







SOFTWARE TOOL ARTICLE

RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 1; peer review: 3 approved]

Charity W. Law ^{1,2}, Monther Alhamdoosh ³, Shian Su¹, Gordon K. Smyth ^{1,4}, Matthew E. Ritchie ^{1,2,4}

¹The Walter and Eliza Hall Institute of Medical Research, Parkville, 3052, Australia

²Department of Medical Biology, The University of Melbourne, Parkville, 3010, Australia

³CSL Limited, Parkville, Victoria, 3010, Australia

⁴School of Mathematics and Statistics, The University of Melbourne, Parkville, 3010, Australia

V1 First published: 17 Jun 2016, 5:1408
<https://doi.org/10.12688/f1000research.9005.1>
 Second version: 30 Nov 2016, 5:1408
<https://doi.org/10.12688/f1000research.9005.2>
 Latest published: 28 Dec 2018, 5:1408
<https://doi.org/10.12688/f1000research.9005.3>

Abstract

The ability to easily and efficiently analyse RNA-sequencing data is a key strength of the Bioconductor project. Starting with counts summarised at the gene-level, a typical analysis involves pre-processing, exploratory data analysis, differential expression testing and pathway analysis with the results obtained informing future experiments and validation studies. In this workflow article, we analyse RNA-sequencing data from the mouse mammary gland, demonstrating use of the popular edgeR package to import, organise, filter and normalise the data, followed by the limma package with its voom method, linear modelling and empirical Bayes moderation to assess differential expression and perform gene set testing. This pipeline is further enhanced by the Glimma package which enables interactive exploration of the results so that individual samples and genes can be examined by the user. The complete analysis offered by these three packages highlights the ease with which researchers can turn the raw counts from an RNA-sequencing experiment into biological insights using Bioconductor.

Keywords

RNA sequencing , data analysis , gene expression



This article is included in the **Bioconductor** gateway.

Open Peer Review

Reviewer Status 

Invited Reviewers

1 2 3

version 3

(revision)

28 Dec 2018

version 2

(revision)

30 Nov 2016

version 1

17 Jun 2016



report



report



report

1. **James W. MacDonald** , University of Washington, Seattle, USA
2. **Maria A. Doyle**, Peter MacCallum Cancer Centre, Melbourne, Australia
3. **Jovana Maksimovic** , Murdoch Children's Research Institute, Parkville, Australia

Any reports and responses or comments on the article can be found at the end of the article.



This article is included in the **International Society for Computational Biology Community Journal gateway**.

Corresponding authors: Gordon K. Smyth (smyth@wehi.edu.au), Matthew E. Ritchie (mritchie@wehi.edu.au)

Competing interests: The authors declare that they have no competing interests.

Grant information: This work was funded by the National Health and Medical Research Council (NHMRC) (Fellowship GNT1058892 and Program GNT1054618 to GKS, Project GNT1050661 to MER and GKS and Fellowship GNT1104924 to MER), Victorian State Government Operational Infrastructure Support and Australian Government NHMRC IRIISS.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2016 Law CW *et al.* This is an open access article distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Law CW, Alhamdoosh M, Su S *et al.* **RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 1; peer review: 3 approved]** F1000Research 2016, 5:1408 <https://doi.org/10.12688/f1000research.9005.1>

First published: 17 Jun 2016, 5:1408 <https://doi.org/10.12688/f1000research.9005.1>

Introduction

RNA-sequencing (RNA-seq) has become the primary technology for gene expression profiling and the detection of differentially expressed genes between two or more conditions is one of the most commonly asked questions using this approach. The **edgeR**¹ and **limma** packages² available from the Bioconductor project³ offer a well-developed suite of statistical methods for dealing with this type of analysis. In this article, we describe an **edgeR - limma** workflow for analysing RNA-seq data that takes gene-level counts as its input, and moves through pre-processing and exploratory data analysis before obtaining lists of differentially expressed (DE) genes and gene signatures. This analysis is enhanced through the use of interactive graphics from the **Glimma** package⁴, that allows for a more detailed exploration of the data at both the sample and gene-level than is possible using static R plots.

The experiment analysed in this workflow is from Sheridan *et al.* (2015)⁵ and consists of three cell populations (basal, luminal progenitor (LP) and mature luminal (ML)) sorted from the mammary glands of female virgin mice, each profiled in triplicate. RNA samples were sequenced across three batches on an Illumina HiSeq 2000 to obtain 100 base-pair single-end reads. The analysis outlined in this article assumes that reads obtained from an RNA-seq experiment have been aligned to an appropriate reference genome and summarised into counts associated with gene-specific regions. In this instance, reads were aligned to the mouse reference genome (mm10) using the R based pipeline available in the **Rsubread** package (specifically the `align` function⁶ followed by `featureCounts`⁷ for gene-level summarisation based on the in-built *mm10* RefSeq-based annotation). Count data for these samples can be downloaded from the Gene Expression Omnibus (GEO) <http://www.ncbi.nlm.nih.gov/geo/> using GEO Series accession number GSE63310. Further information on experimental design and sample preparation is also available from GEO under this accession number.

Data packaging

Reading in count-data

To get started with this analysis, download the file `GSE63310_RAW.tar` available online from <http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE63310&format=file>, and extract the relevant files from this archive. Each of these text files contains the raw gene-level counts for a given sample. Note that our analysis only includes the basal, LP and ML samples from this experiment (see associated file names below).

```
files <- c("GSM1545535_10_6_5_11.txt", "GSM1545536_9_6_5_11.txt",
  "GSM1545538_purep53.txt", "GSM1545539_JMS8-2.txt",
  "GSM1545540_JMS8-3.txt", "GSM1545541_JMS8-4.txt",
  "GSM1545542_JMS8-5.txt", "GSM1545544_JMS9-P7c.txt",
  "GSM1545545_JMS9-P8c.txt")
read.delim(files[1], nrow=5)

##      EntrezID GeneLength Count
## 1      497097      3634      1
## 2     100503874      3259      0
## 3     100038431      1634      0
## 4       19888      9747      0
## 5       20671      3130      1
```

Whilst each of the nine text files can be read into R separately and combined into a matrix of counts, **edgeR** offers a convenient way to do this in one step using the `readDGE` function. The resulting `DGEList`-object contains a matrix of counts with 27,179 rows associated with unique Entrez gene identifiers (IDs) and nine columns associated with the individual samples in the experiment.

```
library(limma)
library(edgeR)
x <- readDGE(files, columns=c(1,3))
class(x)

## [1] "DGEList"
## attr(,"package")
## [1] "edgeR"

dim(x)
## [1] 27179      9
```

If the counts from all samples were stored in a single file, the data can be read into R and then converted into a `DGEList`-object using the `DGEList` function.

Organising sample information

For downstream analysis, sample-level information related to the experimental design needs to be associated with the columns of the counts matrix. This should include experimental variables, both biological and technical, that could have an effect on expression levels. Examples include cell type (basal, LP and ML in this experiment), genotype (wild-type, knock-out), phenotype (disease status, sex, age), sample treatment (drug, control) and batch information (date experiment was performed if samples were collected and analysed at distinct time points) to name just a few.

Our `DGEList`-object contains a `samples` data frame that stores both cell type (or group) and batch (sequencing lane) information, each of which consists of three distinct levels. Note that within `x$samples`, library sizes are automatically calculated for each sample and normalisation factors are set to 1. For simplicity, we remove the GEO sample IDs (GSM*) from the column names of our `DGEList`-object `x`.

```

samplernames <- substring(colnames(x), 12, nchar(colnames(x)))
samplernames

## [1] "10_6_5_11" "9_6_5_11" "purep53" "JMS8-2" "JMS8-3" "JMS8-4" "JMS8-5"
## [8] "JMS9-P7c" "JMS9-P8c"

colnames(x) <- samplernames
group <- as.factor(c("LP", "ML", "Basal", "Basal", "ML", "LP",
                    "Basal", "ML", "LP"))
x$samples$group <- group
lane <- as.factor(rep(c("L004", "L006", "L008"), c(3, 4, 2)))
x$samples$lane <- lane
x$samples

##               files group lib.size norm.factors lane
## 10_6_5_11 GSM1545535_10_6_5_11.txt    LP 32863052      1 L004
## 9_6_5_11  GSM1545536_9_6_5_11.txt    ML 35335491      1 L004
## purep53   GSM1545538_purep53.txt Basal 57160817      1 L004
## JMS8-2    GSM1545539_JMS8-2.txt Basal 51368625      1 L006
## JMS8-3    GSM1545540_JMS8-3.txt    ML 75795034      1 L006
## JMS8-4    GSM1545541_JMS8-4.txt    LP 60517657      1 L006
## JMS8-5    GSM1545542_JMS8-5.txt Basal 55086324      1 L006
## JMS9-P7c  GSM1545544_JMS9-P7c.txt    ML 21311068      1 L008
## JMS9-P8c  GSM1545545_JMS9-P8c.txt    LP 19958838      1 L008

```

Organising gene annotations

A second data frame named `genes` in the `DGEList`-object is used to store gene-level information associated with rows of the counts matrix. This information can be retrieved using organism specific packages such as **Mus.musculus**⁸ for mouse (or **Homo.sapiens**⁹ for human) or the **biomaRt** package^{10,11} which interfaces the Ensembl genome databases in order to perform gene annotation. The type of information that can be retrieved includes gene symbols, gene names, chromosome names and locations, Entrez gene IDs, Refseq gene IDs and Ensembl gene IDs to name just a few. **biomaRt** primarily works off Ensembl gene IDs, whereas **Mus.musculus** packages information from various sources and allows users to choose between many different gene IDs as the key. The Entrez gene IDs available in our dataset were annotated using the **Mus.musculus** package to retrieve associated gene symbols and chromosome information.

```

library(Mus.musculus)
geneid <- rownames(x)
genes <- select(Mus.musculus, keys=geneid, columns=c("SYMBOL", "TXCHROM"),
               keytype="ENTREZID")
head(genes)

##   ENTREZID  SYMBOL TXCHROM
## 1   497097   Xkr4    chr1
## 2 100503874 Gm19938  <NA>
## 3 100038431 Gm10568  <NA>

```

```
## 4      19888      Rp1      chr1
## 5      20671      Sox17     chr1
## 6      27395      Mrpl15     chr1
```

As with any gene ID, Entrez gene IDs may not map one-to-one to the gene information of interest. It is important to check for duplicated gene IDs and resolve them to ensure the gene order is consistent between our annotation and the counts in the DGEList-object.

```
dup <- genes$ENTREZID[duplicated(genes$ENTREZID)]
genes[genes$ENTREZID %in% dup,][1:10,]
```

```
##      ENTREZID      SYMBOL TXCHROM
## 5360 100316809 Mir1906-1  chr12
## 5361 100316809 Mir1906-1  chrX
## 7128 218963      Gm1821  chr11
## 7129 218963      Gm1821  chr14
## 9564 12228       Btg3     chr16
## 9565 12228       Btg3     chr17
## 11351 433182     Eno1b    chr4
## 11352 433182     Eno1b    chr18
## 11544 100217457 Snord58b  chr14
## 11545 100217457 Snord58b  chr18
```

```
mat <- match(geneid, genes$ENTREZID)
genes <- genes[mat,]
genes[genes$ENTREZID %in% dup,][1:5,]
```

```
##      ENTREZID      SYMBOL TXCHROM
## 5360 100316809 Mir1906-1  chr12
## 7128 218963      Gm1821  chr11
## 9564 12228       Btg3     chr16
## 11351 433182     Eno1b    chr4
## 11544 100217457 Snord58b  chr14
```

The match-ing process above takes care of the duplication by keeping the first occurrence of each gene ID. Now that non-unique gene IDs have been resolved, the data frame of gene annotations is added to the data object:

```
x$genes <- genes
```

Now our data is packaged neatly in a DGEList-object containing raw count data with associated sample information and gene annotations.

```
x
## An object of class "DGEList"
## $samples
##      files group lib.size norm.factors lane
## 10_6_5_11 GSM1545535_10_6_5_11.txt  LP 32863052      1 L004
## 9_6_5_11   GSM1545536_9_6_5_11.txt   ML 35335491      1 L004
## purep53    GSM1545538_purep53.txt Basal 57160817      1 L004
## JMS8-2     GSM1545539_JMS8-2.txt Basal 51368625      1 L006
## JMS8-3     GSM1545540_JMS8-3.txt   ML 75795034      1 L006
## JMS8-4     GSM1545541_JMS8-4.txt   LP 60517657      1 L006
## JMS8-5     GSM1545542_JMS8-5.txt Basal 55086324      1 L006
## JMS9-P7c   GSM1545544_JMS9-P7c.txt  ML 21311068      1 L008
## JMS9-P8c   GSM1545545_JMS9-P8c.txt  LP 19958838      1 L008
##
## $counts
```

```
##
##      Samples
## Tags    10_6_5_11  9_6_5_11 purep53 JMS8-2 JMS8-3 JMS8-4 JMS8-5 JMS9-P7c JMS9-P8c
## 497097      1      2      342      526      3      3      535      2      0
## 100503874    0      0      5      6      0      0      5      0      0
## 100038431    0      0      0      0      0      0      1      0      0
## 19888        0      1      0      0      17     2      0      1      0
## 20671        1      1      76     40     33     14     98     18     8
## 27174 more rows ...
##
## $genes
##      ENTREZID  SYMBOL TXCHROM
## 1      497097    Xkr4     chr1
## 2 100503874    Gm19938    <NA>
## 3 100038431    Gm10568    <NA>
## 4      19888     Rp1      chr1
## 5      20671     Sox17     chr1
## 27174 more rows ...
```

Data pre-processing

Transformations from the raw-scale

For differential expression and related analyses, gene expression is rarely considered at the level of raw counts since libraries sequenced at a greater depth will result in higher counts. Rather, it is common practice to transform raw counts onto a scale that accounts for such library size differences. Popular transformations include counts per million (CPM), \log_2 -counts per million (log-CPM), reads per kilobase of transcript per million (RPKM), and fragments per kilobase of transcript per million (FPKM).

In our analyses, CPM and log-CPM transformations are used regularly although they do not account for feature length differences which RPKM and FPKM values do. Whilst RPKM and FPKM values can just as well be used, CPM and log-CPM values can be calculated using a counts matrix alone and will suffice for the type of comparisons we are interested in. Differential expression analyses look at gene expression differences between conditions, rather than comparing expression across multiple genes or drawing conclusions on absolute levels of expression. In other words, gene lengths remain constant for comparisons of interest and any observed differences are a result of changes in condition rather than changes in gene length.

Here raw counts are converted to CPM and log-CPM values using the `cpm` function in **edgeR**, where log-transformations use a prior count of 0.25 to avoid taking the log of zero. RPKM values are just as easily calculated as CPM values using the `rpkm` function in **edgeR** if gene lengths are available.

```
cpm <- cpm(x)
lcpm <- cpm(x, log=TRUE)
```

Removing genes that are not expressed

All datasets will include a mix of genes that are expressed and those that are not expressed. Whilst it is of interest to examine genes that are expressed in one condition but not in another, some genes are unexpressed throughout all samples. In fact, 19% of genes in this dataset have zero counts across all nine samples.

```
table(rowSums(x$counts==0)==9)

##
## FALSE TRUE
## 22026 5153
```

Genes that are not expressed at a biologically meaningful level in any condition should be discarded to reduce the subset of genes to those that are of interest, and to reduce the number of tests carried out downstream when looking at differential expression. Upon examination of log-CPM values, it can be seen that a large proportion of genes within each sample is unexpressed or lowly-expressed (Figure 1A). Using a nominal CPM value of 1 (which is equivalent to a log-CPM value of 0) a gene is deemed to be expressed in a given sample if its transformed count is above this threshold, and unexpressed otherwise. Genes must be expressed in at least one group (or roughly any three samples across the entire experiment) to be kept for downstream analysis.

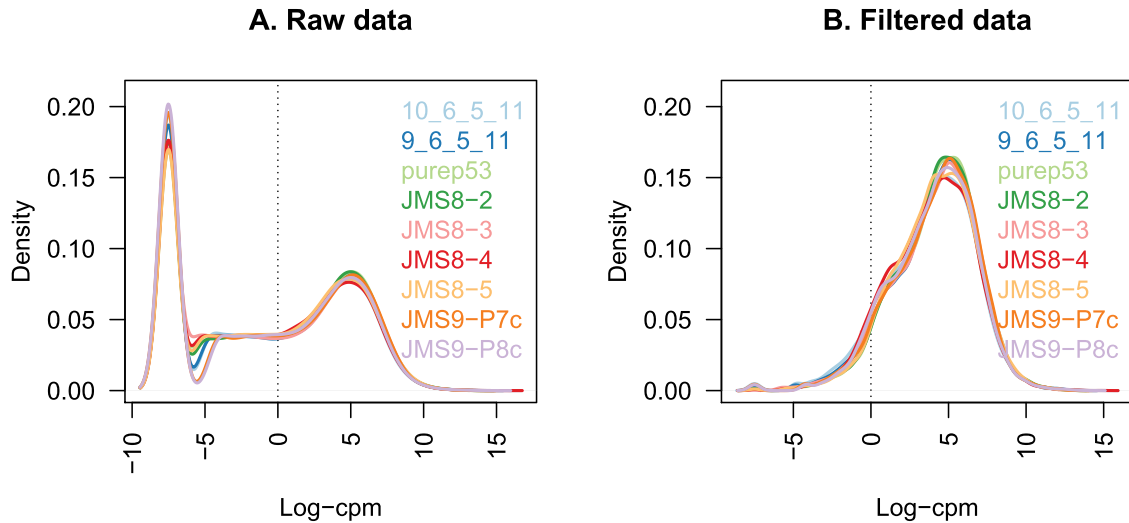


Figure 1. The density of log-CPM values for raw pre-filtered data (**A**) and post-filtered data (**B**) are shown for each sample. Dotted vertical lines mark the log-CPM of zero threshold (equivalent to a CPM value of 1) used in the filtering step.

```
keep.exprs <- rowSums(cpm>1)>=3
x <- x[keep.exprs,, keep.lib.sizes=FALSE]
dim(x)
```

```
## [1] 14165      9
```

Using this criterion, the number of genes is reduced to approximately half the number that we started with (14,165 genes, **Figure 1B**). Note that subsetting the entire DGEList-object removes both the counts as well as the associated gene information. Code to produce **Figure 1** is given below.

```
library(RColorBrewer)
nsamples <- ncol(x)
col <- brewer.pal(nsamples, "Paired")
par(mfrow=c(1,2))
plot(density(lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.21), las=2,
     main="", xlab="")
title(main="A. Raw data", xlab="Log-cpm")
abline(v=0, lty=3)
for (i in 2:nsamples){
  den <- density(lcpm[,i])
  lines(den$x, den$y, col=col[i], lwd=2)
}
legend("topright", samplenames, text.col=col, bty="n")
lcpm <- cpm(x, log=TRUE)
plot(density(lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.21), las=2,
     main="", xlab="")
title(main="B. Filtered data", xlab="Log-cpm")
abline(v=0, lty=3)
for (i in 2:nsamples){
  den <- density(lcpm[,i])
  lines(den$x, den$y, col=col[i], lwd=2)
}
legend("topright", samplenames, text.col=col, bty="n")
```

Normalising gene expression distributions

During the sample preparation or sequencing process, external factors that are not of biological interest can affect the gene expression levels of individual samples. For example, samples processed in the first batch of an experiment may have been sequenced more deeply than samples processed in a second batch. It is assumed that all samples should have a similar range and distribution of expression values. Normalisation is generally required to ensure that the expression distributions of each sample are similar across the entire experiment.

Any plot showing the per sample expression distributions, such as a density or boxplot, is useful in determining whether any samples are dissimilar to others. Distributions of log-CPM values are similar throughout all samples within this dataset ([Figure 1B](#)).

Nonetheless, normalisation by the method of trimmed mean of M-values (TMM)¹² is performed using the `calcNormFactors` function in **edgeR**. The normalisation factors calculated here are used as a scaling factor for the library sizes. When working with `DGEList`-objects, these normalisation factors are automatically stored in `x$samples$norm.factors`. For this dataset the effect of TMM-normalisation is mild, as evident in the magnitude of the scaling factors, which are all relatively close to 1.

```
x <- calcNormFactors(x, method = "TMM")
x$samples$norm.factors

## [1] 0.896 1.035 1.044 1.041 1.032 0.922 0.984 1.083 0.979
```

To give a better visual representation of the effects of normalisation, the data was duplicated then adjusted so that the counts of the first sample are reduced to 5% of their original values, and in the second sample they are inflated to be 5-times larger.

```
x2 <- x
x2$samples$norm.factors <- 1
x2$counts[,1] <- ceiling(x2$counts[,1]*0.05)
x2$counts[,2] <- x2$counts[,2]*5
```

[Figure 2](#) shows the expression distribution of samples for unnormalised and normalised data, where distributions are noticeably different pre-normalisation and are similar post-normalisation. Here the first sample has a small TMM scaling factor of 0.05, whereas the second sample has a large scaling factor of 6.13 – neither values are close to 1.

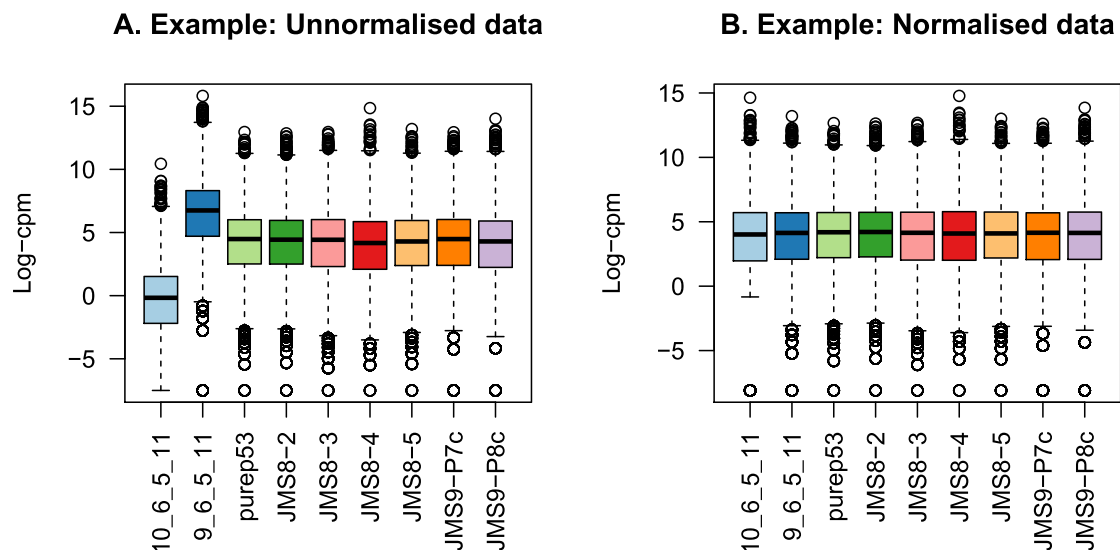


Figure 2. Example data: Boxplots of log-CPM values showing expression distributions for unnormalised data (**A**) and normalised data (**B**) for each sample in the modified dataset where the counts in samples 1 and 2 have been scaled to 5% and 500% of their original values respectively.


```

par(mfrow=c(1,2))
lcpm <- cpm(x2, log=TRUE)
boxplot(lcpm, las=2, col=col, main="")
title(main="A. Example: Unnormalised data",ylab="Log-cpm")
x2 <- calcNormFactors(x2)
x2$samples$norm.factors

## [1] 0.0547 6.1306 1.2293 1.1705 1.2149 1.0562 1.1459 1.2613 1.1170

lcpm <- cpm(x2, log=TRUE)
boxplot(lcpm, las=2, col=col, main="")
title(main="B. Example: Normalised data",ylab="Log-cpm")

```

Unsupervised clustering of samples

In our opinion, one of the most important exploratory plots to examine for gene expression analyses is the multidimensional scaling (MDS) plot, or similar. The plot shows similarities and dissimilarities between samples in an unsupervised manner so that one can have an idea of the extent to which differential expression can be detected before carrying out formal tests. Ideally, samples would cluster well within the primary condition of interest, and any sample straying far from its group could be identified and followed up for sources of error or extra variation. If present, technical replicates should lie very close to one another.

Such a plot can be made in **limma** using the `plotMDS` function. The first dimension represents the leading-fold-change that best separates samples and explains the largest proportion of variation in the data, with subsequent dimensions having a smaller effect and being orthogonal to the ones before it. When experimental design involves multiple factors, it is recommended that each factor is examined over several dimensions. If samples cluster by a given factor in any of these dimensions, it suggests that the factor contributes to expression differences and is worth including in the linear modelling. On the other hand, factors that show little or no effect may be left out of downstream analysis.

In this dataset, samples can be seen to cluster well within experimental groups over dimension 1 and 2, and then separate by sequencing lane (sample batch) over dimension 3 (Figure 3). Keeping in mind that the first dimension explains the largest proportion of variation in the data, notice that the range of values over the dimensions become smaller as we move to higher dimensions. Whilst all samples cluster by groups, the largest transcriptional difference is observed between basal and LP, and basal and ML over dimension 1. For this reason, it is expected that pairwise comparisons between cell populations will result in a greater number of DE genes for comparisons involving basal samples,

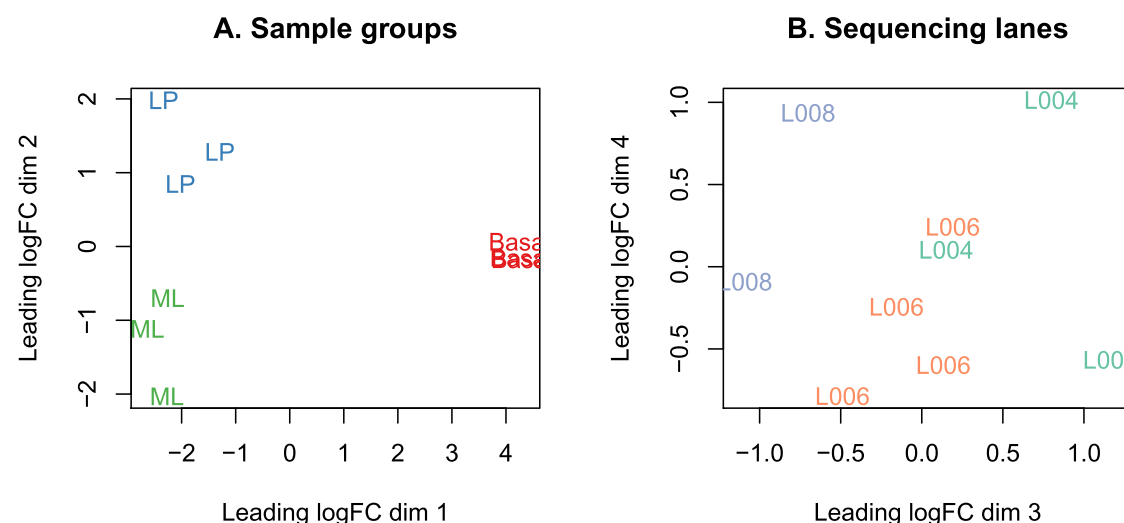


Figure 3. MDS plots of log-CPM values over dimensions 1 and 2 with samples coloured and labeled by sample groups (A) and over dimensions 3 and 4 with samples coloured and labeled by sequencing lane (B). Distances on the plot correspond to the leading fold-change, which is the average (root-mean-square) \log_2 -fold-change for the 500 genes most divergent between each pair of samples by default.

and relatively small numbers of DE genes when comparing ML to LP. In other datasets, samples that do not cluster by their groups of interest may also show little or no evidence of differential expression in the downstream analysis.

To create the MDS plots, different colour groupings are assigned to factors of interest. Dimensions 1 and 2 are examined using the color grouping defined by cell types.

```
lcpm <- cpm(x, log=TRUE)
par(mfrow=c(1,2))
col.group <- group
levels(col.group) <- brewer.pal(nlevels(col.group), "Set1")
col.group <- as.character(col.group)
col.lane <- lane
levels(col.lane) <- brewer.pal(nlevels(col.lane), "Set2")
col.lane <- as.character(col.lane)
plotMDS(lcpm, labels=group, col=col.group)
title(main="A. Sample groups")
```

Dimensions 3 and 4 are examined using the colour grouping defined by sequencing lanes (batch).

```
plotMDS(lcpm, labels=lane, col=col.lane, dim=c(3,4))
title(main="B. Sequencing lanes")
```

Alternatively, the **Glimma** package offers the convenience of an interactive MDS plot where multiple dimensions can be explored. The `glMDSPlot` function generates an html page (that is opened in a browser if `launch=TRUE`) with an MDS plot in the left panel and a barplot showing the proportion of variation explained by each dimension in the right panel. Clicking on the bars of the bar plot changes the pair of dimensions plotted in the MDS plot, and hovering over the individual points reveals the sample label. The colour scheme can be changed as well to highlight cell population or sequencing lane (batch). An interactive MDS plot of this dataset can be found at <http://bioinf.wehi.edu.au/folders/limaWorkflow/glimma-plots/MDS-Plot.html>.

```
library(Glimma)
glMDSPlot(lcpm, labels=paste(group, lane, sep="_"),
          groups=x$samples[,c(2,5)], launch=FALSE)
```

Differential expression analysis

Creating a design matrix and contrasts

In this study, it is of interest to see which genes are expressed at different levels between the three cell populations profiled. In our analysis, linear models are fitted to the data with the assumption that the underlying data is normally distributed. To get started, a design matrix is set up with both the cell population and sequencing lane (batch) information.

```
design <- model.matrix(~0+group+lane)
colnames(design) <- gsub("group", "", colnames(design))
design

##      Basal LP ML laneL006 laneL008
## 1      0  1  0          0          0
## 2      0  0  1          0          0
## 3      1  0  0          0          0
## 4      1  0  0          1          0
## 5      0  0  1          1          0
## 6      0  1  0          1          0
## 7      1  0  0          1          0
## 8      0  0  1          0          1
## 9      0  1  0          0          1
## attr("assign")
## [1] 1 1 1 2 2
## attr("contrasts")
## attr("contrasts")$group
```

```
## [1] "contr.treatment"
##
## attr(,"contrasts")$lane
## [1] "contr.treatment"
```

For a given experiment, there are usually several equivalent ways to set up an appropriate design matrix. For example, `~0+group+lane` removes the intercept from the first factor, `group`, but an intercept remains in the second factor `lane`. Alternatively, `~group+lane` could be used to keep the intercepts in both `group` and `lane`. Understanding how to interpret the coefficients estimated in a given model is key here. We choose the first model for our analysis, as setting up model contrasts is more straight forward in the absence of an intercept for `group`. Contrasts for pairwise comparisons between cell populations are set up in **limma** using the `makeContrasts` function.

```
contr.matrix <- makeContrasts(
  BasalvsLP = Basal-LP,
  BasalvsML = Basal - ML,
  LPvsML = LP - ML,
  levels = colnames(design))
contr.matrix
```

##		Contrasts		
##	Levels	BasalvsLP	BasalvsML	LPvsML
##	Basal	1	1	0
##	LP	-1	0	1
##	ML	0	-1	-1
##	laneL006	0	0	0
##	laneL008	0	0	0

A key strength of **limma**'s linear modelling approach, is the ability accommodate arbitrary experimental complexity. Simple designs, such as the one in this workflow, with cell type and batch, through to more complicated factorial designs and models with interaction terms can be handled relatively easily. Where experimental or technical effects can be modelled using a random effect, another possibility in **limma** is to estimate correlations using `duplicateCorrelation` by specifying a `block` argument for both this function and in the `lmFit` linear modelling step.

Removing heteroscedasticity from count data

It has been shown that for RNA-seq count data, the variance is not independent of the mean¹³ – this is true of raw counts or when transformed to log-CPM values. Methods that model counts using a Negative Binomial distribution assume a quadratic mean-variance relationship. In **limma**, linear modelling is carried out on the log-CPM values which are assumed to be normally distributed and the mean-variance relationship is accommodated using precision weights calculated by the `voom` function.

When operating on a `DGEList`-object, `voom` converts raw counts to log-CPM values by automatically extracting library sizes and normalisation factors stored in the object. For a matrix of counts, the method of normalisation can be specified within `voom` using the `normalize.method` (by default no normalisation is performed).

The mean-variance relationship of log-CPM values for this dataset is shown in [Figure 4A](#). Typically, the “voom-plot” shows a decreasing trend between the means and variances resulting from a combination of technical variation in the sequencing experiment and biological variation amongst the replicate samples from different cell populations. Experiments with high biological variation usually result in flatter trends, where variance values plateau at high expression values. Experiments with low biological variation tend to result in sharp decreasing trends.

Moreover, the voom-plot provides a visual check on the level of filtering performed upstream. If filtering of lowly-expressed genes is insufficient, a drop in variance levels can be observed at the low end of the expression scale due to very small counts. If this is observed, one should return to the earlier filtering step and increase the expression threshold applied to the dataset.

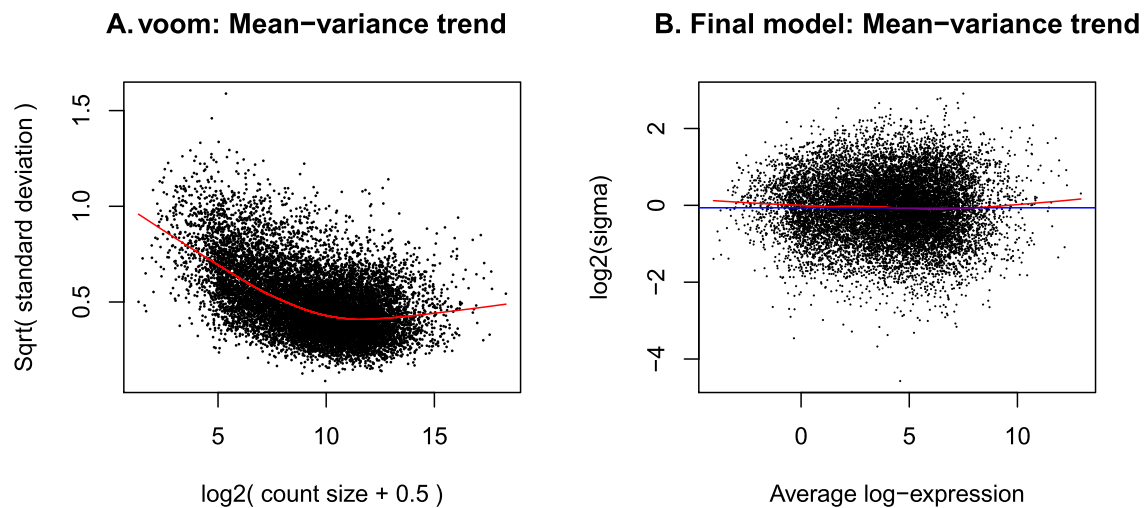


Figure 4. Means (x-axis) and variances (y-axis) of each gene are plotted to show the dependence between the two before `voom` is applied to the data (**A**) and how the trend is removed after `voom` precision weights are applied to the data (**B**). The plot on the left is created within the `voom` function which extracts residual variances from fitting linear models to log-CPM transformed data. Variances are then rescaled to quarter-root variances (or square-root of standard deviations) and plotted against the mean expression of each gene. The means are \log_2 -transformed mean-counts with an offset of 0.5. The plot on the right is created using `plotSA` which plots \log_2 residual standard deviations against mean log-CPM values. The average \log_2 residual standard deviation is marked by a horizontal blue line. In both plots, each black dot represents a gene and a red curve is fitted to these points.

Where sample-level variation is evident from earlier inspections of the MDS plot, the `voomWithQualityWeights` function can be used to simultaneously incorporate sample-level weights together with the abundance dependent weights estimated by `voom`¹⁴. For an example of this, see Liu *et al.* (2016)¹⁵.

```
v <- voom(x, design, plot=TRUE)
v
```

```
## An object of class "EList"
## $genes
##      ENTREZID SYMBOL TXCHROM
## 1      497097   Xkr4      chr1
## 6       27395 Mrpl15      chr1
## 7       18777 Lypla1      chr1
## 9       21399 Tceal       chr1
## 10      58175 Rgs20       chr1
## 14160 more rows ...
##
## $targets
##
##          files      group lib.size norm.factors lane
## 10_6_5_11 GSM1545535_10_6_5_11.txt LP 29409426      0.896 L004
## 9_6_5_11   GSM1545536_9_6_5_11.txt ML 36528591      1.035 L004
## purep53    GSM1545538_purep53.txt Basal 59598629      1.044 L004
## JMS8-2     GSM1545539_JMS8-2.txt Basal 53382070      1.041 L006
## JMS8-3     GSM1545540_JMS8-3.txt ML 78175314      1.032 L006
## JMS8-4     GSM1545541_JMS8-4.txt LP 55762781      0.922 L006
## JMS8-5     GSM1545542_JMS8-5.txt Basal 54115150      0.984 L006
## JMS9-P7c   GSM1545544_JMS9-P7c.txt ML 23043111      1.083 L008
## JMS9-P8c   GSM1545545_JMS9-P8c.txt LP 19525423      0.979 L008
##
## $E
##
## Samples
```

```
## Tags      10_6_5_11 9_6_5_11 purep53 JMS8-2 JMS8-3 JMS8-4 JMS8-5 JMS9-P7c JMS9-P8c
## 497097    -4.29    -3.87    2.52    3.30    -4.48    -3.9    3.31    -3.20    -5.29
## 27395      3.88     4.40     4.52     4.57     4.32     3.79     3.92     4.35     4.13
## 18777      4.71     5.56     5.40     5.17     5.63     5.08     5.08     5.76     5.15
## 21399      4.78     4.74     5.37     5.13     4.85     4.94     5.16     5.04     4.99
## 58175      3.94     3.29    -1.77    -1.88     2.99     3.36    -2.11     3.14     3.52
## 14160 more rows ...
##
## $weights
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,] 1.18 1.18 20.53 20.98 1.77 1.22 21.13 1.18 1.18
## [2,] 20.88 26.56 31.60 29.66 32.56 26.75 29.79 21.90 17.15
## [3,] 28.00 33.70 34.85 34.46 35.15 33.55 34.52 31.44 25.23
## [4,] 27.67 29.60 34.90 34.43 34.84 33.16 34.49 26.14 24.50
## [5,] 19.74 18.66 3.18 2.63 24.19 24.01 2.65 13.15 14.35
## 14160 more rows ...
##
## $design
##      Basal LP ML laneL006 laneL008
## 1      0  1  0      0      0
## 2      0  0  1      0      0
## 3      1  0  0      0      0
## 4      1  0  0      1      0
## 5      0  0  1      1      0
## 6      0  1  0      1      0
## 7      1  0  0      1      0
## 8      0  0  1      0      1
## 9      0  1  0      0      1
## attr(,"assign")
## [1] 1 1 1 2 2
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
##
## attr(,"contrasts")$lane
## [1] "contr.treatment"
```

Note that the other data frames stored within the `DGEList`-object that contain gene- and sample-level information, are retained in the `EList`-object `v` created by `voom`. The `v$genes` data frame is equivalent to `x$genes`, `v$targets` is equivalent to `x$samples`, and the expression values stored in `v$E` is analogous to `x$counts`, albeit on a transformed scale. In addition to this, the `voom` `EList`-object has a matrix of precision weights `v$weights` and stores the design matrix in `v$design`.

Fitting linear models for comparisons of interest

Linear modelling in **limma** is carried out using the `lmFit` and `contrasts.fit` functions which can be used for both microarray and RNA-seq data and fit a separate model to the expression values for each gene. Next, empirical Bayes moderation is carried out by the `eBayes` function which borrows information across all the genes to obtain more precise estimates of gene-wise variability¹⁶. The model's residual variances are plotted against average expression values in [Figure 4B](#). It can be seen from this plot that the variance is no longer dependent on the mean expression level.

```
vfit <- lmFit(v, design)
vfit <- contrasts.fit(vfit, contrasts=contr.matrix)
efit <- eBayes(vfit)
plotSA(efit)
```

Examining the number of DE genes

For a quick look at differential expression levels, the number of significantly up- and down-regulated genes can be summarised in a table. Significance is defined using an adjusted *p*-value cutoff that is set at 5% by default. For the

comparison between expression levels in basal and LP, 4,127 genes are found to be down-regulated in basal relative to LP and 4,298 genes are up-regulated in basal relative to LP – a total of 8,425 DE genes. A total of 8,510 DE genes are found between basal and ML (4,338 down- and 4,172 up-regulated genes), and a total of 5,340 DE genes are found between LP and ML (2,895 down- and 2,445 up-regulated). The larger numbers of DE genes observed for comparisons involving the basal population are consistent with our observations from the MDS plots.

```
summary(decideTests(efit))
```

```
##      BasalvsLP BasalvsML LPvsML
## -1         4127         4338         2895
##  0         5740         5655         8825
##  1         4298         4172         2445
```

Some studies require more than an adjusted *p*-value cut-off. For a stricter definition on significance, one may require log-fold-changes (log-FCs) to be above a minimum value. The `treat` method¹⁷ can be used to calculate *p*-values from empirical Bayes moderated *t*-statistics with a minimum log-FC requirement. The number of differentially expressed genes are reduced to a total of 3,135 DE genes for basal versus LP, 3,270 DE genes for basal versus ML, and 385 DE genes for LP versus ML when testing requires genes to have a log-FC that is significantly greater than 1 (equivalent to a 2-fold difference between cell types on the original scale).

```
tfit <- treat(vfit, lfc=1)
dt <- decideTests(tfit)
summary(dt)
##      BasalvsLP BasalvsML LPvsML
## -1         1417         1512         203
##  0        11030        10895       13780
##  1         1718         1758         182
```

Genes that are DE in multiple comparisons can be extracted using the results from `decideTests`, where 0s represent genes that are not DE, 1s represent genes that are up-regulated, and -1s represent genes that are down-regulated. A total of 2,409 genes are DE in both basal versus LP and basal versus ML (Figure 5), twenty of which are listed below. The `write.fit` function can be used to extract and write results for all three comparisons to a single output file.

```
de.common <- which(dt[,1]!=0 & dt[,2]!=0)
length(de.common)
## [1] 2409
```

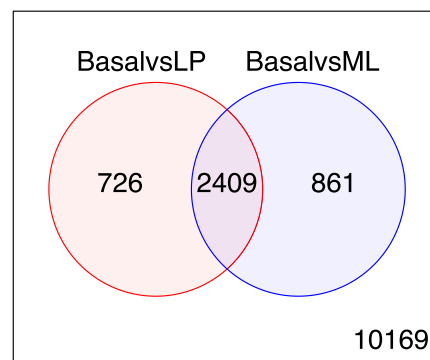


Figure 5. Venn diagram showing the number of genes DE in the comparison between basal versus LP only (left), basal versus ML only (right), and the number of genes that are DE in both comparisons (center). The number of genes that are not DE in either comparison are marked in the bottom-right.

```
head(tfit$genes$SYMBOL[de.common], n=20)

## [1] "Xkr4"          "Rgs20"          "Cpa6"           "Sulf1"          "Eya1"
## [6] "Msc"           "Sbspon"         "Pi15"           "Crispld1"       "Kcnq5"
## [11] "Ptpn18"        "Arhgef4"        "2010300C02Rik" "Aff3"           "Npas2"
## [16] "Tbc1d8"        "Creg2"          "Il1r1"          "Il18r1"         "Il18rap"

vennDiagram(dt[,1:2], circle.col=c("turquoise", "salmon"))
write.fit(tfit, dt, file="results.txt")
```

Examining individual DE genes from top to bottom

The top DE genes can be listed using `topTreat` for results using `treat` (or `topTable` for results using `eBayes`). By default `topTreat` arranges genes from smallest to largest adjusted *p*-value with associated gene information, log-FC, average log-CPM, moderated *t*-statistic, raw and adjusted *p*-value for each gene. The number of top genes displayed can be specified, where `n=Inf` includes all genes. Genes *Cldn7* and *Rasef* are amongst the top DE genes for both basal versus LP and basal versus ML.

```
basal.vs.lp <- topTreat(tfit, coef=1, n=Inf)
basal.vs.ml <- topTreat(tfit, coef=2, n=Inf)
head(basal.vs.lp)

##      ENTREZID SYMBOL TXCHROM logFC AveExpr      t P.Value adj.P.Val
## 12759      12759   Clu   chr14 -5.44   8.86 -33.4 3.99e-10  2.7e-06
## 53624      53624  Cldn7  chr11 -5.51   6.30 -32.9 4.50e-10  2.7e-06
## 242505     242505  Rasef   chr4 -5.92   5.12 -31.8 6.06e-10  2.7e-06
## 67451      67451  Pkp2  chr16 -5.72   4.42 -30.7 8.01e-10  2.7e-06
## 228543     228543  Rhov   chr2 -6.25   5.49 -29.5 1.11e-09  2.7e-06
## 70350      70350  Basp1  chr15 -6.07   5.25 -28.6 1.38e-09  2.7e-06

head(basal.vs.ml)

##      ENTREZID SYMBOL TXCHROM logFC AveExpr      t P.Value adj.P.Val
## 242505     242505  Rasef   chr4 -6.51   5.12 -35.5 2.57e-10  1.92e-06
## 53624      53624  Cldn7  chr11 -5.47   6.30 -32.5 4.98e-10  1.92e-06
## 12521      12521  Cd82   chr2 -4.67   7.07 -31.8 5.80e-10  1.92e-06
## 71740      71740  Pvr14  chr1 -5.56   5.17 -31.3 6.76e-10  1.92e-06
## 20661      20661  Sort1  chr3 -4.91   6.71 -31.2 6.76e-10  1.92e-06
## 15375      15375  Foxa1  chr12 -5.75   5.63 -28.3 1.49e-09  2.28e-06
```

Useful graphical representations of differential expression results

To summarise results for all genes visually, mean-difference plots, which display log-FCs from the linear model fit against the average log-CPM values can be generated using the `plotMD` function, with the differentially expressed genes highlighted.

```
plotMD(tfit, column=1, status=dt[,1], main=colnames(tfit)[1],
       xlim=c(-8,13))
```

Glimma extends this functionality by providing an interactive mean-difference plot via the `glMDPlot` function. The output of this function is an html page, with summarised results in the left panel (similar to what is output by `plotMD`), and the log-CPM values from individual samples in the right panel, with a table of results below the plots (Figure 6). This interactive display allows the user to search for particular genes based on their Gene symbol, which is not possible in a static R plot. The `glMDPlot` function is not limited to mean-difference plots, with a default version allowing a data frame to be passed with the user able to select the columns of interest to plot in the left panel.

```
glMDPlot(tfit, coef=1, status=dt[,1], main=colnames(tfit)[1],
         counts=x$counts, samples=colnames(x), anno=x$genes, groups=group,
         id.column="ENTREZID", display.columns=c("SYMBOL", "ENTREZID"),
         search.by="SYMBOL", launch=FALSE)
```

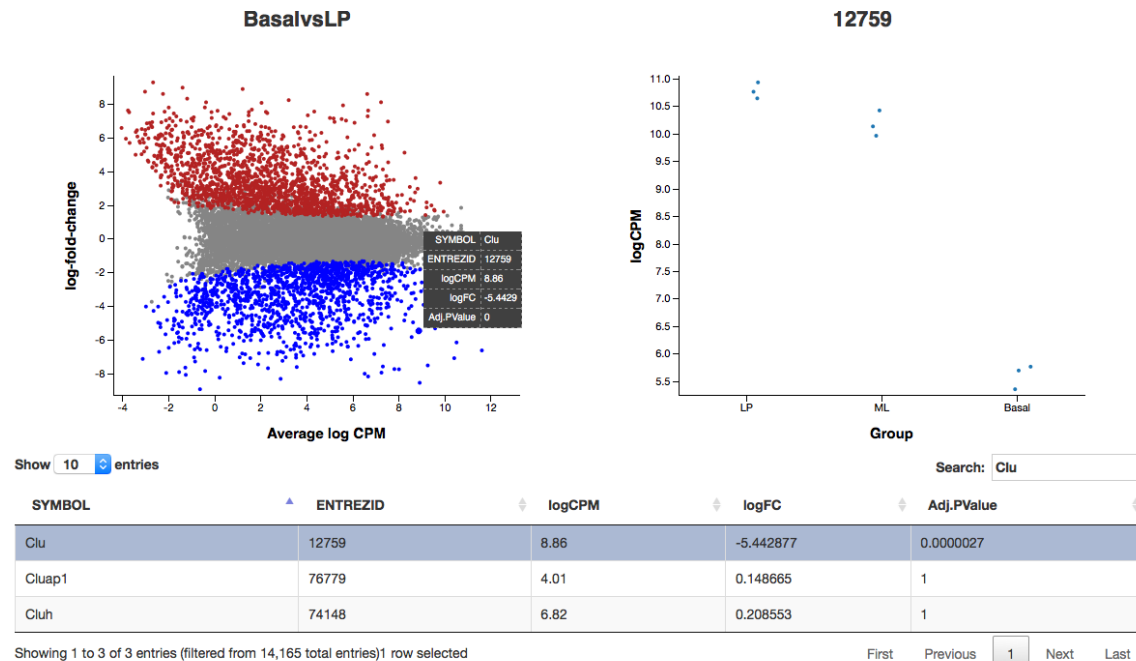


Figure 6. Interactive mean-difference plot generated using Glimma. Summary data (log-FCs versus log-CPM values) are shown in the left panel which is linked to the individual values per sample for a selected gene in the right panel. A table of results is also displayed below these figures, along with a search bar to allow users to look up a particular gene using its Gene symbol identifier, e.g. *Clu*.

The mean-difference plot generated by the command above is available online (see <http://bioinf.wehi.edu.au/folders/limmaWorkflow/glimma-plots/MD-Plot.html>). The interactivity provided by the **Glimma** package allows additional information to be presented in a single graphical window. **Glimma** is implemented in R and Javascript, with the R code generating the data which is converted into graphics using the Javascript library D3 (<https://d3js.org>), with the Bootstrap library handling layouts and Datatables generating the interactive searchable tables. This allows plots to be viewed in any modern browser, which is convenient for including them as linked files from an Rmarkdown report of the analysis.

Plots shown previously include either all of the genes that are expressed in any one condition (such as the Venn diagram of common DE genes or mean-difference plot) or look at genes individually (log-CPM values shown in right panel of the interactive mean-difference plot). Heatmaps allow users to look at the expression of a subset of genes. This can give useful insight into the expression of individual groups and samples without losing perspective of the overall study when focusing on individual genes, or losing resolution when examining patterns averaged over thousands of genes at the same time.

A heatmap is created for the top 100 DE genes (as ranked by adjusted p-value) from the basal versus LP contrast using the `heatmap.2` function from the **gplots** package (Figure 7). The heatmap correctly clusters samples into cell type and rearranges the order of genes to form blocks of similar expression. From the heatmap, we observe that the expression of ML and LP samples are very similar for the top 100 DE genes between basal and LP.

```
library(gplots)
basal.vs.lp.topgenes <- basal.vs.lp$ENTREZID[1:100]
i <- which(v$genes$ENTREZID %in% basal.vs.lp.topgenes)
mycol <- colorpanel(1000,"blue","white","red")
heatmap.2(v$E[i,], scale="row",
  labRow=v$genes$SYMBOL[i], labCol=group,
  col=mycol, trace="none", density.info="none",
  margin=c(8,6), lhei=c(2,10), dendrogram="column")
```

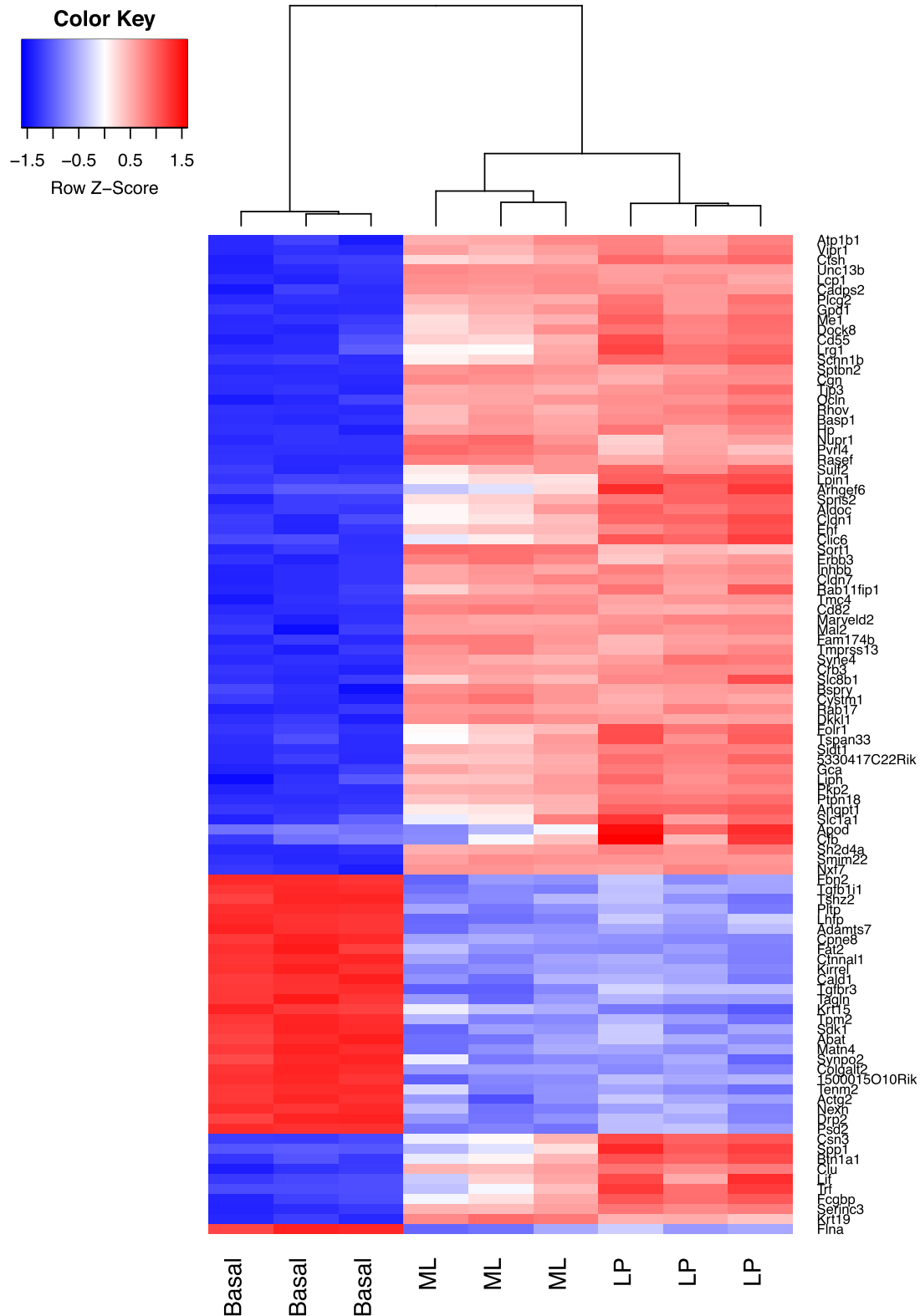



Figure 7. Heatmap of log-CPM values for top 100 genes DE in basal versus LP. Expression across each gene (or row) have been scaled so that mean expression is zero and standard deviation is one. Samples with relatively high expression within a gene are marked in red, samples with relatively low expression are marked in blue. Lighter shades and white represent genes with intermediate expression levels. Samples and genes have been reordered by the method of hierarchical clustering. A dendrogram is shown for the clustering of samples.

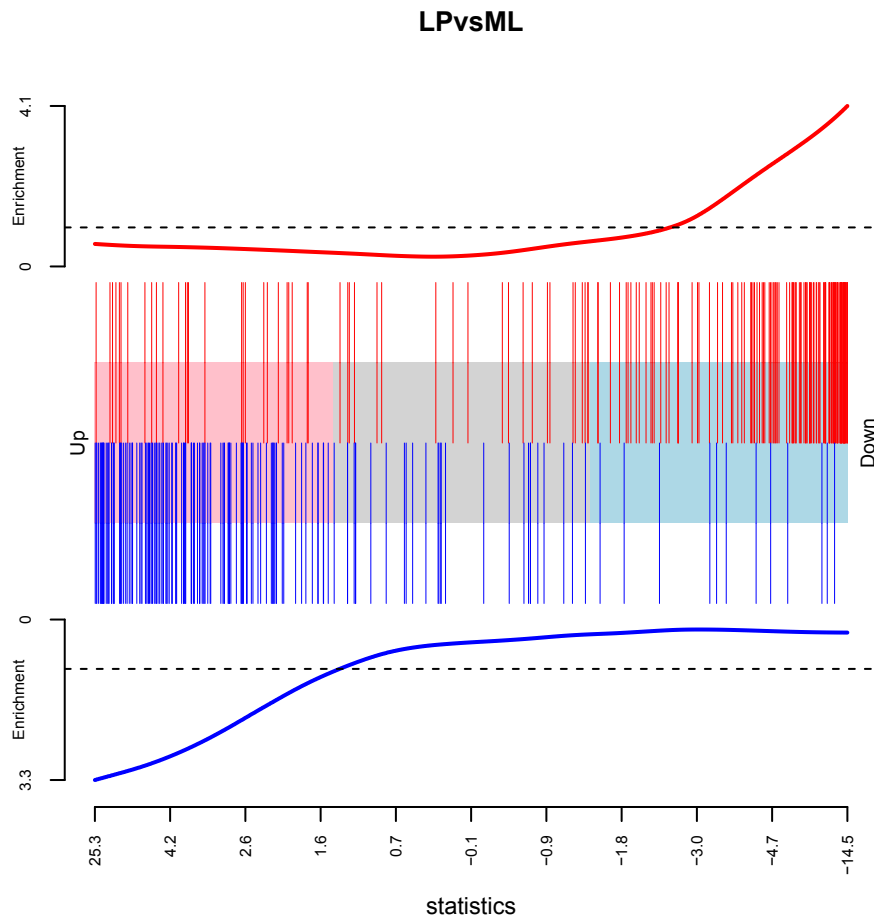


Figure 8. Barcode plot of LIM_MAMMARY_LUMINAL_MATURE_UP (red bars, top of plot) and LIM_MAMMARY_LUMINAL_MATURE_DN (blue bars, bottom of plot) gene sets in the LP versus ML contrast with an enrichment line for each set that shows the relative enrichment of the vertical bars in each part of the plot. The experiment of Lim *et al.* (2010) is very similar to the current one, with the same sorting strategy used to obtain the different cell populations, except that microarrays were used instead of RNA-seq to profile gene expression. Note that the inverse correlation (the up gene set is down and the down gene set is up) is a result of the way the contrast has been set up (LP versus ML) – if reversed, the directionality would agree.

Gene set testing with camera

We finish this analysis with gene set testing by applying the *camera* method¹⁸ to the *c2* gene signatures from the Broad Institute's MSigDB *c2* collection¹⁹ that have been adapted for mouse and are available as Rdata objects from <http://bioinf.wehi.edu.au/software/MSigDB/>.

```
load(url("http://bioinf.wehi.edu.au/software/MSigDB/mouse_c2_v5p1.rdata"))
idx <- ids2indices(Mm.c2, id=rownames(v))
cam.BasalvsLP <- camera(v, idx, design=contr.matrix[,1],
                        inter.gene.cor=0.01)
```

```
head(cam.BasalvsLP, 5)
```

	NGenes	Correlation	Direction	PValue
LIM_MAMMARY_STEM_CELL_UP	739	0.01	Up	1.13e-18
LIM_MAMMARY_STEM_CELL_DN	630	0.01	Down	1.57e-15
ROSTY_CERVICAL_CANCER_PROLIFERATION_CLUSTER	163	0.01	Up	1.44e-13
SOTIRIOU_BREAST_CANCER_GRADE_1_VS_3_UP	183	0.01	Up	2.18e-13
LIM_MAMMARY_LUMINAL_PROGENITOR_UP	87	0.01	Down	6.73e-13

```
##
## FDR
## LIM_MAMMARY_STEM_CELL_UP 5.36e-15
## LIM_MAMMARY_STEM_CELL_DN 3.71e-12
## ROSTY_CERVICAL_CANCER_PROLIFERATION_CLUSTER 2.26e-10
## SOTIRIOU_BREAST_CANCER_GRADE_1_VS_3_UP 2.58e-10
## LIM_MAMMARY_LUMINAL_PROGENITOR_UP 6.36e-10
cam.BasalvsML <- camera(v,idx,design,contrast=contr.matrix[,2],
                        inter.gene.cor=0.01)

head(cam.BasalvsML, 5)
```

	NGenes	Correlation	Direction	PValue
LIM_MAMMARY_STEM_CELL_UP	739	0.01	Up	5.09e-23
LIM_MAMMARY_STEM_CELL_DN	630	0.01	Down	5.13e-19
LIM_MAMMARY_LUMINAL_MATURE_DN	166	0.01	Up	8.88e-16
LIM_MAMMARY_LUMINAL_MATURE_UP	180	0.01	Down	6.29e-13
ROSTY_CERVICAL_CANCER_PROLIFERATION_CLUSTER	163	0.01	Up	1.68e-12

```
##
## FDR
## LIM_MAMMARY_STEM_CELL_UP 2.40e-19
## LIM_MAMMARY_STEM_CELL_DN 1.21e-15
## LIM_MAMMARY_LUMINAL_MATURE_DN 1.40e-12
## LIM_MAMMARY_LUMINAL_MATURE_UP 7.43e-10
## ROSTY_CERVICAL_CANCER_PROLIFERATION_CLUSTER 1.59e-09

cam.LPvsML <- camera(v,idx,design,contrast=contr.matrix[,3],
                     inter.gene.cor=0.01)

head(cam.LPvsML, 5)
```

	NGenes	Correlation	Direction	PValue	FDR
LIM_MAMMARY_LUMINAL_MATURE_UP	180	0.01	Down	8.50e-14	3.40e-10
LIM_MAMMARY_LUMINAL_MATURE_DN	166	0.01	Up	1.44e-13	3.40e-10
LIM_MAMMARY_LUMINAL_PROGENITOR_UP	87	0.01	Up	3.84e-11	6.05e-08
REACTOME_RESPIRATORY_ELECTRON_TRANSPORT	91	0.01	Down	2.66e-08	3.14e-05
NABA_CORE_MATRISOME	222	0.01	Up	4.43e-08	4.19e-05

The camera function performs a competitive test to assess whether the genes in a given set are highly ranked in terms of differential expression relative to genes that are not in the set. It uses **limma**'s linear model framework, taking both the design matrix and contrast matrix (if present) and accommodates the observational-level weights from *voom* in the testing procedure. After adjusting the variance of the resulting gene set test statistic by a variance inflation factor, that depends on the gene-wise correlation (which can be estimated from the data or specified by the user) and the size of the set, a *p*-value is returned and adjusted for multiple testing. Here we set `inter.gene.cor` to 0.01 for all gene sets which produces a less conservative test that performs well in many practical situations.

This experiment is the RNA-seq equivalent of Lim *et al.* (2010)²⁰, who used Illumina microarrays to profile the same sorted cell populations, so it is reassuring to see the gene signatures from this earlier publication coming up at the top of the list for each contrast. We make a barcodeplot of the Lim *et al.* (2010) Mature Luminal gene sets (Up and Down) in the LP versus ML contrast. Note that these sets go in the opposite direction in our dataset due to our parameterization which compares LP against ML rather than the other way around (if the contrast were reversed, the directions would be consistent).

```
barcodeplot(efit$t[,3], index=idx$LIM_MAMMARY_LUMINAL_MATURE_UP,
            index2=idx$LIM_MAMMARY_LUMINAL_MATURE_DN, main="LPvsML")
```

Software availability

This RNA-seq workflow makes use of various packages available from version 3.4 of the Bioconductor project, running on R²¹ version 3.3.0 or higher. Besides the packages highlighted in this article (**limma**, **Glimma** and **edgeR**) it requires a number of other software packages, including **gplots**²² and **RColorBrewer** and the gene annotation package **Mus.musculus**. This document was compiled using **knitr**²³⁻²⁵. Version numbers for all packages used are shown below. Code to perform this analysis is available from the Supplementary Materials website at <http://bioinf.wehi.edu.au/folders/limmaWorkflow/> and as a Bioconductor workflow from <http://www.bioconductor.org/help/workflows/>.

```
sessionInfo()

## R version 3.3.0 (2016-05-03)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS release 6.4 (Final)
##
## locale:
##  [1] LC_CTYPE=en_AU.UTF-8  LC_NUMERIC=C           LC_TIME=en_AU.UTF-8
##  [4] LC_COLLATE=en_AU.UTF-8 LC_MONETARY=en_AU.UTF-8 LC_MESSAGES=en_AU.UTF-8
##  [7] LC_PAPER=en_AU.UTF-8  LC_NAME=C              LC_ADDRESS=C
## [10] LC_TELEPHONE=C        LC_MEASUREMENT=en_AU.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel stats4      stats      graphics grDevices utils      datasets methods
## [9] base
##
## other attached packages:
##  [1] gplots_3.0.1              Glimma_1.1.1
##  [3] RColorBrewer_1.1-2        Mus.musculus_1.3.1
##  [5] TxDb.Mmusculus.UCSC.mm10.knownGene_3.2.2 org.Mm.eg.db_3.3.0
##  [7] GO.db_3.3.0               OrganismDbi_1.14.0
##  [9] GenomicFeatures_1.24.0    GenomicRanges_1.24.0
## [11] GenomeInfoDb_1.8.1        AnnotationDbi_1.34.3
## [13] IRanges_2.6.0             S4Vectors_0.10.1
## [15] Biobase_2.32.0            BiocGenerics_0.18.0
## [17] edgeR_3.14.0              limma_3.28.5
##
## loaded via a namespace (and not attached):
##  [1] locfit_1.5-9.1            Rcpp_0.12.5             lattice_0.20-33
##  [4] gtools_3.5.0             Rsamtools_1.24.0        Biostrings_2.40.1
##  [7] plyr_1.8.3               chron_2.3-47            acepack_1.3-3.3
## [10] RSQLite_1.0.0            evaluate_0.9            ggplot2_2.1.0
## [13] highr_0.5.1              BiocInstaller_1.23.4    zlibbioc_1.18.0
## [16] gdata_2.17.0             annotate_1.50.0         data.table_1.9.6
## [19] R.utils_2.3.0            R.oo_1.20.0            rpart_4.1-10
## [22] Matrix_1.2-6             splines_3.3.0          BiocParallel_1.6.2
## [25] geneplotter_1.50.0       stringr_1.0.0          foreign_0.8-66
## [28] RCurl_1.95-4.8           biomaRt_2.28.0         munsell_0.4.3
## [31] rtracklayer_1.32.0       nnet_7.3-12            SummarizedExperiment_1.2.2
## [34] gridExtra_2.2.1          Hmisc_3.17-4           XML_3.98-1.4
## [37] GenomicAlignments_1.8.0 bitops_1.0-6           R.methodsS3_1.7.1
## [40] grid_3.3.0              RBGL_1.48.0            xtable_1.8-2
## [43] gtable_0.2.0            DBI_0.4-1              magrittr_1.5
## [46] formatR_1.3             scales_0.4.0           KernSmooth_2.23-15
## [49] graph_1.50.0            stringi_1.0-1          XVector_0.12.0
## [52] genefilter_1.54.2       latticeExtra_0.6-28    Formula_1.2-1
## [55] tools_3.3.0            DESeq2_1.12.2          survival_2.39-3
## [58] colorspace_1.2-6        cluster_2.0.4          caTools_1.17.1
## [61] knitr_1.12.3
```

Author contributions

All authors were involved in writing and contributing code for the article.

Competing interests

The authors declare that they have no competing interests.

Grant information

This worked was funded by the National Health and Medical Research Council (NHMRC) (Fellowship GNT1058892 and Program GNT1054618 to GKS, Project GNT1050661 to MER and GKS and Fellowship GNT1104924 to MER), Victorian State Government Operational Infrastructure Support and Australian Government NHMRC IRIISS.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

We thank Dr Julie Sheridan for generating this dataset and for advice on its analysis.

References

- Robinson MD, McCarthy DJ, Smyth GK: **edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.** *Bioinformatics*. 2010; **26**(1): 139–140.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Ritchie ME, Phipson B, Wu D, *et al.*: **limma powers differential expression analyses for RNA-seq and microarray studies.** *Nucleic Acids Res*. 2015; **43**(7): e47.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor.** *Nat Methods*. 2015; **12**(2): 115–121.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Su S, Ritchie ME: **Glimma: Interactive HTML graphics for RNA-seq data.** 2016; R package version 1.1.1.
- Sheridan JM, Ritchie ME, Best SA, *et al.*: **A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*.** *BMC Cancer*. 2015; **15**(1): 221.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Liao Y, Smyth GK, Shi W: **The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote.** *Nucleic Acids Res*. 2013; **41**(10): e108.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Liao Y, Smyth GK, Shi W: **featureCounts: an efficient general-purpose program for assigning sequence reads to genomic features.** *Bioinformatics*. 2014; **30**(7): 923–30.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Bioconductor Core Team: **Mus.musculus: Annotation package for the Mus.musculus object.** 2016; R package version 1.3.1.
[Reference Source](#)
- Bioconductor Core Team: **Homo.sapiens: Annotation package for the Homo.sapiens object.** 2016; R package version 1.3.1.
[Reference Source](#)
- Durink S, Moreau Y, Kasprzyk A, *et al.*: **BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis.** *Bioinformatics*. 2005; **21**(16): 3439–40.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Durink S, Spellman PT, Birney E, *et al.*: **Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt.** *Nat Protoc*. 2009; **4**(8): 1184–91.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Robinson MD, Oshlack A: **A scaling normalization method for differential expression analysis of RNA-seq data.** *Genome Biol*. 2010; **11**(3): R25.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Law CW, Chen Y, Shi W, *et al.*: **Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts.** *Genome Biol*. 2014; **15**(2): R29.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Liu R, Holik AZ, Su S, *et al.*: **Why weight? Modelling sample and observational level variability improves power in RNA-seq analyses.** *Nucleic Acids Res*. 2015; **43**(15): e97.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Liu R, Chen K, Jansz N, *et al.*: **Transcriptional profiling of the epigenetic regulator Smc4d1.** *Genom Data*. 2015; **7**: 144–7.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Smyth GK: **Linear models and empirical bayes methods for assessing differential expression in microarray experiments.** *Stat Appl Genet Mol Biol*. 2004; **3**(1): Article3.
[PubMed Abstract](#) | [Publisher Full Text](#)
- McCarthy DJ, Smyth GK: **Testing significance relative to a fold-change threshold is a TREAT.** *Bioinformatics*. 2009; **25**(6): 765–71.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Wu D, Smyth GK: **Camera: a competitive gene set test accounting for inter-gene correlation.** *Nucleic Acids Res*. 2012; **40**(17): e133.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Subramanian A, Tamayo P, Mootha VK, *et al.*: **Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles.** *Proc Natl Acad Sci U S A*. 2005; **102**(43): 15545–50.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Lim E, Wu D, Pal B, *et al.*: **Transcriptome analyses of mouse and human mammary cell subpopulations reveal multiple conserved genes and pathways.** *Breast Cancer Res*. 2010; **12**(2): R21.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- R Development Core Team: **R: A language and environment for statistical computing.** R Foundation for Statistical Computing, Vienna, 2016.
[Reference Source](#)
- Warnes GR, Bolker B, Bonebakker L, *et al.*: **gplots: Various R Programming Tools for Plotting Data.** 2016; R package version 3.0.1.
[Reference Source](#)
- Xie Y: **knitr: A comprehensive tool for reproducible research in R.** In V. Stodden, F. Leisch, and R. D. Peng, editors, *Implementing Reproducible Computational Research*. Chapman and Hall/CRC, 2014; ISBN 9781466561595.
[Reference Source](#)
- Xie Y: **Dynamic Documents with R and knitr.** Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015; ISBN 9781498716963.
[Reference Source](#)
- Xie Y: **knitr: A General-Purpose Package for Dynamic Report Generation in R.** 2016; R package version 1.12.3.
[Reference Source](#)

Open Peer Review

Current Peer Review Status:   

Version 1

Reviewer Report 06 July 2016

<https://doi.org/10.5256/f1000research.9688.r14437>

© 2016 Maksimovic J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Jovana Maksimovic 

Bioinformatics research group, Murdoch Childrens Research Institute, Murdoch Children's Research Institute, Parkville, Vic, Australia

This workflow outlines a step-by-step differential expression analysis of a publicly available RNA-seq dataset using several well-established Bioconductor packages such as limma and edgeR and the novel Glimma package. The workflow describes in detail the key steps that are generally performed as part of most standard, count-based RNAseq analyses: pre-processing, exploratory data analysis, differential expression testing and pathway analysis. Each of the steps is carefully explained and broken down into a series of logical substeps, which also include R code and example output. Furthermore, this workflow is a good example of how multiple Bioconductor packages can be applied in a real-world RNAseq analysis.

Overall, I think this is a very clearly written and well-explained workflow that not only highlights the utility of combining Bioconductor packages for RNAseq analysis but also demystifies the process for those who may be interested in learning what an RNAseq analysis involves. Although some familiarity with R and statistical concepts would be advantageous, they are not necessary to be able to follow the logic behind the steps and what each step is trying to achieve.

I have tested the R code under R 3.3.0 and Bioconductor 3.0 and it ran without errors.

Minor comments:

- As this workflow will be of particular use for those who are very new to RNAseq analysis, I think it should be explained why 1 CPM is chosen as the cutoff for lowly expressed genes. In addition, it may be useful to show the relationship between CPM and raw counts and how the selected cutoff relates to numbers of reads.
- In the gene set testing section, it is worth mentioning that the Hallmark gene sets (also available from the Broad MSigDB) may be a good starting point for pathway analysis as they "summarise and represent specific well-defined, states or processes".
- Also, for those who are new to RNAseq analysis, it may be worth mentioning that there are other gene set testing methods available in limma e.g. roast etc. and a very brief

explanation of when the different methods are appropriate.

- General comment regarding Glimma plots: it would be handy to see the gene symbol (if available) as well as EntrezID in the title of the scatter plot next to the MD plot.

Competing Interests: Matthew E. Ritchie and I are both organisers of the Bioconductor Asia-Pacific Developer meeting, which is to be held in Brisbane in November 2016.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 30 Nov 2016

Charity Law, The Walter and Eliza Hall Institute of Medical Research, Parkville, Australia

Dear Jovana, thank-you for the helpful suggestions. We have added extra text to explain why a CPM value of 1 was chosen and how this value relates to raw counts. We have also given more detail on c2 and Hallmark gene sets, as well as mentioning roast and when one might chose to use roast over camera.

Competing Interests: None

Reviewer Report 06 July 2016

<https://doi.org/10.5256/f1000research.9688.r14439>

© 2016 Doyle M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Maria A. Doyle

Research Computing Facility, Peter MacCallum Cancer Centre, Melbourne, Vic, Australia

Editorial Note from F1000Research - 7th July 2016:

the status of this review was changed after publication from "Approved with Reservations" to "Approved". The referee's reservation was caused by using the current release (Bioconductor 3.3) as opposed to the development release (Bioconductor 3.4) as used in the article.

The following text has been removed:

My only reservation to approving the article is that I encountered these two errors when running the Glimma code in the article. I used R 3.3.0 and the most recent version of Glimma from Bioconductor (Glimma 1.0.0) but the sessionInfo() in the article shows Glimma 1.1.1 was used which may account for the errors:

1) With this line of code no plot was produced and it gave the error below:

```
glMDSPlot(lcpm, labels=paste(group, lane, sep="_"),
```

```
+ groups=x$samples[,c(2,5)], launch=TRUE)
```

Error in glScatter.default(points, xval = "dim1", yval = "dim2", xlab = "Dimension 1", :
group does not correspond to a column

2) With this line of code the table (shown in Fig 6.) didn't appear, just the 2 plots:

```
glMDPlot(tfit, coef=1, status=dt[,1], main=colnames(tfit)[1],
```

```
counts=x$counts, samples=colnames(x), anno=x$genes, groups=group,
```

```
id.column="ENTREZID", display.columns=c("SYMBOL", "ENTREZID"),
```

```
search.by="SYMBOL", launch=FALSE)
```

This article is a very nice description (nicely written, nicely explained) of an RNA-seq workflow, starting from gene counts and proceeding through a standard differential expression analysis, using 3 complementary Bioconductor packages edgeR, limma & Glimma. It explains clearly how to perform the steps of a differential expression analysis and also how to generate useful plots for visualising the data e.g. MDSplot, MDplot, barcodeplot, heatmap etc.

Two of the packages, edgeR and limma, are well established while Glimma is the new kid on the block. It's a tool along the lines of the web-based Degust (<http://www.vicbioinformatics.com/de gust/>), in that it enables interactive exploration of RNA-seq data. It would seem to be a very useful addition to R RNA-seq workflows as the interactive html plots it can generate (MDS, MD etc) make exploring the results easier (especially by non-R savvy collaborators) and it could help save an analyst's time and effort through not having to reproduce static plots e.g. to highlight different genes. With the Glimma MDplot I like that you can search for a gene and see a plot of the log-cpm counts for that gene in the samples.

It was great to be able to try out the workflow really easily by downloading the data file linked to in the paper (from GEO), with no processing required other than unzipping the files, and the code in the article all worked, giving the same results shown in the paper with the exception of two errors described below.

Minor comments:

The article says the workflow is available from Bioconductor here

<http://www.bioconductor.org/help/workflows/> but it doesn't seem to be there, however it is available at the other location mentioned <http://bioinf.wehi.edu.au/folders/limmaWorkflow/>

The section title "Removing genes that are not expressed" suggests only genes with no expression are removed, however the paragraph then explains that genes lowly expressed are also removed so for greater clarity maybe that could be changed to something like "Removing genes that are not sufficiently expressed".

In the section "Organising gene annotations" I think it's good the authors point out to check for duplicated genes, however in this case the duplication appears to be due to extracting the TXCHROM column (as some genes are reported as being present on more than one chromosome) but the TXCHROM information is not used in this workflow and if the TXCHROM column is omitted then there are no duplicates so it might be worth mentioning that.

With this bit "Differential expression analyses look at gene expression differences between conditions, rather than comparing expression across multiple genes or drawing conclusions on absolute levels of expression. In other words, gene lengths remain constant for comparisons of interest and any observed differences are a result of changes in condition rather than changes in gene length." Wouldn't one caveat to this be if there was a significant change in the length of the isoform(s) expressed from a gene (e.g. from expression of a long isoform to a short isoform) as then the assumption of no change in gene length would no longer be valid.

In the online version Fig.7 is in the middle of a code block could it be moved below.

Figs 5-8 have bold headings but Figs 1-4 don't is that an error.

Could perhaps modify title to reflect the order in which the tools are used - edgeR, limma & Glimma.

It would be nice for consistency if the colours in the Glimma interactive MDS plots matched the colours used in the static MDS plots in Fig 3.

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Response 06 Jul 2016

Maria A. Doyle, Peter MacCallum Cancer Centre, Melbourne, Australia

I've just requested that the status of this article be changed to Approved as the code works fine with the development release of Bioconductor 3.4 but could it be made clearer in the article that 3.4 must be used for the code to work as shown.

Competing Interests: No competing interests were disclosed.

Author Response 30 Nov 2016

Charity Law, The Walter and Eliza Hall Institute of Medical Research, Parkville, Australia

Dear Maria, we have made changes according to your suggestions, including a change in the section title to "Removing genes that are lowly expressed", and we have added comments to point out that our analysis assumes no differential isoform usage. We have also simplified our example in the "Organising gene annotations" section.

Thanks for pointing out the inconsistencies in the figure captions. They have now been addressed.

Indeed, it would be nice to make the colours consistent between the interactive and static MDS plots. We like the colours used in the static plot and can have the interactive version of

the plot updated once color specification becomes available to the MDS plotting function in Glimma.

Competing Interests: None

Reviewer Report 23 June 2016

<https://doi.org/10.5256/f1000research.9688.r14436>

© 2016 MacDonald J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



James W. MacDonald 

Department of Environmental and Occupational Health Sciences, University of Washington, Seattle, WA, USA

In this manuscript the authors perform a step-by-step analysis of a public RNA-Seq data set, using the Bioconductor packages edgeR, limma, and Glimma. The analysis proceeds through four stages (data preparation, preprocessing, univariate analyses, and gene set testing), with each stage broken up into several discrete steps. Each step is clearly explained and is accompanied by R code and output, so the reader can follow along if desired.

The target audience for this paper appears to be someone with passing familiarity with both R and statistics, without requiring expertise in either. Most of the code should be easy to understand, and any complex sections are clearly explained in the text. Similarly, the authors outline the analytic choices they make (e.g., filtering out unexpressed genes, specifying model coefficients, etc) at a more accessible level.

While both edgeR and limma are well known, popular Bioconductor packages, Glimma is a new package that was released in April 2016. This package uses the d3.js JavaScript library to generate interactive HTML documents that can be viewed locally, rather than needing to be accessed from a server running R (as, say a shiny app requires). This is an exciting development, and is unfortunately not as compelling as it could be, if the Glimma plots were part of the HTML version of the manuscript rather than provided as links.

This is a well written paper, and is a useful contribution to the literature; while each package has either an extensive user's guide or vignette, by necessity these documents pertain only to the package at hand. Most RNA-Seq analyses require a combination of multiple packages to complete, and this paper provides a clear example.

Major comments

None.

Minor comments

In the section 'Organising gene annotations', the code used to subset the one-to-many mappings is needlessly complex. Simply doing something like

```
genes <- genes[!duplicated(genes[,1]),]
```

will accomplish the same thing, in a more straightforward way.

In the section 'Removing heteroscedasticity from count data', the authors state:

"When operating on a DGEList-object, voom converts raw counts to log-CPM values by automatically extracting library sizes and normalisation factors stored in the object. For a matrix of counts, the method of normalisation can be specified within voom using the normalize.method (by default no normalisation is performed)."

This is confusing; the authors already showed in an earlier section ('Normalising gene expression distributions') that converting to log-CPM using TMM normalization factors will do a shift-normalization (which is what voom will do in this instance). The normalize.method argument to voom specifies **additional** normalization methods that can be applied to the matrix of TMM normalized log-CPM values.

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 30 Nov 2016

Charity Law, The Walter and Eliza Hall Institute of Medical Research, Parkville, Australia

Dear James, thank-you for your comments. We have considered your suggestions and made changes to simplify the section on 'Organising gene annotations' and to clarify that voom offers additional normalisation ontop of the TMM-normalisation that is carried out.

Competing Interests: None

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research