
          showCategory = 5,
          foldChange = genes_foldchange,
          vertex.label.font = 6, node_label="category")
plot(enrichment_netplot)
ggsave("results/enrichment_netplot_cleaner.svg", plot = enrichment_netplot,
path = ".",
       width = 40, height = 40, units = "cm")
#showing 8 categories
cnetplot(ego.txn,
          categorySize = "pvalue",
          showCategory = 8,
          foldChange = genes_foldchange,
          vertex.label.font = 6, node_label="category")
#relationship between the GO terms identified (haven't found a way to make
the writing bigger!)
go_graph <- plotGOgraph(ego.txn)
#goplot (relationships between GO categories)
goplot(ego.txn)
#upsetplot (requires enrichplot and ggupset) - number of genes overlapping
different data sets
upsetplot(ego.txn)
#As centplots are very crowded, we can focus on subsets of interest
#First, focus on RNA terms
ego.sub <- ego.txn
descriptions <- ego.txn@result$Description
go_term1 <- which(base::grepl("RNA", descriptions) == TRUE)
#go_term2 <- which(base::grepl("splic", descriptions) == TRUE)
#ego.sub@result <- ego.sub@result[c(go_term1, go_term2), ]
ego.sub@result <- ego.sub@result[c(go_term1), ]
# Plotting terms of interest
enrichment_netplot.sub <- cnetplot(ego.sub,
                                categorySize = "pvalue",
                                foldChange = genes_foldchange,
                                showCategory = 5)
plot(enrichment_netplot.sub)
ggsave("enrichment_netplot.sub.svg", plot = enrichment_netplot.sub, path =
"./results",
       width = 25, height = 20, units = "cm")
#Next, focus on "catabolic" processes
ego.sub <- ego.txn
descriptions <- ego.txn@result$Description
go_term1 <- which(base::grepl("catabolic", descriptions) == TRUE)
ego.sub@result <- ego.sub@result[c(go_term1), ]
enrichment_netplot.sub2 <- cnetplot(ego.sub,
                                categorySize = "pvalue",
                                foldChange = genes_foldchange,
                                showCategory = 5)
plot(enrichment_netplot.sub2)
ggsave("enrichment_netplot.sub2.svg", plot = enrichment_netplot.sub2, path =
"./results",
       width = 25, height = 20, units = "cm")
#Next, focus on chromosome
ego.sub <- ego.txn
descriptions <- ego.txn@result$Description
go_term1 <- which(base::grepl("chromosom", descriptions) == TRUE)
ego.sub@result <- ego.sub@result[c(go_term1), ]
enrichment_netplot.sub3 <- cnetplot(ego.sub,
```

~~~~
                                categorySize = "pvalue",
                                foldChange = genes_foldchange,
                                showCategory = 5)
plot(enrichment_netplot.sub3)
ggsave("enrichment_netplot.sub3.svg", plot = enrichment_netplot.sub3, path =
"./results",
       width = 25, height = 20, units = "cm")
#put all the plots together using cowplot
cowplot::plot_grid(enrichment_netplot, enrichment_netplot.sub,
enrichment_netplot.sub2, enrichment_netplot.sub3, ncol=2,
labels=LETTERS[1:4])
#Finally, repeat using ALL categories - molecular function gives no results
(when run on its
#own gives a few non-significant hits)
ego.txn <- enrichGO(gene = df$ens_gene,
                universe = unique(dplyr::filter(sleuth_table_txn,
!is.na(qval))$ens_gene),
                keyType = "ENSEMBL",
                OrgDb = org.Ce.eg.db,
                ont = "ALL",
                pAdjustMethod = "BH",
                qvalueCutoff = 0.05,
                readable = TRUE)
enrichment_dotplot <- clusterProfiler::dotplot(ego.txn,
                                               x="GeneRatio",
                                               showCategory = 30)
plot(enrichment_dotplot)
ggsave("results/enrichment_dotplot_ALL.svg", plot = enrichment_dotplot, path
= ".",
       width = 30, height = 15, units = "cm")
```
### GSEA analysis with clusterProfiler (using transcript-level sleuth
results)
*Analysis adapted from the original done by Diego*
GSEA methods suggest that weaker but coordinated changes in sets of
functionally related genes (i.e., pathways) can also have significant effects
and might not be picked up by analysis of lists of genes that have been
selected using what is essentially a random p-value cutoff. Thus, rather than
setting an arbitrary threshold to identify ‘significant genes’, **all genes
are considered in the GSEA analysis**.
To perform Gene Set Enrichment Analysis (GSEA) of KEGG gene sets,
clusterProfiler requires the genes to be identified using Entrez IDs for all
genes in our results dataset. We also need to remove the NA values and
duplicates (due to gene ID conversion) prior to the analysis.
See also this blog: https://learn.gencore.bio.nyu.edu/rna-seq-analysis/over-
representation-analysis/
~~~~

~~~
** DO NOT RUN GSEA YET **
```{r GSEA with clusterProfiler, eval=FALSE, echo=FALSE}
# Remove any NA or duplicated values
t<- dplyr::filter(sleuth_table_txn, (!is.na(qval)) & (!is.na(entrezgene_id)))
dim(t)  #15382 results remaining
#some gene ids will be duplicates as this was the transcripts table
length(t$entrezgene_id)  #15382 lines remaining
length(unique(t$entrezgene_id))  #10888 unique genes
#filter out the duplicate entrez ids
t<- t %>% distinct(entrezgene_id, .keep_all=TRUE)
dim(t) #should be 10888 lines left
#the required input seems to be a vector, often pvalues or log fold
changes(?), named by the gene Entrez ids
#list needs to be named by the gene ids
gene_list <- abs(t$pval)
names(gene_list) <- t$entrezgene_id
#and sorted in decreasing order
gene_list <- sort(gene_list, decreasing = TRUE)
# GSEA of KEGG
gseaKEGG <- gseKEGG(gene_list,
                    organism = "cel",
                    keyType = "ncbi-geneid",
                    pvalueCutoff = 0.05,
                    verbose = FALSE)
gseaKEGG_results <- gseaKEGG@result
write.csv(gseaKEGG_results, "results/gsea_kegg.csv", quote = F)
~~~
### KEGG pathway enrichment with clusterProfiler
```{r clusterprofiler_kegg}
#check the kegg code for c.elegans
search_kegg_organism('cel', by='kegg_code')
#entrez ids not recognised in this case
sigGenes <- sleuth_significant_txn %>% distinct(ext_gene, .keep_all=FALSE)
sigGenes[,1] <- paste("CELE", sigGenes[,1],sep = "_")
dim(sigGenes)
universe <- sleuth_table_txn %>% filter(!is.na(ext_gene) & !is.na(qval)) %>%
distinct(ext_gene, .keep_all=FALSE)
universe[,1] <- paste("CELE", universe[,1], sep="_")
dim(universe)
keg_res <- enrichKEGG(gene=sigGenes[,1],
                      organism="cel",
                      keyType="kegg",
                      universe= universe[,1],
                      minGSSize = 2, #default is 10
                      qvalueCutoff = 0.1)
head(keg_res)
```

~~~~
#this does not appear to return any results, even if the minGSSize is set to
1 and the qvalue cutoff further relaxed.
#however, setting the universe to all C. elegans genes in KEGG returns three
#pathways
keg_res <- enrichKEGG(gene=sigGenes[,1],
                      organism="cel",
                      keyType="kegg",
                      minGSSize = 2, #default is 10
                      qvalueCutoff = 0.1)
head(keg_res)
```
### Visualisation of KEGG pathways with pathview
We can use the `pathview` function for visualising individual kegg pathways.
The `pathview` function requires a numeric vector with gene IDs as names. It
looks like the IDs it accepts in this case are Entrez ids?
The colours in the following plots indicate low (green) or high (red) beta
values for any transcript included
in the analysis (can be significant or  not).
*Note: I used here all transcripts, not just DE ones; however the log fold
change could be meaningless for insignificant genes, so perhaps a better
approach is to select significant transcripts first and then order them by
beta values*
*Note: unlikely to be that useful here as KEGG enrichment returned no
results; but equivalent with reactome or something else might be useful*
*Analysis adapted from Diego's script*
```{r pathview, message=FALSE, fig.align="center"}
#pathview can be used to show which genes are significantly differentially
expressed or log fold changes
#can be added to the vector provided to give a range of colours on the plot
#sigGenes <- sleuth_significant_txn %>% dplyr::distinct(ext_gene,
.keep_all=TRUE) %>% dplyr::select(ext_gene, b)
#keggGenes <- sigGenes$b
#names(keggGenes) <- paste("CELE", sigGenes[,1], sep="_")
#t <- sleuth_table_txn %>% filter(!is.na(ext_gene) & !is.na(qval)) %>%
dplyr::distinct(entrezgene_id, .keep_all=TRUE) %>%
dplyr::select(entrezgene_id, b)
#keggGenes <- t$b
#names(keggGenes) <- t$entrezgene_id
t <- sleuth_table_txn %>% filter(!is.na(ext_gene) & !is.na(qval)) %>%
dplyr::distinct(ext_gene, .keep_all=TRUE) %>% dplyr::select(ext_gene, b)
keggGenes <- t$b
names(keggGenes) <- paste("CELE", t$ext_gene, sep="_")
#In the images below, I use a permissive 0.5 value for beta otherwise many
#genes would come out as grey
#NOTE: these can be misleading for various reasons: a) the differences
#may not be significant (I'm not using the significant only genes)
~~~~

```
#b) I've only kept one isoform from each gene so maybe one isoform goes up
but
#others go down...
# Output image for ribosome biogenesis KEGG pathway: "cel03008"
pv.out <- pathview(gene.data = keggGenes,
         pathway.id = "cel03008",
         species = "cel",
         gene.idtype = "kegg",  #default is entrez ids
         #gene.idtype = "entrez",
         #limit = list(gene = c(round(min(keggGenes)),
          #                      round(max(keggGenes))),
          #            cpd = 1),
         limit = list(gene = c(-0.5,
                               0.5),
                      cpd = 1),
         plot.col.key = TRUE,
         kegg.native=T,
         kegg.dir = "./results/")
# Output image for spliceosome KEGG pathway: "cel03040"
pv.out <- pathview(gene.data = keggGenes,
         pathway.id = "cel03040",
         species = "cel",
         gene.idtype = "kegg",  #default is entrez ids
         #gene.idtype = "entrez",
         #limit = list(gene = c(round(min(keggGenes)),
          #                      round(max(keggGenes))),
          #            cpd = 1),
         limit = list(gene = c(-0.5,
                               0.5),
                      cpd = 1),
         plot.col.key = TRUE,
         kegg.native=T,
         kegg.dir = "./results/")
# Output image for fatty acid elongation in mitochondria KEGG pathway:
"cel00062"
pv.out <- pathview(gene.data = keggGenes,
         pathway.id = "cel00062",
         species = "cel",
         gene.idtype = "kegg",  #default is entrez ids
         #gene.idtype = "entrez",
         #limit = list(gene = c(round(min(keggGenes)),
          #                      round(max(keggGenes))),
          #            cpd = 1),
         limit = list(gene = c(-0.5,
                               0.5),
                      cpd = 1),
         plot.col.key = TRUE,
         kegg.native=T,
         kegg.dir = "./results/")
# Output image for  TCA KEGG pathway: "cel00020"
pv.out <- pathview(gene.data = keggGenes,
         pathway.id = "cel00020",
         species = "cel",
         #gene.idtype = "entrez",
         gene.idtype = "kegg",  #default is entrez ids
         #limit = list(gene = c(round(min(keggGenes)),
          #                     round(max(keggGenes))),
           #           cpd = 1),
         limit = list(gene = c(-0.5,
                              0.5),
                     cpd = 1),
         kegg.native=T,
         kegg.dir = "./results/")
# Output image for  lysosome KEGG pathway: "cel04142"
pv.out <- pathview(gene.data = keggGenes,
         pathway.id = "cel04142",
         species = "cel",
         #gene.idtype = "entrez",
         gene.idtype = "kegg",  #default is entrez ids
         #limit = list(gene = c(round(min(keggGenes)),
          #                     round(max(keggGenes))),
           #           cpd = 1),
         limit = list(gene = c(-0.5,
                              0.5),
                     cpd = 1),
         both.dirs=list(gene=TRUE, cpd=FALSE),
         kegg.native=T,
         kegg.dir = "./results/")
#tidy up (pathview prints out the annotated pathway pics in the current
directory
#but puts the original ones in the kegg.dir)
pathview_dir <- "results/pathview_kegg"
ifelse(!dir.exists(file.path("./", pathview_dir)), dir.create(file.path("./",
pathview_dir)), FALSE)
files_to_move<- list.files("./", pattern="*.pathview.png")
file.rename(files_to_move, paste(pathview_dir, files_to_move, sep="/"))
pathview_dir <- "results/pathview_kegg"
files_to_move <- list.files("./results", pattern="cel*[png|xml]",
full.names=F)
file.rename(paste("results/", files_to_move,sep="/"), paste(pathview_dir,
files_to_move, sep="/"))
```
```
```

~~~
![Alt text](results/pathview_kegg/cel00020.pathview.png)
![Alt text](results/pathview_kegg/cel00062.pathview.png)
![Alt text](results/pathview_kegg/cel03008.pathview.png)
![Alt text](results/pathview_kegg/cel03040.pathview.png)
![Alt text](results/pathview_kegg/cel04142.pathview.png)
### Functional enrichment analysis with gProfiler (using transcript-level
sleuth results)
```{r gProfiler}
#code follows: https://cran.r-
project.org/web/packages/gprofiler2/vignettes/gprofiler2.html#gene-list-
functional-enrichment-analysis-with-gost
#Analysis with gProfiler can also be done online at:
#install.packages("gprofiler2")
#install.packages("treemap")
library(treemap)
library(gprofiler2)
#use genes rather than transcripts
df <- sleuth_significant_txn %>% distinct(ens_gene, .keep_all=TRUE)
# check it worked
dim(df)
length(unique(sleuth_significant_txn$ens_gene)) == dim(df)[1]
gostres <- gost(query = df$ext_gene,
                                  organism = "celegans",
                                  ordered_query = F,
                                  exclude_iea = T,   #exclude computational
GO annotations
                                  significant=TRUE,
                                  evcodes = TRUE, #evidence codes for the
results
                                  correction_method = "fdr", #tailor-made
algorithm provided by gProfiler
                                  custom_bg = dplyr::filter(sleuth_table_txn,
!is.na(qval))$target_id)
head(gostres$result)
#interactive plot highlighting significant terms
gostplot(gostres,
         capped = TRUE, # all pvalues less than 10e-16 will appear above this
line together
         interactive = TRUE)
#create a table with results
publish_gosttable(gostres, highlight_terms = gostres$result[c(1:10),],
                   use_colors = TRUE,
                   show_columns = c("source", "term_name", "term_size",
"intersection_size"),
                   filename = NULL)
gostres$result[,c("term_id", "p_value")]
~~~

```
#save results for GO terms only so they can be uploaded to REVIGO
t<- gostres$result[,c("term_id", "p_value")]
write.table(t[grep("GO", t$term_id),], "results/gProfiler_GO.txt",
quote=FALSE, row.names = FALSE, col.names = FALSE)
#Then copy and pase the content of gProfiler_GO.txt into REVIGO at:
http://revigo.irb.hr/revigo.jsp
#and download the R script for plotting from that website. REVIGO reduces the
GO terms by removng redundancy and making it easier to look at the results.
#Can download the treemap script or the other R scripts from the individual
tabs
#NOTE: The treemap R script needs a small modification to avoid warnings
(change tmPlot() to treemap())
```
Visualisation of gProfiler GO analysis with REVIGO plots:
![Alt text](results/revigo_treemap.pdf)
![Alt text](results/revigo-plot.pdf)
### Over-representation analysis with Panther
This analysis was carried out online (but when I have time I'll implement it
with the API)
using the up- and down-regulated gene lists I created earlier using the
significant transcript
table from sleuth. As background, we used all genes from sleuth_table_wt with
no NAs (11337
genes), i.e.
`dplyr::filter(sleuth_table_wt, !is.na(qval))$target_id`
The results are in both json and csv formats but here we'll use the csv as
the json
files contain a lot more information than we need to make the summary figures
below.
Information about individual genes mapping to the gene sets though is held in
the .json
files only.
```{r panther over-representation analysis}
#we will use the GOslim annotations as the others contain a lot of redundant
GO terms.
# ##########################################
# first look at ***UPREGULATED*** gene set
# ##########################################
panther.bpslim <-
read.csv(file="results/panther/panther_GOslim_BP_sleugh_sig_txn_up_vs_all.txt
", skip = 11, header=T, sep="\t")
dim(panther.bpslim)
#filter out the very generic GO terms (put a filter at 200 genes for the
background) and the very small term (filter at 10 genes for the background)
```

```
panther.bpslim <-
panther.bpslim[!((panther.bpslim$bg_genes_all.txt...REFLIST..10765. > 200) |
(panther.bpslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.bpslim)
#we limit the selection of terms to those that are over-represented (under-
represented ones will probably appear in the downregulated set of genes
anyway)
panther.bpslim <- panther.bpslim %>%
filter(sleuth_significant_q005_up.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Biological.Process,
sleuth_significant_q005_up.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Biological.Process, log10FDR=
sleuth_significant_q005_up.txt..FDR.)
panther.bpslim$term <- "BP"
panther.ccslim <-
read.csv(file="results/panther/panther_GOslim_CC_sleugh_sig_txn_up_vs_all.txt
", skip = 11, header=T, sep="\t")
#filter out the very generic GO terms (put a filter at 500 genes for the
background) and the very small term (filter at 10 genes for the background)
#note the use of 500 instead of 200 genes here as upper max as cellular
components generally contain a lot more genes
panther.ccslim <-
panther.ccslim[!((panther.ccslim$bg_genes_all.txt...REFLIST..10765. > 500) |
(panther.ccslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.ccslim)
panther.ccslim <- panther.ccslim %>%
filter(sleuth_significant_q005_up.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Cellular.Component,
sleuth_significant_q005_up.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Cellular.Component, log10FDR=
sleuth_significant_q005_up.txt..FDR.)
panther.ccslim$term <- "CC"
panther.mfslim <-
read.csv(file="results/panther/panther_GOslim_MF_sleugh_sig_txn_up_vs_all.txt
", skip = 11, header=T, sep="\t")
#filter out the very generic GO terms (put a filter at 200 genes for the
background) and the very small term (filter at 10 genes for the background)
panther.mfslim <-
panther.mfslim[!((panther.mfslim$bg_genes_all.txt...REFLIST..10765. > 200) |
(panther.mfslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.mfslim)
panther.mfslim <- panther.mfslim %>%
filter(sleuth_significant_q005_up.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Molecular.Function,
sleuth_significant_q005_up.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Molecular.Function, log10FDR=
sleuth_significant_q005_up.txt..FDR.)
panther.mfslim$term <- "MF"
bars.up <- rbind(panther.bpslim, panther.ccslim, panther.mfslim)
bars.up$log10FDR <- -log10(bars.up$log10FDR)
bars.up$up_or_down <- "up"
dim(bars.up)
p.up <-ggplot(bars.up, aes(x=GO, y=log10FDR, fill=term))
p.up <- p.up +geom_bar(stat = "identity") + coord_flip() + ylab("-
log10(FDR)") + xlab("GO term") + ggtitle("Over-represented GO terms in
upregulated gene set (fdr < 0.05)")
# ########################################
#now look at ***DOWNREGULATED*** gene set
# ########################################
panther.bpslim <-
read.csv(file="results/panther/panther_GOslim_BP_sleugh_sig_txn_down_vs_all.t
xt", skip = 11, header=T, sep="\t")
dim(panther.bpslim)
#filter
panther.bpslim <-
panther.bpslim[!((panther.bpslim$bg_genes_all.txt...REFLIST..10765. >200) |
(panther.bpslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.bpslim)
#we limit the selection of terms to those that are over-represented (under-
represented ones will probably appear in the downregulated set of genes
anyway)
panther.bpslim <- panther.bpslim %>%
filter(sleuth_significant_q005_down.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Biological.Process,
sleuth_significant_q005_down.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Biological.Process, log10FDR=
sleuth_significant_q005_down.txt..FDR.)
panther.bpslim$term <- "BP"
panther.ccslim <-
read.csv(file="results/panther/panther_GOslim_CC_sleugh_sig_txn_down_vs_all.t
xt", skip = 11, header=T, sep="\t")
dim(panther.ccslim)
#filter
panther.ccslim <-
panther.ccslim[!((panther.ccslim$bg_genes_all.txt...REFLIST..10765. >500) |
(panther.ccslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.ccslim)
panther.ccslim <- panther.ccslim %>%
filter(sleuth_significant_q005_down.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Cellular.Component,
sleuth_significant_q005_down.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Cellular.Component, log10FDR=
sleuth_significant_q005_down.txt..FDR.)
panther.ccslim$term <- "CC"
panther.mfslim <-
read.csv(file="results/panther/panther_GOslim_MF_sleugh_sig_txn_down_vs_all.t
xt", skip = 11, header=T, sep="\t")
dim(panther.mfslim)
#filter
panther.mfslim <-
panther.mfslim[!((panther.mfslim$bg_genes_all.txt...REFLIST..10765. > 200) |
(panther.mfslim$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.mfslim)
panther.mfslim <- panther.mfslim %>%
filter(sleuth_significant_q005_down.txt..over.under. != "-") %>%
dplyr::select(PANTHER.GO.Slim.Molecular.Function,
sleuth_significant_q005_down.txt..FDR.) %>% dplyr::rename(GO=
PANTHER.GO.Slim.Molecular.Function, log10FDR=
sleuth_significant_q005_down.txt..FDR.)
panther.mfslim$term <- "MF"
bars.down <- rbind(panther.bpslim, panther.ccslim, panther.mfslim)
bars.down$log10FDR <- -log10(bars.down$log10FDR)
dim(bars.down)
#there are too many GO terms to display on the plot; will filter on -
log10(FDR)
bars.down <- bars.down %>% filter(log10FDR > 2)
bars.down$up_or_down <- "down" #only needed if we put the barplots together
but at this moment not used
p.down <-ggplot(bars.down, aes(x=GO, y=log10FDR, fill=term))
p.down <- p.down +geom_bar(stat = "identity") + coord_flip() + ylab("-
log10(FDR)") + xlab("GO term") + ggtitle("Over-represented GO terms in
downregulated gene set (filter: fdr < 10e-2)")
#FIGURE fig_panther_up_down
pdf(file="figures/fig_panther_up_down.pdf", width=15, height=13)
cowplot::plot_grid(p.down, p.up, ncol=1, labels=LETTERS[1:2])
dev.off()
#REACTOME
#this is interesting to include as it has more informative and descriptive
terms
panther.reactome <-
read.csv(file="results/panther/panther_reactome_sleugh_sig_txn_down_vs_all.tx
t", skip = 11, header=T, sep="\t")
dim(panther.reactome)
#filter
panther.reactome <-
panther.reactome[!((panther.reactome$bg_genes_all.txt...REFLIST..10765. >
200) | (panther.reactome$bg_genes_all.txt...REFLIST..10765. < 10)),]
```

```
dim(panther.reactome)
panther.reactome <- panther.reactome %>%
filter(sleuth_significant_q005_down.txt..over.under. != "-") %>%
dplyr::select(Reactome.pathways, sleuth_significant_q005_down.txt..FDR.) %>%
dplyr::rename(GO= Reactome.pathways, log10FDR=
sleuth_significant_q005_down.txt..FDR.)
panther.reactome$term <- "reactome"
bars.down <- panther.reactome
bars.down$log10FDR <- -log10(bars.down$log10FDR)
dim(bars.down)
#there are too many GO terms to display on the plot; will filter on -
log10(FDR)
bars.down <- bars.down %>% filter(log10FDR > 2)
bars.down$up_or_down <- "down" #only needed if we put the barplots together
but at this moment not used
#FIGURE fig_panther_reactome
p.down <-ggplot(bars.down, aes(x=GO, y=log10FDR, fill=term))
p.down <- p.down +geom_bar(stat = "identity") + coord_flip() + ylab("-
log10(FDR)") + xlab("Reactome term") + ggtitle("Over-represented reactome
terms in downregulated genes; fdr < 10e-2")
p.down
panther.reactome <-
read.csv(file="results/panther/panther_reactome_sleugh_sig_txn_up_vs_all.txt"
, skip = 11, header=T, sep="\t")
dim(panther.reactome)
#filter
panther.reactome <-
panther.reactome[!((panther.reactome$bg_genes_all.txt...REFLIST..10765. >
200) | (panther.reactome$bg_genes_all.txt...REFLIST..10765. < 10)),]
dim(panther.reactome)
panther.reactome <- panther.reactome %>%
filter(sleuth_significant_q005_up.txt..over.under. != "-") %>%
dplyr::select(Reactome.pathways, sleuth_significant_q005_up.txt..FDR.) %>%
dplyr::rename(GO= Reactome.pathways, log10FDR=
sleuth_significant_q005_up.txt..FDR.)
panther.reactome$term <- "reactome"
bars.up <- panther.reactome
bars.up$log10FDR <- -log10(bars.up$log10FDR)
dim(bars.up)
#if there are too many GO terms to display on the plot; will filter on -
log10(FDR)
#bars.up <- bars.up %>% filter(log10FDR > 2)
bars.up$up_or_down <- "up" #only needed if we put the barplots together but
at this moment not used
p.up <-ggplot(bars.up, aes(x=GO, y=log10FDR, fill=term))
```

~~~
p.up <- p.up +geom_bar(stat = "identity") + coord_flip() + ylab("-
log10(FDR)") + xlab("Reactome term") + ggtitle("Over-represented reactome
terms in upregulated genes")
p.up
#results for upregulated genes do not add anything to what we found before
```
### Gene expression among genes in Wormbook pathways
Here, we explore selected gene sets, as defined in the Wormbook, that
correspond to well-established biological networks in C. elegans.
Much of this code was written by Diego Chillon Pino.
All genes analysed in this section were retrieved from
[WormBook](http://www.wormbook.org/). For every pathway a data frame object
is created. The user has to manually copy and paste the appropiate table from
[WormBook](http://www.wormbook.org/) in a .txt file before running the code.
Exceptions are pointed out. Note that the code slightly differ among pathways
due to inconsistency of the tables' format.
This script assumes the existence of a subdirectory within the working
directory called `wormbook_pathways`, which contains the copied and pasted
tables.
```{r expression_within_wormbook_pathways}
# Function to print results
print_hits <- function(x){
  cat(
    "Number of downregulated transcripts: ",
    length(dplyr::filter(sleuth_table_txn, ext_gene %in% x &
                           b < 0)[, c(2, 4, 7, 8)]$target_id),
    " (",
    length(dplyr::filter(sleuth_significant_txn, ext_gene %in% x &
                           b < 0)[, c(2, 4, 7, 8)]$target_id),
    " significant)\n",
    sep = ""
  )

  cat(
    "Number of upregulated transcripts: ",
    length(dplyr::filter(sleuth_table_txn, ext_gene %in% x &
                           b > 0)[, c(2, 4, 7, 8)]$target_id),
    " (",
    length(dplyr::filter(sleuth_significant_txn, ext_gene %in% x &
                           b > 0)[, c(2, 4, 7, 8)]$target_id),
    " significant)",
    sep = ""
  )
}
### NEUROTRANSMITTERS
#### Acetylcholine
~~~

~~~
list.acetylcholine <- read.delim(
  "./wormbook_pathways/acetylcholine.txt",
  header = FALSE,
  stringsAsFactors = FALSE,
  na.strings = ""
)
# Remove leading and trailing spaces in all elements of the data frame
list.acetylcholine[] <- lapply(list.acetylcholine, trimws)
# Remove headers from the original table and overwrite
list.acetylcholine <- list.acetylcholine$V1[
  which(grepl("[[:upper:]]+[[:punct:]][[:digit:]]*", list.acetylcholine$V1)
== TRUE)
  ] %>%
  tolower()
# add the genes mentioned in text but not appearing in the table (not a full
list!)
list.acetylcholine <- c(list.acetylcholine, c("cha-1", "unc-17", "cho-1",
"snf-6", "ace-1", "ace-2", "ace-3", "ric-3", "lev-10", "eat-18", "cam-1"))
# Print results (NOTE: only first transcript counts if many are in the gene)
print_hits(list.acetylcholine)
```
## Analysis of isoform switching using isoformAnalyzer
We now use the isoformAnalyzer package to examine the changes at transcript
level that are picked up by kallisto's transcript isoform quantification
pipeline.
```{r isoformAnalyzer}
#This is following the isoformAnalyzeR documentation
#BiocManager::install("IsoformSwitchAnalyzeR")
library(IsoformSwitchAnalyzeR)
subDir="isoformAnalyzer"
mainDir="results/isoformAnalyzer"
if (!file.exists(mainDir)){
    dir.create(file.path("./results", subDir))
}
#Import quantifications from kallisto
kallistoQuant <- importIsoformExpression(parentDir =
"kallisto_with_ncRNA_norRNA/")
#check it worked
kallistoQuant$abundance[1,]
kallistoQuant$counts[1,]
#create the design table
myDesign <- data.frame(
    sampleID = colnames(kallistoQuant$abundance)[-1],
    condition = c("control", "control", "control", "control",
                  "neurexin", "neurexin", "neurexin", "neurexin")
)
~~~

```
myDesign
#now build the list of isoforms
#isoformNtFasta is only available in the latest release - needed to update R
to 3.6
#isoSwitchList is equivalent of "aSwitchList" in documentation
isoSwitchList <- importRdata(
    isoformCountMatrix   = kallistoQuant$counts,
    isoformRepExpression = kallistoQuant$abundance,
    designMatrix         = myDesign,
    isoformExonAnnoation =
"genomes/Caenorhabditis_elegans.WBcel235.99.gtf.gz",
    isoformNtFasta       =
"genomes/Caenorhabditis_elegans.WBcel235.cdna_and_ncrna_norRNA.fa.gz",
    showProgress = FALSE
)
#examples of the data held in isoSwitchList to show import worked
head(isoSwitchList$isoformFeatures,2)
head(isoSwitchList$exons,2)
head(isoSwitchList$ntSequence,2)
#filtering to remove genes with a single isoform (removes 19941 or 62.5% of
all transcripts) and remove isoforms/genes with low expression (removes
additionally 6108 transcripts so in the end 81.6% of transcripts are removed;
5867 isoforms left )
isoSwitchListFiltered <- preFilter(
    switchAnalyzeRlist = isoSwitchList,
    geneExpressionCutoff = 5,
    isoformExpressionCutoff = 2,
    removeSingleIsoformGenes = TRUE
)
#testing for isoform switches using DEXSeq
isoSwitchListAnalyzed <- isoformSwitchTestDEXSeq(
    switchAnalyzeRlist = isoSwitchListFiltered,
    reduceToSwitchingGenes=TRUE
)
#extract a summary - 32 isoforms found to switch (corresponding to 19 genes)
extractSwitchSummary(isoSwitchListAnalyzed)
```

## Session information

```{r session_info}
date()
sessionInfo()
#save.image(file="neurexin_analysis.Rdata")
```