Creating Differential Transcript Expression Results with DESeq2 V.2

2

David A Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

2

Apr 11, 2022

1

dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

David Eccles
Malaghan Institute of Medical Research (NZ)

Differential expression analysis of transcript count tables using DESeq2

DOI

dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

David A Eccles 2022. Creating Differential Transcript Expression Results with DESeq2. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2
David Eccles

Nanopore Data Analysis

Nanopore Data Analysis

DESeq2, nanopore, transcript, DE, expression

protocol ,

Apr 08, 2022

Apr 11, 2022

60467

Part of collection
Nanopore Data Analysis
Nanopore Data Analysis

Note: this is a demonstrative experimental protocol for a specific differential expression analysis; variables, file names and some other code will need to be changed for your own circumstances

You should have gene-annotated transcript count tables for multiple sequencing libraries, renamed to match the form of "<library_identifier>_wide_transcript_counts_<mapper>.csv". See here for how to create these tables from nanopore cDNA transcripts.

You should also have a sample metadata file that matches library/barcode pairs to experimental conditions, ***with at least "SampleID" and "Label" fields***. The SampleID field should be of the form "<library_identifier>.BCXX"; this will be replaced by the "Label" field when results are output to a file.

Collating count data

1 Combine the transcript count data into a single data structure. To reduce confusion when this protocol is run multiple times, we declare an *analysisDate* variable to be used for output file names:

```
countDate <- format(Sys.Date(), "%Y-%b-%d");
```

We also load libraries that will be used in the protocol:

```
## steps 1-2
library(tidyverse);    # data table manipulation
## steps 3-8
library(DESeq2);   # differential expression analysis
library(ashr);     # for log fold change shrinking
## step 9
options(java.parameters = "-Xmx10G"); # may be needed to avoid
Excel errors
library(xlsx);                        # writing to an Excel file
```

1.1 Collect a list of the count data files:

```
data.files <- list.files(pattern =
"_wide_transcript_counts.*\\.csv$");
names(data.files) <-
sub("_wide_transcript_counts.*\\.csv$","",data.files);
```

CHECK - print out the file names to make sure they're correct:

Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
data.files

## example output
#                                    CGAug18_004
#"CGAug18_004_wide_transcript_counts_LAST.csv"
#                                       CGDec17
#    "CGDec17_wide_transcript_counts_LAST.csv"
#                                    CGJan19_006
#"CGJan19_006_wide_transcript_counts_LAST.csv"
#                                    CGMar19_009
#"CGMar19_009_wide_transcript_counts_LAST.csv"
#                                    CGNov18_005
#"CGNov18_005_wide_transcript_counts_LAST.csv"
```

1.2 Set up the skeleton structures for creating the combined table. This is created in two parts:

1. A gene lookup table, containing gene metadata
2. A count table, containing transcript counts

```
geneLookup.tbl <- NULL;
counts.raw.tbl <- tibble(tdir=character());
```

1.3 The skeleton structures are then populated with the data from individual library files:

```
for(dfi in seq_along(data.files)){
  data.files[dfi] %>%
    read_csv %>%
    mutate(tdir = paste0(transcript, "_", dir)) ->
      counts.sub.tbl;
  ## append columns without barcode names to gene table
  geneLookup.tbl %>%
      rbind(geneLookup.tbl, select(counts.sub.tbl,
                  -contains("BC"),
                  -contains("RB"))) -> geneLookup.tbl;
  ## append columns *with* barcode names to count table
  counts.sub.tbl %>%
      select("tdir", contains("BC"), contains("RB")) ->
counts.sub.tbl;
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
  ## add file label to barcode name column
  bcCols <- grep("(BC|RB)", colnames(counts.sub.tbl));
  colnames(counts.sub.tbl)[bcCols] %<>%
    paste0(names(data.files)[dfi], ".", .);
  counts.raw.tbl %>%
      full_join(counts.sub.tbl, by="tdir") ->
counts.raw.tbl;
}
## Remove duplicates from the gene table
geneLookup.tbl %>% unique -> geneLookup.tbl;
```

CHECK - make sure that the column headings of the aggregated count table match the expected names:

```
colnames(counts.raw.tbl)

## Example output
# [1] "tdir"             "CGAug18_004.BC04"
"CGAug18_004.BC05" "CGAug18_004.BC06"
# [5] "CGDec17.BC06"     "CGDec17.BC07"
"CGJan19_006.BC03" "CGJan19_006.BC04"
# [9] "CGJan19_006.BC05" "CGJan19_006.BC06"
"CGJan19_006.BC07" "CGJan19_006.BC08"
#[13] "CGMar19_009.BC07" "CGMar19_009.BC08"
"CGMar19_009.BC09" "CGMar19_009.BC10"
#[17] "CGNov18_005.BC01" "CGNov18_005.BC02"
"CGNov18_005.BC03" "CGNov18_005.BC04"
#[21] "CGNov18_005.BC05" "CGNov18_005.BC06"
```

2 Do some cleaning / reordering of the data, then create an intermediate aggregate count table

2.1 The sample metadata file (see ***Guidelines & Warnings***) is read in, mostly as factors; the sample ID is converted to a character vector:

```
read.csv("metadata.csv") %>%
    mutate(SampleID = as.character(SampleID)) ->
    meta.df;
```

Metadata rows are subset and to match IDs present in the count table, and underscores are replaced with dots in the sample IDs:

```
## Filter metadata to only include the desired samples
meta.df %>%
    filter(SampleID %in% colnames(counts.raw.tbl)) %>%
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
    ## Sort by condition
    arrange(Condition) -> meta.df
## Replace '_' with '.' in column and metadata names
meta.df$SampleID <- gsub("_", ".", meta.df$SampleID);
colnames(counts.raw.tbl) <- gsub("_", ".",
colnames(counts.raw.tbl));
```

2.2 Count table rows are subset and re-ordered to match the order of the metadata table, and counts are combined with gene annotation:

```
counts.raw.tbl %>%
    ## only select forward-oriented transcripts
    filter(endsWith(tdir, "_fwd")) %>%
    ## convert to integer
    mutate_at(grep("\\.BC", colnames(.)), as.integer)
%>%
    ## convert NA values to 0 counts
    mutate_at(grep("\\.BC", colnames(.)), function(x)
{coalesce(x, 0L)}) %>%
    distinct() %>%
    ## join to transcript metadata
    left_join(geneLookup.tbl) %>%
    ## reorder to match metadata table
    select(transcript, Chr, Strand, Start, End,
           Description, Gene, dir, tdir,
           match(meta.df$SampleID[meta.df$SampleID %in%
colnames(.)],
                 colnames(.))) -> filt.count.tbl;
```

2.3 Filters are applied to make sure that only reliably-expressed genes are included:

```
minCounts <- 3; # minimum number of counts per
transcript (across all samples)
minSamples <- 3; # minimum number of samples with
expressed transcript
bcCols <- grep("\\.BC",colnames(filt.count.tbl));
## Exclude genes with fewer than minimum total counts
filt.count.tbl <-
    filt.count.tbl[rowSums(filt.count.tbl[,bcCols]) >=
minCounts,];
## Exclude genes with fewer than minimum totals samples
filt.count.tbl  <-
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
    filt.count.tbl[apply(filt.count.tbl[,bcCols],1,
                         function(x){sum(x >= minCounts)
> minSamples}),]
```

2.4 The combined table is sorted based on total expression across all samples and output to an intermediate file, using the analysis date as a file name:

```
filt.count.tbl %>%
    arrange(-rowSums(filt.count.tbl[,bcCols])) %>%
        write_csv(sprintf("raw_counts_%s.csv",
countDate));
```

Differential Expression

3 ***Note: What follows from here will be more case-specific. The steps outlined here represent one method of carrying out a differential expression analysis with a relatively simple statistical model. For further ideas for how to carry out a differential expression analysis, see the DESeq2 Workflow.***

Set up variables to change output file names and behaviour:

Note: the countDate *is not "today" because different explorations of differential expression could be done on the same count data.*

```
countDate <- "2022-Apr-11"; # date of count aggregation
l2FCShrink <- TRUE; # whether the Log2FC values should be shrunk
analysisDate <- format(Sys.Date(), "%Y-%b-%d"); # date of DESeq
analysis
resultSource <- "GRCm39_ATP5KO"; # descriptive label for results
excluded.factors <- "Treatment"; # factors to exclude from
statistical model
```

Read in the intermediate aggregated count file and the metadata file:

```
sprintf("raw_counts_%s.csv", countDate) %>%
    read_csv() -> count.tbl;

read.csv("metadata.csv") %>%
    mutate(SampleID = gsub("_", ".",as.character(SampleID))) %>%
    ## Make sure metadata information only includes samples in the
count table
    filter(SampleID %in% colnames(count.tbl)) -> meta.df;
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

4 Carry out metadata filtering (i.e. sample exclusion) and count filtering (e.g. gene / sample QC)

4.1 Filter the metadata table to only include the desired samples:

Note: this step will be situation specific

```
meta.df %>%
    ## Only keep 4T1 strain data
    filter(Strain == "4T1") %>%
    ## Sort by cell line, then replicate
    arrange(Line, Replicate) -> meta.df
```

4.2 Filter the metadata file to match the count table (i.e. removing any samples filtered out in previous steps), and exclude any columns that have single values. Count tables are finally arranged as in the metadata table:

```
for(x in colnames(meta.df)){
    if(is.factor(meta.df[[x]])){
        meta.df[[x]] %>% factor -> meta.df[[x]];
    }
}

count.tbl %>%
   select(transcript, Chr, Strand, Start, End,
           Description, Gene, dir, tdir,
           match(meta.df$SampleID[meta.df$SampleID %in%
colnames(.)],
                 colnames(.))) -> filt.count.tbl;
```

CHECK - make sure the SampleIDs in the column names match the exact order of the metadata table:

```
colnames(filt.count.tbl)[grepl("(BC|RB)",
colnames(filt.count.tbl)]] == meta.df$SampleID

## Example output
# [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE TRUE
#[16] TRUE TRUE
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

5 Create DESeq2 data structure, and run a differential expression analysis

5.1 Identify factors for the statistical model from the metadata file:

```
setdiff(colnames(meta.df),
        c(c("SampleID","Label","Replicate","Notes"),
excluded.factors)) ->
    factorNames;
```

5.2 Create the transcript count matrix:

```
count.mat <- as.matrix(count.tbl[grepl("(BC|RB)",
colnames(filt.count.tbl))]);
rownames(count.mat) <- count.tbl$transcript;
```

5.3 Convert to DESeq2 structure and run DESeq2:

```
DESeqDataSetFromMatrix(count.mat, meta.df,
        as.formula(paste0("~ ", paste(factorNames,
collapse=" + ")))) %>%
    DESeq ->
    dds;
```

CHECK - make sure dispersion and result names look reasonable:

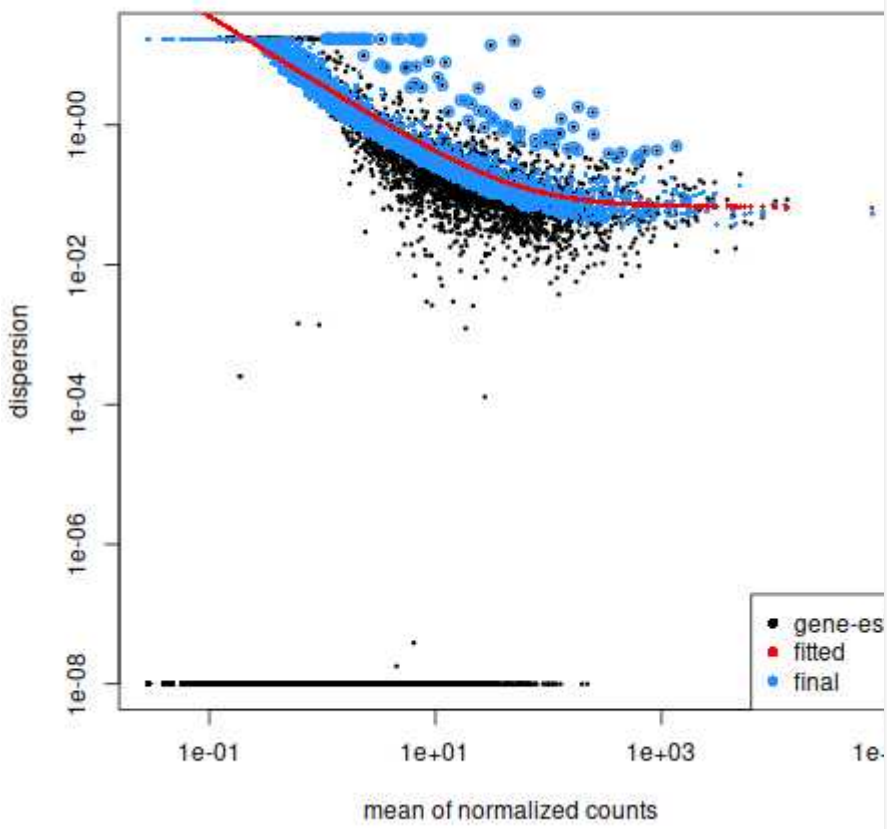
Note: The result names will not indicate every possible comparison; just a subset from which all comparisons can be derived from.

```
png("dispEsts.png");
plotDispEsts(dds);
dev.off();
resultsNames(dds);

## Example output
# [1] "Intercept"
"Experiment_CG005_vs_CG004"
# [3] "Experiment_CG006_vs_CG004"
"Experiment_CG009_vs_CG004"
# [5] "Experiment_CGDec17_vs_CG004" "Line_p0D25_vs_p0"
# [7] "Line_p0SC_vs_p0"             "Line_p0SCL_vs_p0"
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
# [9] "Line_WT_vs_p0"
```



Dispersion plot for nanopore transcript data. Ideally there should be a smooth curv
higher dispersion for low-count genes, and lower dispersion for high-count genes.

Result Collation

6 Prepare results table skeletons for DESeq2 results

6.1 Collect up comparisons to make:

```
resultList <- NULL;
for(fi in factorNames){
    vn <- unique(meta.df[[fi]]);
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
    vn <- vn[order(-xtfrm(vn))];
    if(length(vn) == 1){
        next;
    }
    for(fai in seq(1, length(vn)-1)){
        for(fbi in seq(fai+1, length(vn))){
            cat(sprintf("%s: %s vs %s\n", fi, vn[fai],
vn[fbi]));
            resultList <- c(resultList,
                            list(c(fi,
as.character(vn[fai]), as.character(vn[fbi]))));
        }
    }
}
```

6.2 Remove any unwanted comparisons for results output:

```
## Only keep comparisons that we want
resultList <-
    resultList[sapply(resultList, function(x){x[1] !=
"Batch"})];
```

6.3 Create variance-stabilised [log2] count matrix from DESeq2 structure:

```
dds.counts <- assay(vst(dds, blind=FALSE));
```

Rescale VST values to have the same 99th percentile, but a minimum value of zero. This makes the scaled counts resemble more closely the actual read counts:

```
dds.quantile99 <- quantile(dds.counts[dds.counts >
min(dds.counts)], 0.99);
((dds.counts - min(dds.counts)) /
 (dds.quantile99 - min(dds.counts))) * (dds.quantile99)
->
    dds.counts;
```

Replace column names in the VST matrix with labels from the metadata:

Note: the substution removes any initial whitespace from the label

```
colnames(dds.counts)[match(meta.df$SampleID,
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
                                colnames(dds.counts))] <-
    paste0("adj.",sub("^ +", "",
as.character(meta.df$Label)));
```

6.4 Generate base count table:

```
dds.withCounts.tbl <- count.tbl
## replace column names with metadata labels
colnames(dds.withCounts.tbl)[match(meta.df$SampleID,

colnames(dds.withCounts.tbl))] <-
    paste0("raw.",sub("^ +", "",
as.character(meta.df$Label)));
```

Tack on VST matrix:

```
dds.withCounts.tbl[,colnames(dds.counts)] <-
round(dds.counts,2);
```

Pre-populate wth minimum p-value column:

```
dds.withCounts.tbl$min.p.val <- 0;
```

7 Fetch DESeq2 results for each comparison from the DESeq2 data structure and add to the base table:

```
for(rn in resultList){
    print(rn);
    results.df <-
        if(l2FCShrink){
            as.data.frame(lfcShrink(dds, contrast=rn, type =
"ashr"));
        } else {
           as.data.frame(results(dds, contrast=rn));
        }
    results.df$log2FoldChange <- round(results.df$log2FoldChange,
2);
    results.df$pvalue <- signif(results.df$pvalue, 3);
    results.df$padj <- signif(results.df$padj, 3);
    rn.label <- paste(rn, collapse="-");
    results.tbl <- as.tbl(results.df[, c("log2FoldChange", "lfcSE",
"pvalue", "padj")]);
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
    colnames(results.tbl) <- paste0(c("L2FC.", "lfcSE.", "pval.",
"padj."), rn.label);
    results.tbl$transcript <- rownames(results.df);
    dds.withCounts.tbl <-
        left_join(dds.withCounts.tbl, results.tbl,
by="transcript");
}
```

8 Write results out to a CSV file:

```
dds.withCounts.tbl$min.p.val <-

apply(dds.withCounts.tbl[,grep("^padj\\.",colnames(dds.withCounts.t
bl))],
          1, min, na.rm=TRUE);
write.csv(dds.withCounts.tbl, row.names=FALSE,
          gzfile(sprintf("DE_%s_VST_%s_%s.csv.gz",
                         if(l2FCShrink){"shrunk"} else {"orig"},
                         resultSource, analysisDate)));
```

Excel Worksheet Output

9 Separate results and put into an Excel file

Note: this step will be situation specific

9.1 Split out mitochondrial genes:

```
filtered.dds.tbl <- filter(dds.withCounts.tbl, Chr !=
"MT");
MT.dds.tbl <- filter(dds.withCounts.tbl, Chr == "MT");
```

9.2 Write split datasets out to the Excel file:

```
write.xlsx2(as.data.frame(filtered.dds.tbl),
            sprintf("DE_%s_VST_%s_%s.xlsx",
                    if(l2FCShrink){"shrunk"} else
{"orig"},
                    resultSource, analysisDate),
            sheetName="Genome Data", row.names=FALSE);
write.xlsx2(as.data.frame(MT.dds.tbl),
            sprintf("DE_%s_VST_%s_%s.xlsx",
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2

```
                    if(l2FCShrink){"shrunk"} else
{"orig"},
                    resultSource, analysisDate),
            sheetName="MT Data", append=TRUE,
row.names=FALSE);
```

9.3 Add worksheets for differentially-expressed pairs:

```
for(rn in resultList){
    print(rn);
    if(rn[1] == "Experiment"){
        next;
    }
    sheetName <- sprintf("%s; %s vs %s", rn[1], rn[2],
rn[3]);
    meta.df <- meta.df[order(meta.df$Line,
meta.df$Replicate),];
    cnames <- sub("^
+","",as.character(meta.df$Label[meta.df[[rn[1]]] %in%
rn[2:3]]));
    cnames <- c(paste0("raw.",cnames),
paste0("adj.",cnames));
    rn.label <- paste(rn, collapse="-");
    res.tbl <-
filtered.dds.tbl[,c(colnames(filtered.dds.tbl)
[c(6,7,8)],

cnames,paste0(c("L2FC.","pval.", "padj."), rn.label))];
    res.tbl <- res.tbl[res.tbl[[paste0("padj.",
rn.label)]] <= 0.1,];
    res.tbl <- res.tbl[order(-res.tbl[[paste0("L2FC.",
rn.label)]]),];
    write.xlsx2(as.data.frame(res.tbl),
                sprintf("DE_%s_VST_%s_%s.xlsx",
                        if(l2FCShrink){"shrunk"} else
{"orig"},
                        resultSource, analysisDate),
               sheetName=sheetName, append=TRUE,
row.names=FALSE);
}
```


Citation: David A Eccles Creating Differential Transcript Expression Results with DESeq2
https://dx.doi.org/10.17504/protocols.io.8epv51686l1b/v2