RNA seq Analysis

Abhilash Mohan

April 10, 2019

Contents

1 The libraries required for the analysis in R **1**

1.1 Interpreting Read Counts . . . 1

1.1.1 Data Preprocessing and cleaning up . . . 2

1.1.2 Creating Groups for analysis . . . 4

1.2 Normalization . . . 4

1.2.1 What are counts per million reads? . . . 5

1.2.2 Creating a DGE object after first round of normalization . . . 8

1.2.3 Boxplot before normalization . . . 9

1.2.4 MDS plots . . . 10

1.2.5 Plotting Heatmap using highest variable genes . . . 11

1.2.6 Normalization for compositional bias . . . 13

1.2.7 Controling for Dispersion (Biological coefficient of variation) . . . 14

1.2.8 Box plot after normalization . . . 14

1.3 Differential Gene Expression . . . 16

1.3.1 Multiple Correction adjusted p-value . . . 17

1.3.2 Annotate the expression data and saving . . . 18

1.3.3 Creating Heatmap of DEGs . . . 19

1 The libraries required for the analysis in R

```
In [1]: library(edgeR)
        library(limma)
        library(Glimma)
        library(gplots)
        library(org.Hs.eg.db)
        library(RColorBrewer)
        library(dplyr)
        library(topGO)
        library(pathview)
        library(AnnotationDbi)
```

```
Loading required package: limma

Attaching package: 'gplots'

The following object is masked from 'package:stats':

    lowess

<<snipped>>
```

```
In [2]: countdata <- read.delim("SRP002628_ReadCounts.txt", sep=',',stringsAsFactors = FALSE)
```

```
In [3]: head(countdata)
```

Gene_ID	C11	C15	C19	C23	N11	N15	N19	N23
NR_075077	0	5	0	4	8	2	2	6
NM_001276352	0	5	0	4	8	2	2	6
NM_001276351	0	5	0	4	8	2	2	6
NM_000299	1220	5980	5089	2792	5223	9731	4365	4755
NM_001005337	1220	5980	5089	2792	5223	9731	4365	4755
NM_012102	7436	15741	13205	14024	15995	14659	9167	12504

1.1 Interpreting Read Counts

For almost all experiments, biological replicates of each condition must be used. Resulting count files therefore will look something like this, where the number of reads aligning to each annotated

	Sample 1	Sample 2	Sample 3
Gene A	5	3	8
Gene B	17	23	42
Gene C	10	13	27
Gene D	752	615	1203
Gene E	1507	1225	2455

gene are aggregated

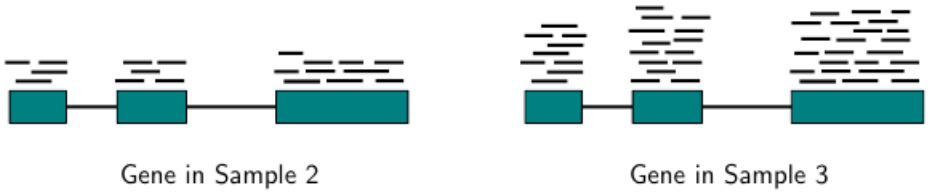
In the above example we can see that Gene E has about twice as many reads aligned to it as Gene D. This could mean:

1. Gene E is expressed with twice as many transcripts as Gene D

2. Both genes are expressed with the same number of transcripts but Gene E is twice as long as Gene D and produces twice as many fragments. At same expression level, a long transcript will have more reads than a shorter transcript

This implies number of reads are not equal to expression levels. Also we can see that genes in the sample 3 has about twice as many reads aligned to it as sample 2.

3. Differences in sequencing depth



The difference in counts between genes are likely to have been caused by some combination of all three reasons. The preprocessing of reads attempts to remove these biases.

1.1.1 Data Preprocessing and cleaning up

```
In [4]: countdata <- aggregate(countdata[-1], countdata[1], mean)
 #making sure the transcripts are all added up
        countdata$GeneID <- gsub("\\..*","",countdata$Gene_ID) # replacing any transcript IDs
        rownames(countdata) <- countdata[,1]
        #converting the processed GeneID columns to row.names
```

```
In [5]: head(countdata)
```

	Gene_ID	C11	C15	C19	C23	N11	N15	N19	N23	GeneID
NM_000014	NM_000014	4422	14216	8885	17031	8162	4811	12536	8273	NM_000014
NM_000015	NM_000015	3	0	7	2	0	9	2	6	NM_000015
NM_000016	NM_000016	1063	1192	1608	1345	1118	951	943	1120	NM_000016
NM_000017	NM_000017	164	424	463	507	603	692	494	653	NM_000017
NM_000018	NM_000018	5193	12982	11382	11716	10030	14180	9379	13316	NM_000018
NM_000019	NM_000019	654	1103	1106	1184	743	497	569	844	NM_000019

```
In [6]: annotations <-AnnotationDbi::select(org.Hs.eg.db, keys = row.names(countdata)
,column = c("SYMBOL","GENENAME"), keytype = "REFSEQ", multiVals = "first")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
In [7]: head(annotations)
        countdata <- merge(countdata, annotations, by.x=0, by.y="REFSEQ")

        head(countdata)
```

REFSEQ	SYMBOL	GENENAME
NM_000014	A2M	alpha-2-macroglobulin
NM_000015	NAT2	N-acetyltransferase 2
NM_000016	ACADM	acyl-CoA dehydrogenase medium chain
NM_000017	ACADS	acyl-CoA dehydrogenase short chain
NM_000018	ACADVL	acyl-CoA dehydrogenase very long chain
NM_000019	ACAT1	acetyl-CoA acetyltransferase 1

Row.names	Gene_ID	C11	C15	C19	C23	N11	N15	N19	N23	GeneID
NM_000014	NM_000014	4422	14216	8885	17031	8162	4811	12536	8273	NM_000014
NM_000015	NM_000015	3	0	7	2	0	9	2	6	NM_000015
NM_000016	NM_000016	1063	1192	1608	1345	1118	951	943	1120	NM_000016
NM_000017	NM_000017	164	424	463	507	603	692	494	653	NM_000017
NM_000018	NM_000018	5193	12982	11382	11716	10030	14180	9379	13316	NM_000018
NM_000019	NM_000019	654	1103	1106	1184	743	497	569	844	NM_000019

```
In [8]: countdata2<-countdata[,c(12,3:10)]
#subsetting and extracting only what we require at the moment
```

```
In [9]: head(countdata2)
```

SYMBOL	C11	C15	C19	C23	N11	N15	N19	N23
A2M	4422	14216	8885	17031	8162	4811	12536	8273
NAT2	3	0	7	2	0	9	2	6
ACADM	1063	1192	1608	1345	1118	951	943	1120
ACADS	164	424	463	507	603	692	494	653
ACADVL	5193	12982	11382	11716	10030	14180	9379	13316
ACAT1	654	1103	1106	1184	743	497	569	844

```
In [10]: countdata2 <- aggregate(countdata2[-1], countdata2[1], mean)
         rownames(countdata2) <- countdata2[,1]
         head(countdata2)
         countdata2$SYMBOL <- NULL
         head(countdata2)
```

	SYMBOL	C11	C15	C19	C23	N11	N15	N19	N23
A1BG	A1BG	22	82.0	57.0	84.0	127.0	223	115	190.0
A1BG-AS1	A1BG-AS1	25	78.0	111.0	102.0	135.0	177	116	164.0
A1CF	A1CF	6	0.0	3.0	33.0	5.0	12	3	5.0
A2M	A2M	4422	14216.0	8885.0	17031.0	8162.0	4811	12536	8273.0
A2M-AS1	A2M-AS1	222	329.0	310.0	562.0	234.0	114	311	254.0
A2ML1	A2ML1	58	58.5	68.5	48.5	44.5	62	43	46.5

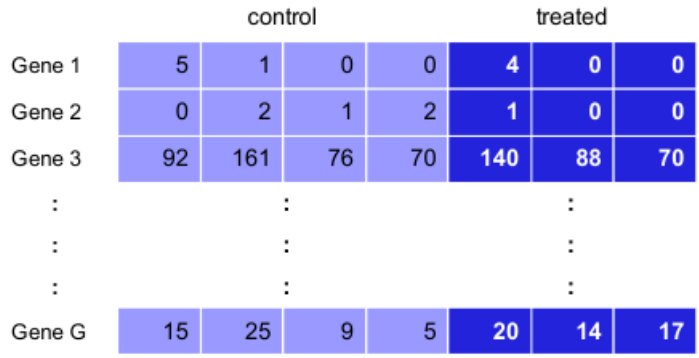
	C11	C15	C19	C23	N11	N15	N19	N23
A1BG	22	82.0	57.0	84.0	127.0	223	115	190.0
A1BG-AS1	25	78.0	111.0	102.0	135.0	177	116	164.0
A1CF	6	0.0	3.0	33.0	5.0	12	3	5.0
A2M	4422	14216.0	8885.0	17031.0	8162.0	4811	12536	8273.0
A2M-AS1	222	329.0	310.0	562.0	234.0	114	311	254.0
A2ML1	58	58.5	68.5	48.5	44.5	62	43	46.5

1.1.2 Creating Groups for analysis

```
In [11]: group <- c(rep("Tumour",4),rep("Control",4))
```

1.2 Normalization

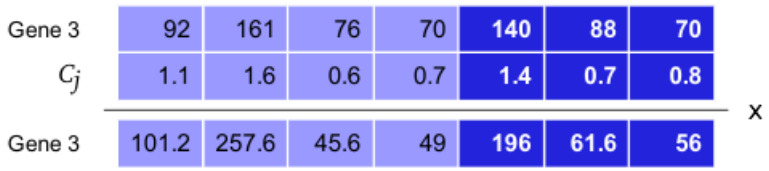
There are many normalisation methods. The normalisation method used in many of the popular software like edgeR and DESeq are broadly called Global normalization methods. In global procedures, a single factor C_j is used to scale the counts for each sample j.



Correction multiplicative factor

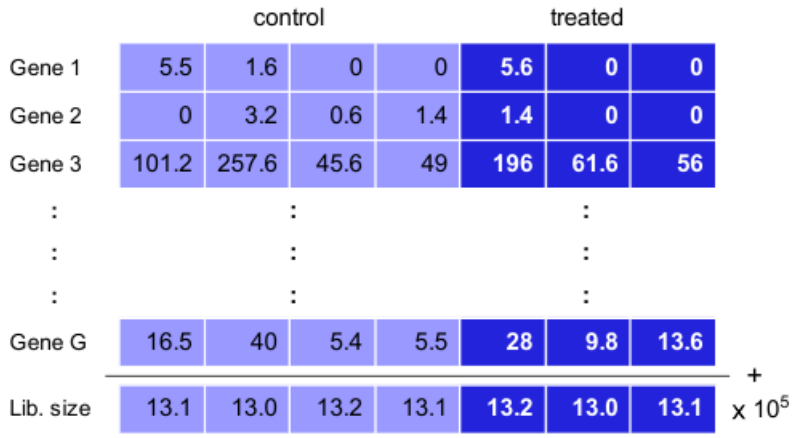
norm2.png

Column multiplication by factor C_j



norm3.png

The purpose of the size factors C_j is to render comparable the counts of different samples, even if these samples have been sequenced with different depths



- K_{gj} : observed count for gene g in sample (library) j
- G : number of genes
- $D_j = \sum_{g=1}^{G} K_{gj}$: total number of reads for sample j
- N : number of samples in the experiment
- C_j : normalization factor associated with sample j

Genes with very low counts across all libraries provide little evidence for differential expression and they interfere with some of the statistical approximations that are used later in the pipeline. They also add to the multiple testing burden when estimating false discovery rates, reducing power to detect differentially expressed genes. These genes should be filtered out prior to further analysis.

There are a few ways to filter out lowly expressed genes. When there are biological replicates in each group, in this case we have a sample size of 2 in each group, we favour filtering on a minimum counts per million threshold present in at least 2 samples. Two represents the smallest sample size for each group in our experiment. In this dataset, we choose to retain genes if they are expressed at a counts-per-million (CPM) above 0.5 in at least two samples.

1.2.1 What are counts per million reads?

The motivation of this method is scaling to library size as a form of normalization. This makes intuitive sense, given it is expected that sequencing a sample to half the depth will give, on average, half the number of reads mapping to each gene.

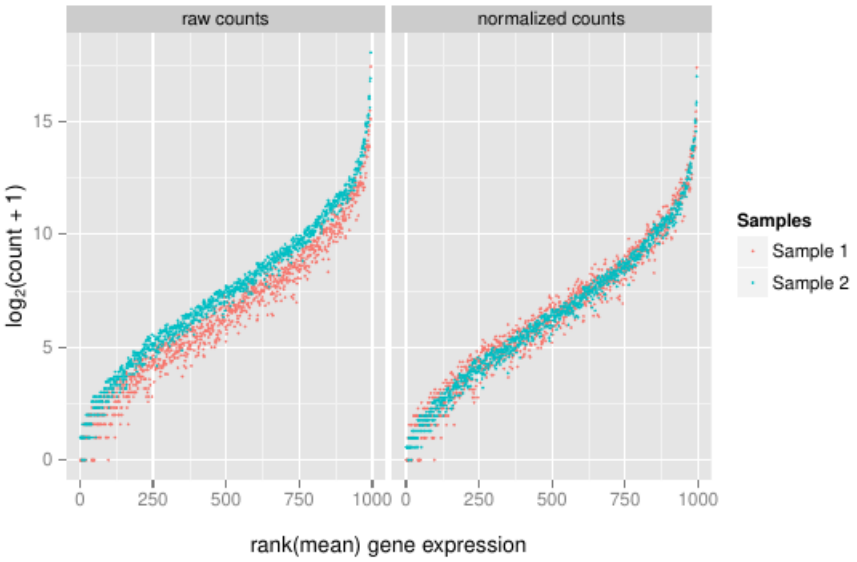
The main assumptions of this method are that read counts are proportional to expression level and sequencing depth. The methodology adpoted is to divide transcript read count by total number of reads and rescale the factors to counts per million

$$C_j = 10^6 / D_j$$

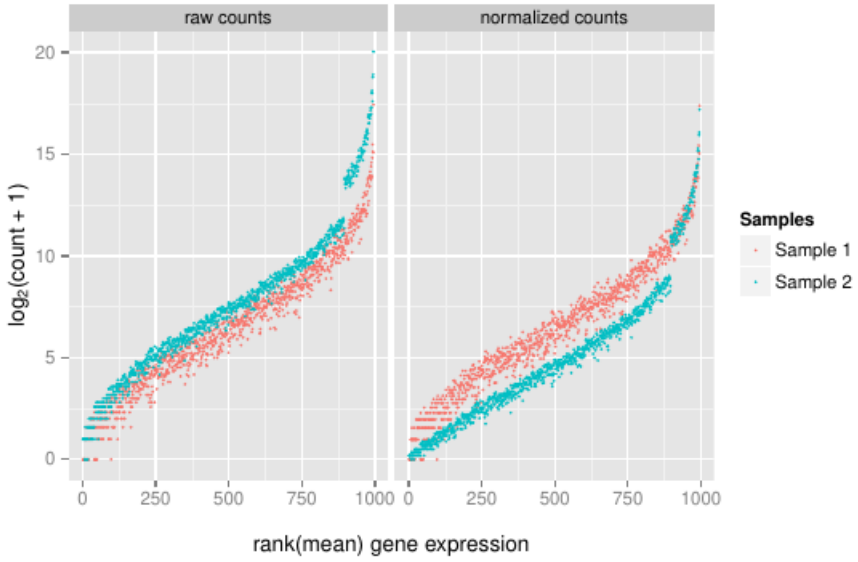
The primary disadvantage of this method is that, if a few number of genes are unique to, or highly expressed in, one experimental condition, the sequencing "space" available for the remaining genes in that sample is decreased. If this is not adjusted for, thissampling artifact can force the differential expression analysis to be skewed towards one experimental condition

When will this matter? Suppose that genes in the Sample 2 has about twice as many reads aligned to it as Sample 1. Then, the total read count normalization would adjust Sample 1 by a

factor of 2 (shown below). Now suppose that Sample 2 contains a small set of highly expressed



genes. Under this scenario, the small fraction of highly expressed genes will skew the counts of lowly expressed genes.




```
In [12]: myCPM <- cpm(countdata2) #CPM
         head(myCPM)
```

	C11	C15	C19	C23	N11	N15	N19
A1BG	1.3901069	3.237054	2.2733975	3.564670	5.5292886	9.9373098	6.4383361
A1BG-AS1	1.5796669	3.079149	4.4271425	4.328528	5.8775903	7.8874611	6.4943217
A1CF	0.3791201	0.000000	0.1196525	1.400406	0.2176885	0.5347431	0.1679566
A2M	279.4114813	561.194680	354.3708194	722.736852	355.3547549	214.3874325	701.834623
A2M-AS1	14.0274421	12.987693	12.3640916	23.849340	10.1878232	5.0800597	17.4115003
A2ML1	3.6648272	2.309362	2.7320654	2.058173	1.9374279	2.7628395	2.4073779

```
In [13]: thresh <- myCPM > 0.75 #setting a CPM threshold
         head(thresh)
```

	C11	C15	C19	C23	N11	N15	N19	N23
A1BG	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
A1BG-AS1	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
A1CF	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
A2M	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
A2M-AS1	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
A2ML1	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

```
In [14]: keep <- rowSums(thresh) >= 2
         head(keep)
```

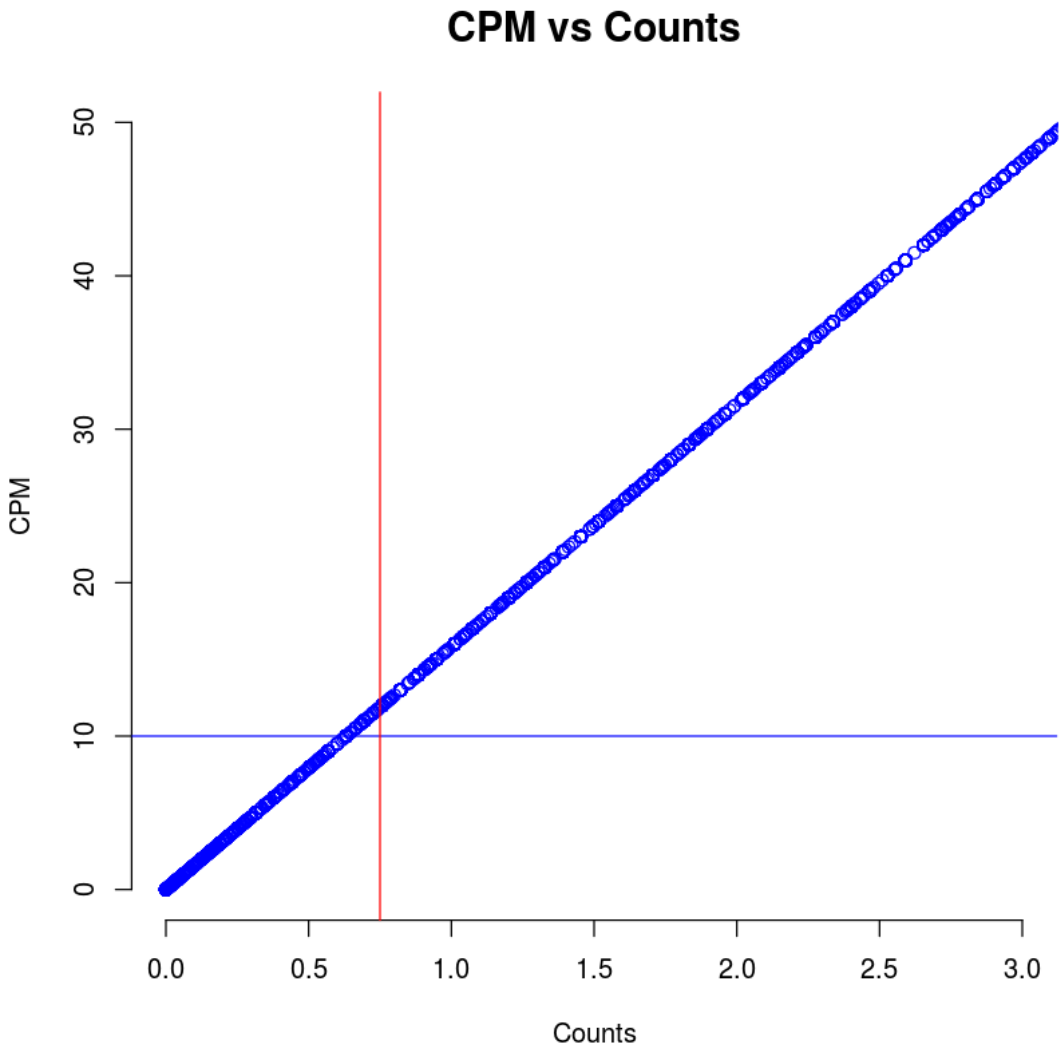
A1BG TRUE **A1BG-AS1** TRUE **A1CF** FALSE **A2M** TRUE **A2M-AS1** TRUE **A2ML1** TRUE

Now lets take a subset of genes that we have to keep Subset the rows of countdata to keep the more highly expressed genes

```
In [15]: countdata2.keep <- countdata2[keep,]
         head(countdata2.keep)
```

	C11	C15	C19	C23	N11	N15	N19	N23
A1BG	22	82.0	57.0	84.0	127.0	223	115	190.0
A1BG-AS1	25	78.0	111.0	102.0	135.0	177	116	164.0
A2M	4422	14216.0	8885.0	17031.0	8162.0	4811	12536	8273.0
A2M-AS1	222	329.0	310.0	562.0	234.0	114	311	254.0
A2ML1	58	58.5	68.5	48.5	44.5	62	43	46.5
A3GALT2	30	87.0	34.0	59.0	38.0	90	42	58.0

```
In [16]: plot(myCPM[,1],countdata2[,1], xlab="Counts", ylab="CPM", ylim=c(0,50),xlim=c(0,3)
, main="CPM vs Counts",
              pch=1, cex.main=1.5, frame.plot=FALSE , col="blue")
         abline(v=0.75, h=10,col=c("blue", "red"))
```

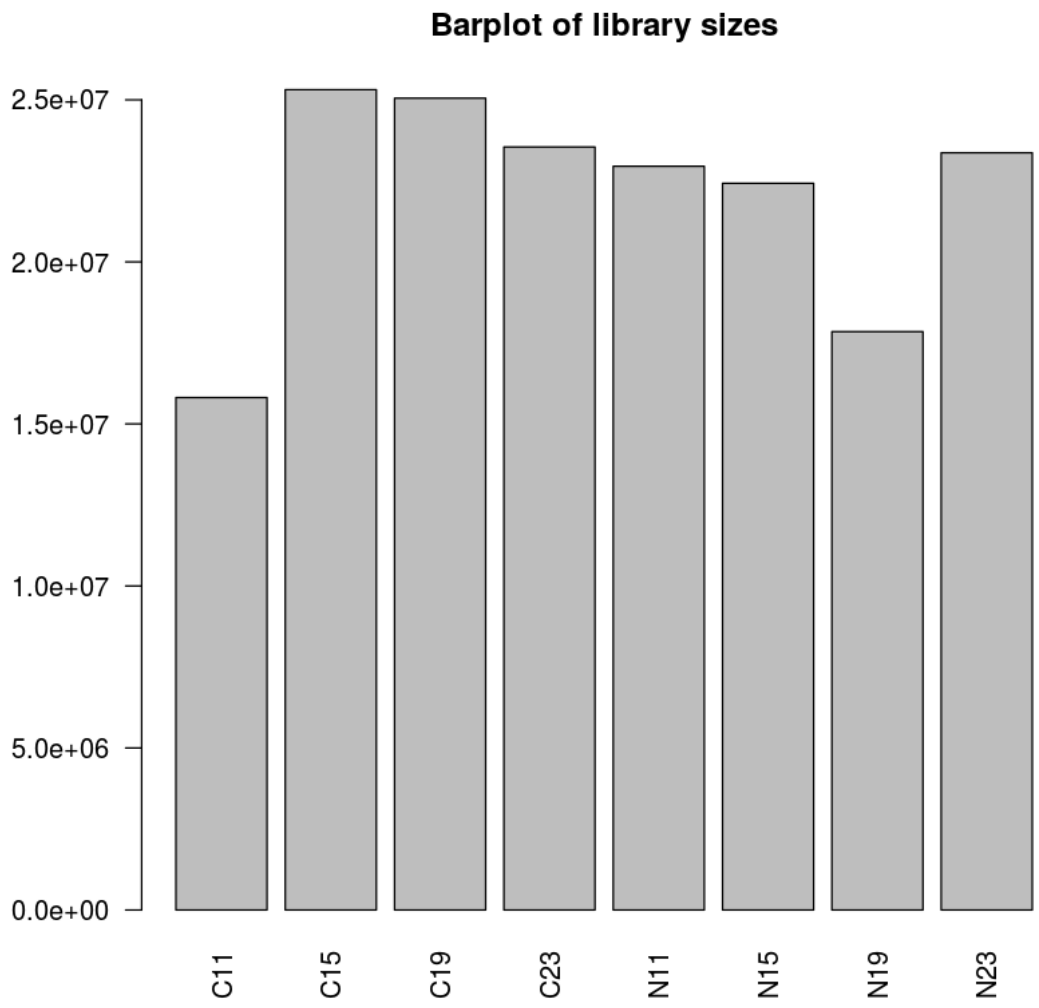


A CPM of 0.75 is used as it corresponds to a count of 10-15 for the library sizes in this data set. If the count is any smaller, it is considered to be very low, indicating that the associated gene is not expressed in that sample. A requirement for expression in two or more libraries is used as each group contains two replicates. This ensures that a gene will be retained if it is only expressed in one group. Smaller CPM thresholds are usually appropriate for larger libraries. As a general rule, a good threshold can be chosen by identifying the CPM that corresponds to a count of 10, which in this case is about 0.75. You should filter with CPMs rather than filtering on the counts directly, as the latter does not account for differences in library sizes between samples.

1.2.2 Creating a DGE object after first round of normalization

```
In [17]: dgeObj <- DGEList(counts = countdata2.keep, group = group)
```

```
In [18]: barplot(dgeObj$samples$lib.size,names=colnames(dgeObj),las=2)
         title("Barplot of library sizes")
```



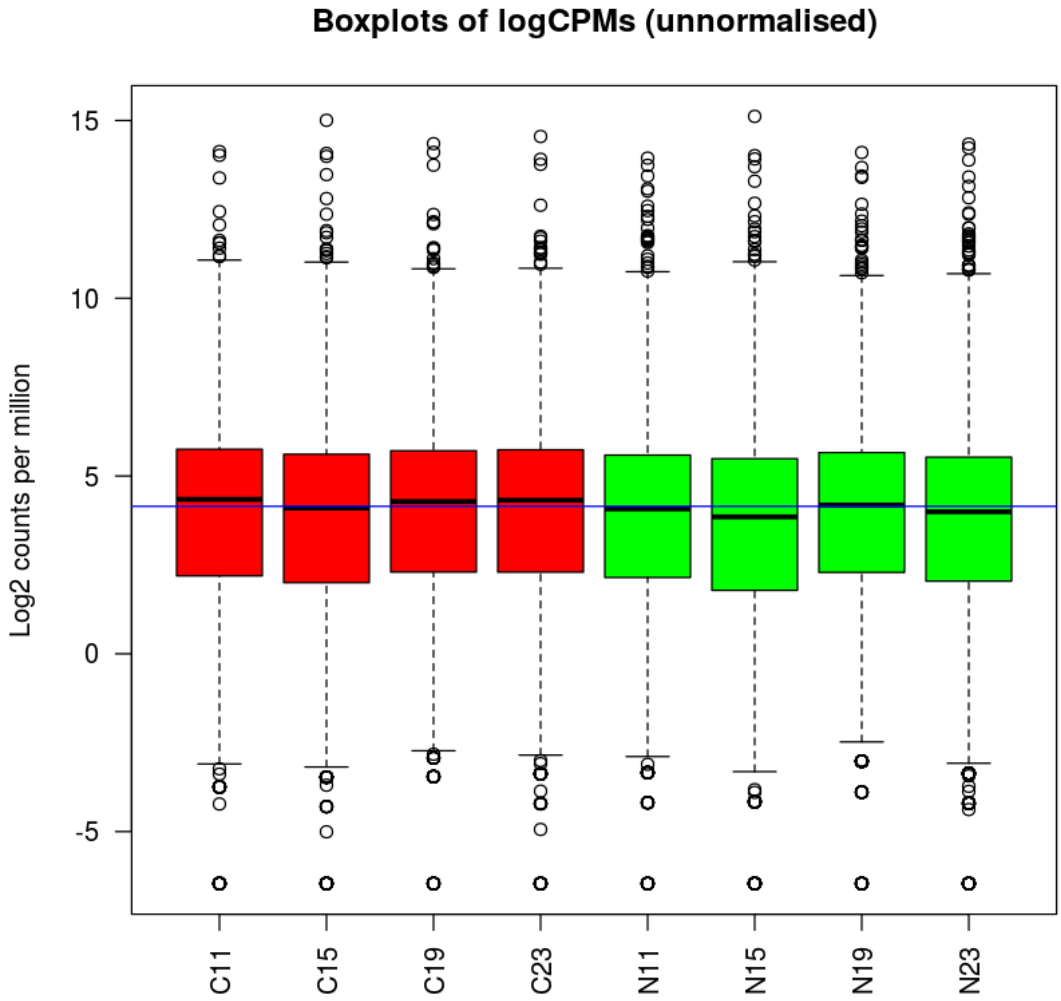

```
In [19]: pdf("library_size_samples.pdf") #to save the file
         barplot(dgeObj$samples$lib.size,names=colnames(dgeObj),las=2)

         title("Barplot of library sizes")
         dev.off()
```

png: 2

1.2.3 Boxplot before normalization

```
In [20]: logcounts <- cpm(dgeObj,log=TRUE)
         col.cell <- c("red","red","red","red","green","green","green","green")
```

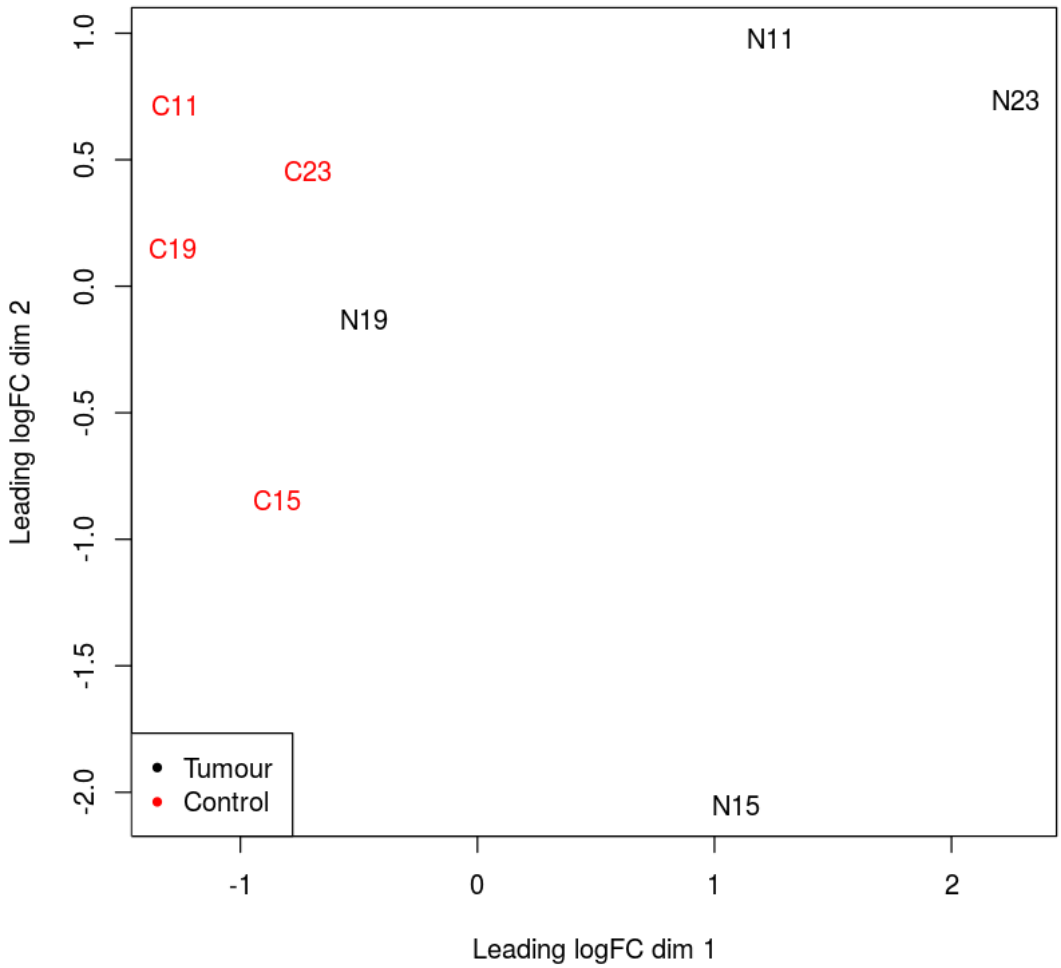
```
In [21]: boxplot(logcounts, xlab="", ylab="Log2 counts per million",col=col.cell,las=2)
         abline(h=median(logcounts),col="blue")
         title("Boxplots of logCPMs (unnormalised)")
```



1.2.4 MDS plots

By far, one of the most important plots we make when we analyse RNA-Seq data are MDSplots. An MDSplot is a visualisation of a principle components analysis, which determines the greatest sources of variation in the data. A principle components analysis is an example of an unsupervised analysis, where we don't need to specify the groups. If your experiment is well controlled and has worked well, what we hope to see is that the greatest sources of variation in the data are the treatments/groups we are interested in. It is also an incredibly useful tool for quality control and checking for outliers. We can use the plotMDS function to create the MDS plot.

```
In [22]: plotMDS(dgeObj,col=as.numeric(dgeObj$samples$group))
         legend("bottomleft", as.character(unique(dgeObj$samples$group)), col=1:3, pch=20)
```



1.2.5 Plotting Heatmap using highest variable genes

```
In [23]: logcounts <- cpm(dgeObj,log=TRUE)
         # We estimate the variance for each row in the logcounts matrix
         var_genes <- apply(logcounts, 1, var)
         head(var_genes)
         # Get the gene names for the top 500 most variable genes
         select_var <- names(sort(var_genes, decreasing=TRUE))[1:500]
         head(select_var)
         # Subset logcounts matrix
```

```
highly_variable_lcpm <- logcounts[select_var,]
dim(highly_variable_lcpm)
head(highly_variable_lcpm)
```

A1BG 0.923880455166182 **A1BG-AS1** 0.577605825616083 **A2M** 0.398289622073014 **A2M-AS1** 0.419086207913979 **A2ML1** 0.0945575884926016 **A3GALT2** 0.267029876642157

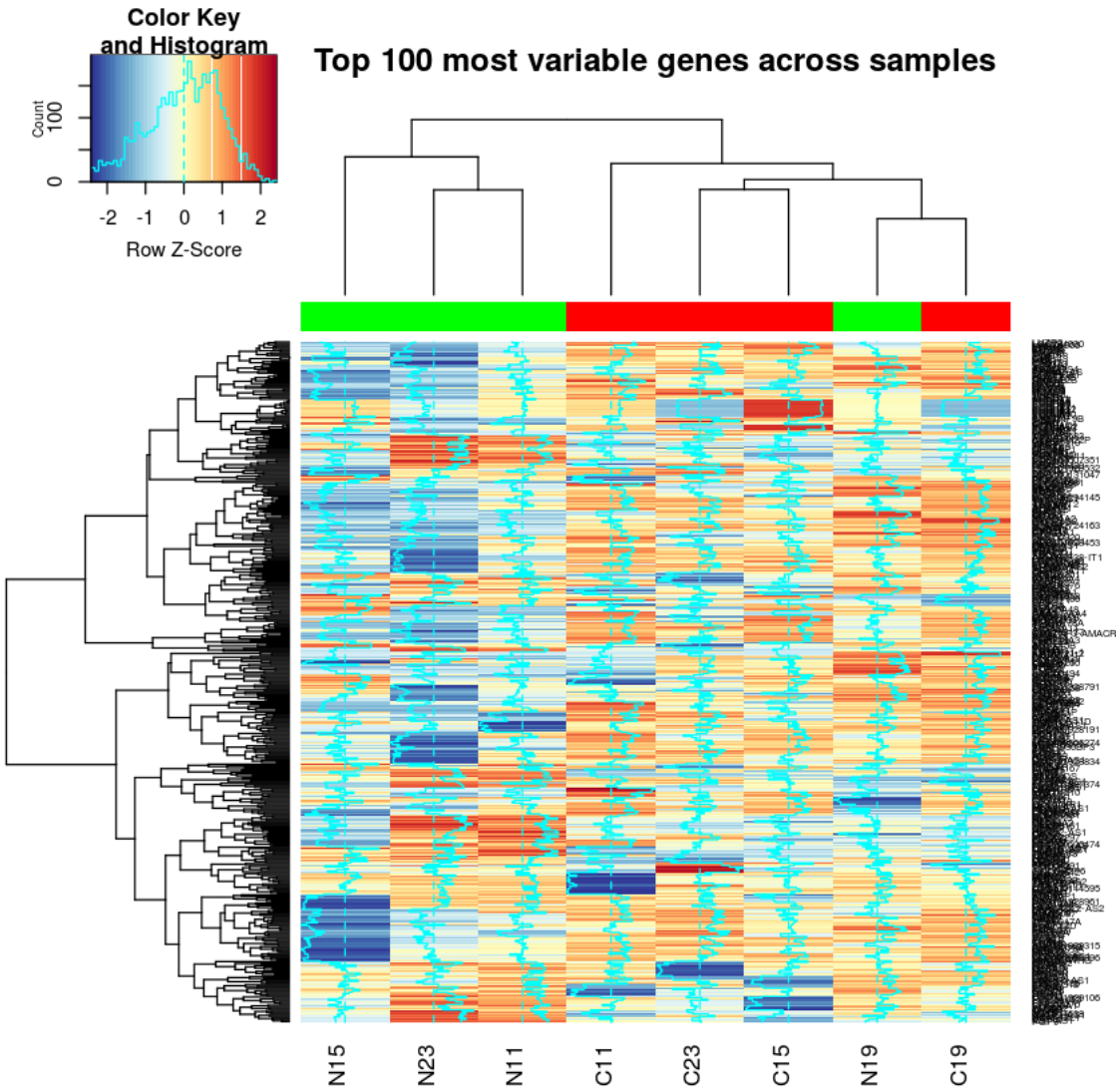
1. 'CRISP3' 2. 'RAX' 3. 'CR2' 4. 'SEMG1' 5. 'SEMG2' 6. 'CHIT1'

1. 500 2. 8

	C11	C15	C19	C23	N11	N15	N19	N23
CRISP3	1.3808069	4.2548944	8.462906	-0.0804716	2.459861	-6.461983	5.619120	-3.3668254
RAX	-6.4619829	-3.4682935	-6.461983	-6.4619829	1.824083	4.362344	-6.461983	-3.3668254
CR2	-6.4619829	0.7403049	1.266899	0.9108503	-6.461983	-6.461983	3.720113	-4.2071688
SEMG1	1.6944475	-0.8251965	-6.461983	1.7330218	5.844023	-1.128906	-6.461983	0.8204739
SEMG2	0.8335246	-6.4619829	-6.461983	-0.3661380	4.497136	-1.441914	-6.461983	-0.0062818
CHIT1	4.2609847	1.1270298	4.436097	-1.1973583	3.066521	-4.160094	2.445274	-6.4619829

In [24]:
```
## Get some nicer colours
mypalette <- brewer.pal(11,"RdYlBu")
morecols <- colorRampPalette(mypalette)


# Plot the heatmap
heatmap.2(highly_variable_lcpm,
          col=rev(morecols(50)),
          trace="column",
          main="Top 100 most variable genes across samples",
          ColSideColors=col.cell,scale="row")

# Save the heatmap
png(file="High_var_genes.heatmap.png")
heatmap.2(highly_variable_lcpm,col=rev(morecols(50)),trace="none",
          main="Top 100 most variable genes\nacross samples",ColSideColors=col.cell
          ,scale="row")
dev.off()
```

png: 2



1.2.6 Normalization for compositional bias

Normalization in the tutorial is performed using "upperquartile". "upperquartile" is the upper-quartile normalization method of Bullard et al (2010), in which the scale factors are calculated from the 75% quantile of the counts for each library, after removing transcripts which are zero in all libraries. This idea is generalized here to allow scaling by any quantile of the distributions.

```
In [41]: dgeObj <- calcNormFactors(dgeObj,method="upperquartile")
         summary(dgeObj)
         head(dgeObj$samples)
```

```
                  Length Class      Mode
counts            154288 -none-     numeric
```

```
samples                 3 data.frame list
common.dispersion       1 -none-     numeric
pseudo.counts      154288 -none-     numeric
pseudo.lib.size         1 -none-     numeric
AveLogCPM           19286 -none-     numeric
trend.method            1 -none-     character
trended.dispersion  19286 -none-     numeric
prior.df                1 -none-     numeric
prior.n                 1 -none-     numeric
tagwise.dispersion  19286 -none-     numeric
span                    1 -none-     numeric
```

	group	lib.size	norm.factors
C11	Tumour	15813043	1.0857351
C15	Tumour	25313627	0.9845101
C19	Tumour	25051207	1.0542876
C23	Tumour	23544397	1.0738806
N11	Control	22949593	0.9675078
N15	Control	22425609	0.9029666

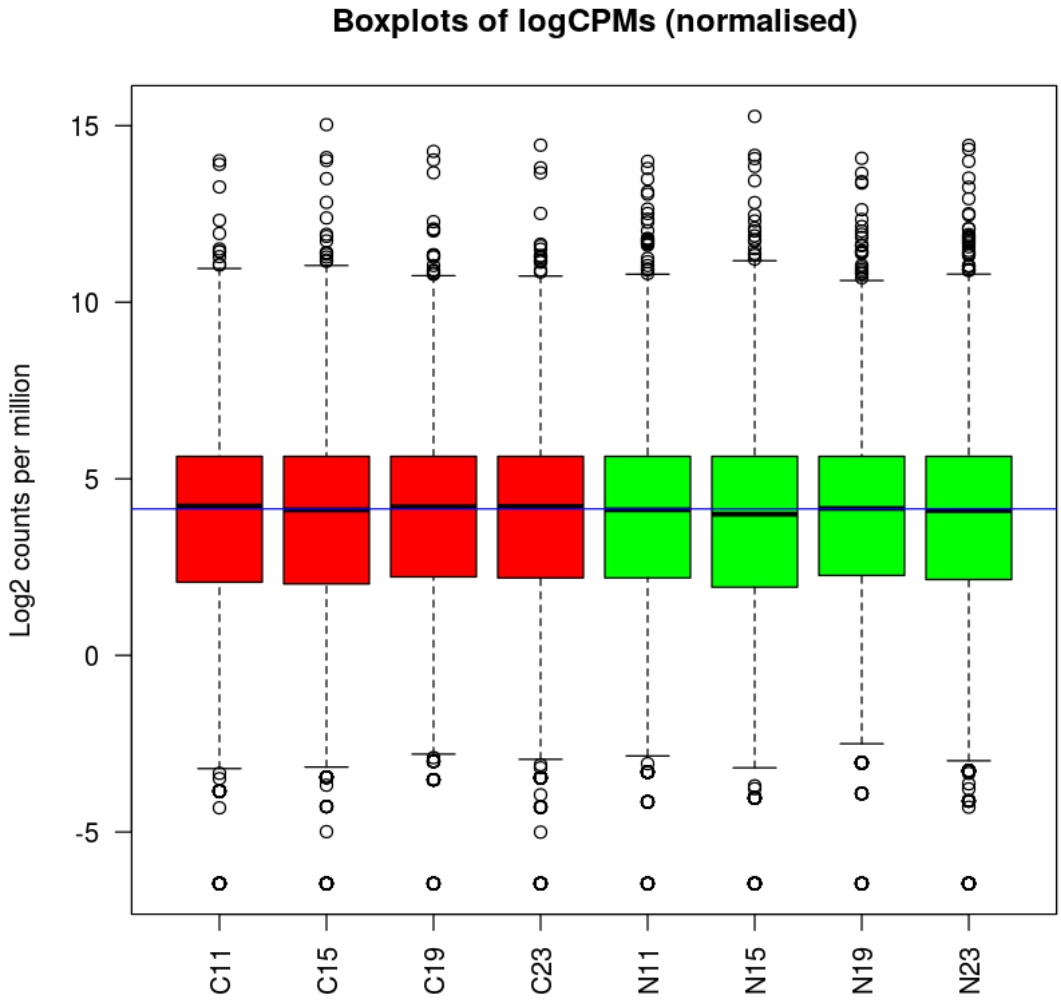
1.2.7 Controling for Dispersion (Biological coefficient of variation)

```
In [26]: dgeObj <- estimateCommonDisp(dgeObj, verbose=TRUE)
         dgeObj <- estimateGLMTrendedDisp(dgeObj,verbose=TRUE)
         dgeObj <- estimateTagwiseDisp(dgeObj,verbose=TRUE)
         #the BCV is used to either reduce or increase the dispersion value of a gene.
```

```
Disp = 0.11385 , BCV = 0.3374
Disp = 0.46496 , BCV = 0.6819
Disp = 0.31695 , BCV = 0.563
Disp = 0.3352 , BCV = 0.579
Disp = 0.3026 , BCV = 0.5501
Disp = 0.32293 , BCV = 0.5683
Disp = 0.30101 , BCV = 0.5486
<<snipped>>

Using interpolation to estimate tagwise dispersion.
```

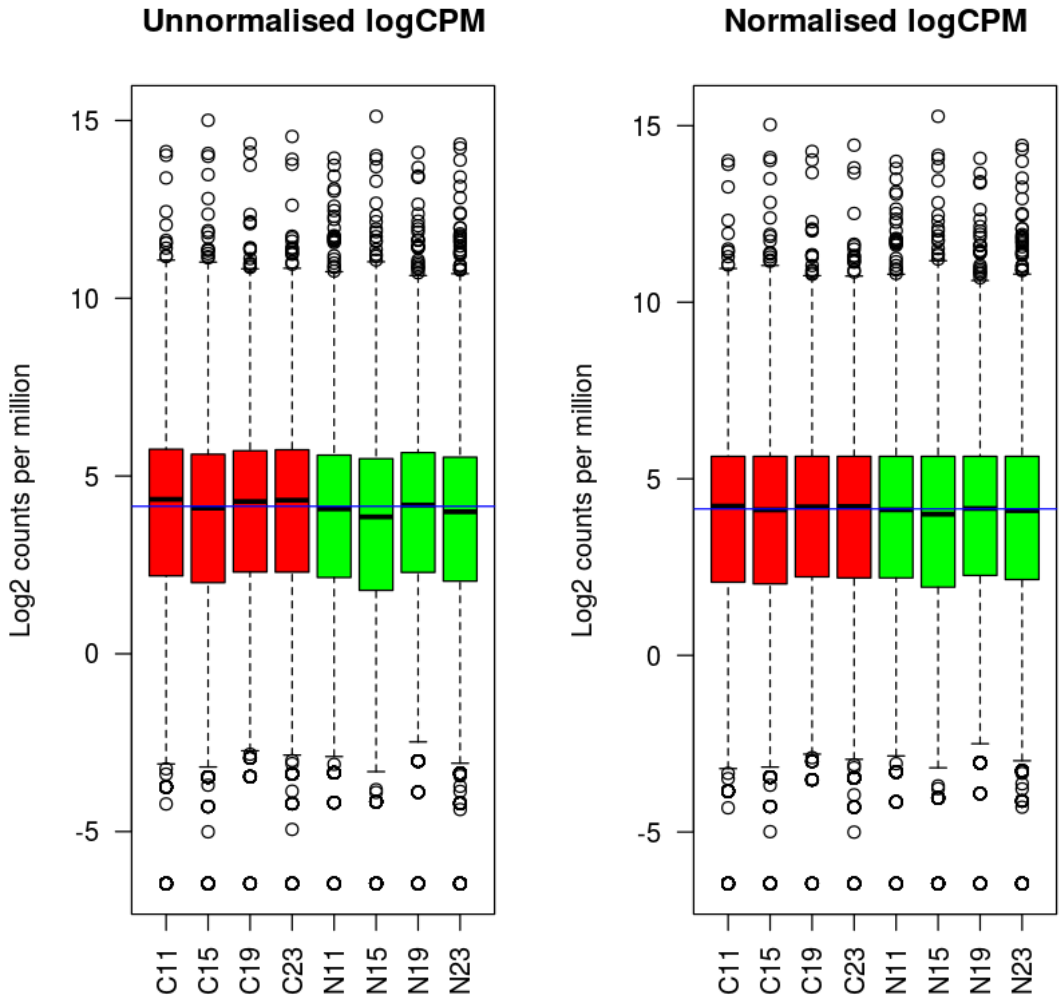
1.2.8 Box plot after normalization

```
In [27]: normlogcounts <- cpm(dgeObj,log=TRUE)
         boxplot(normlogcounts, xlab="", ylab="Log2 counts per million",col=col.cell,las=2)
         abline(h=median(normlogcounts),col="blue")
         title("Boxplots of logCPMs (normalised)")
```



Putting the box-plots side-by-side for comparison

```
In [28]: par(mfrow=c(1,2))
         boxplot(logcounts, xlab="", ylab="Log2 counts per million"
         ,col=col.cell,las=2,main="Unnormalised logCPM")
         ## Let's add a blue horizontal line that corresponds to the median logCPM
         abline(h=median(logcounts),col="blue")
         boxplot(normlogcounts, xlab="", ylab="Log2 counts per million"
         ,col=col.cell,las=2, main="Normalised logCPM")
         abline(h=median(normlogcounts),col="blue")
```



1.3 Differential Gene Expression

```
In [29]: dgeanalysistest <- exactTest(dgeObj)
         etp<-topTags(dgeanalysistest,n=nrow(dgeanalysistest$table))
         head(etp$table)
```

	logFC	logCPM	PValue	FDR
LUZP2	2.951459	4.787516	4.633271e-18	8.935726e-14
GLYATL1	2.383091	6.022541	5.355545e-17	5.164352e-13
FLNC	-2.591764	9.882777	2.007514e-16	1.290564e-12
SLC16A5	-2.191801	5.946399	3.412589e-15	1.645380e-11
SNCG	-2.729800	5.005593	1.536649e-14	5.927163e-11
FLNA	-2.253248	12.904126	5.723457e-14	1.839710e-10

1.3.1 Multiple Correction adjusted p-value

```
In [31]: dgeanalysistest$PValue_fdr <- p.adjust(method="fdr",p=dgeanalysistest$PValue)
         dgetable <- dgeanalysistest$table
         dgetable$PValue_fdr <- p.adjust(method="fdr",p=dgetable$PValue)
         head(dgetable)
```

	logFC	logCPM	PValue	PValue_fdr
A1BG	-1.65273618	2.410876	8.736476e-05	0.002185207
A1BG-AS1	-1.15641769	2.402622	4.166422e-03	0.030984749
A2M	0.13283856	8.779379	7.057349e-01	0.825047131
A2M-AS1	0.41878016	3.725692	2.499603e-01	0.431855443
A2ML1	0.09202135	1.357037	8.296284e-01	0.905906670
A3GALT2	-0.32687838	1.372881	4.842929e-01	0.655946077

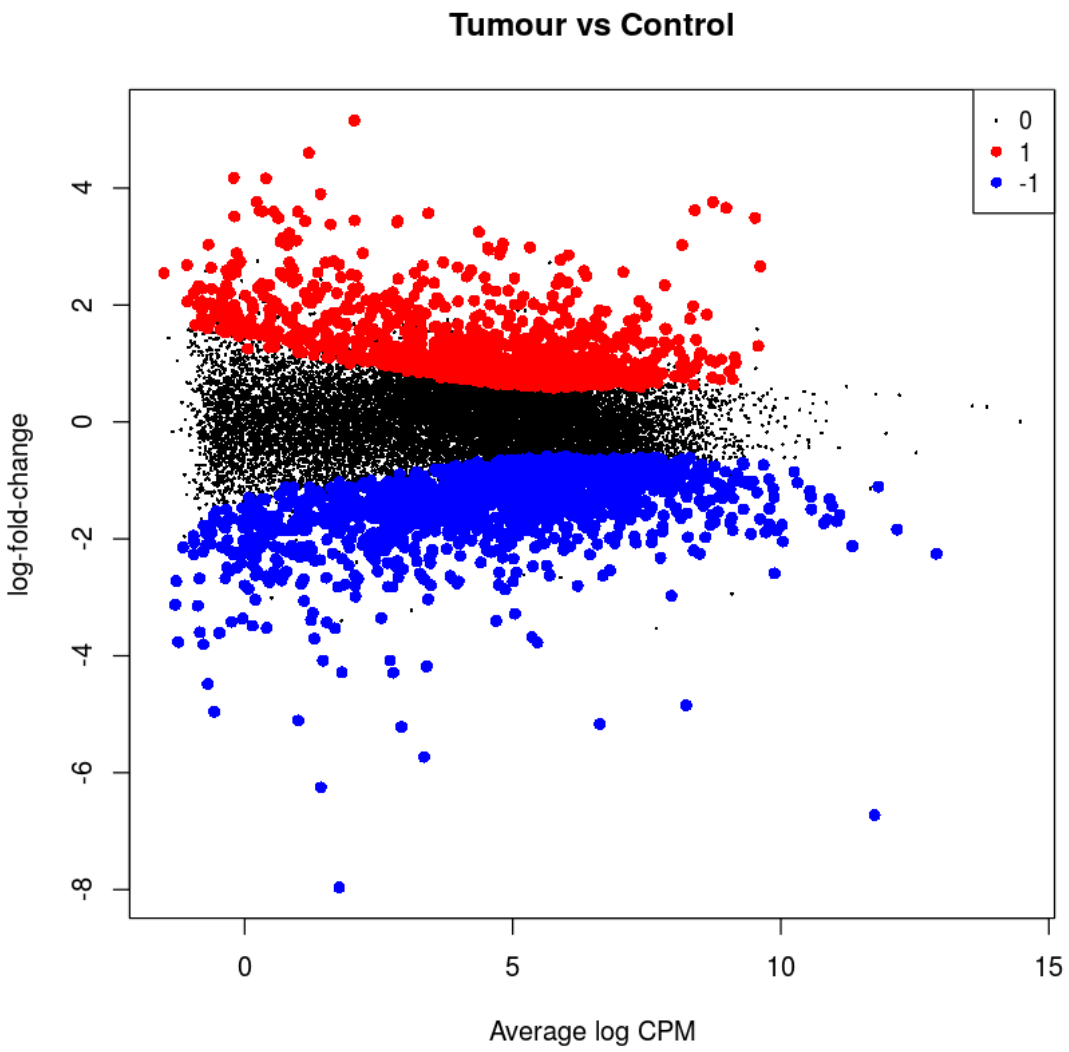
```
In [32]: table(dgetable$PValue_fdr<0.05) #5% False Discovery rate
         isde<-decideTestsDGE(dgeanalysistest,p.value=0.05)
         summary(isde)
         #extracting only the DEGs
         de_results <- as.data.frame(dgetable)
         fdr_cutoff <- 0.2
         de_results$de <- dgetable$PValue_fdr < fdr_cutoff
         head(de_results)
         write.table(de_results, file = "de-results.txt",
                     quote = FALSE, sep = "\t", col.names = NA)
                     #this will be used for gene annotation
```

```
FALSE  TRUE
16015  3271
```

```
       Control+Tumour
Down             1796
NotSig          16015
Up               1475
```

	logFC	logCPM	PValue	PValue_fdr	de
A1BG	-1.65273618	2.410876	8.736476e-05	0.002185207	TRUE
A1BG-AS1	-1.15641769	2.402622	4.166422e-03	0.030984749	TRUE
A2M	0.13283856	8.779379	7.057349e-01	0.825047131	FALSE
A2M-AS1	0.41878016	3.725692	2.499603e-01	0.431855443	FALSE
A2ML1	0.09202135	1.357037	8.296284e-01	0.905906670	FALSE
A3GALT2	-0.32687838	1.372881	4.842929e-01	0.655946077	FALSE

```
In [33]: plotMD(dgeanalysistest, status=isde, values=c(1,-1), col=c("red","blue"),
                legend="topright")
```



1.3.2 Annotate the expression data and saving

```
In [34]: ann <-  try(suppressWarnings(AnnotationDbi::select(org.Hs.eg.db,keys=rownames
(dgetable),columns=c("ENTREZID","GENENAME"), keytype='SYMBOL')))
```

```
'select()' returned 1:many mapping between keys and columns
```

```
In [35]: head(ann)
```

SYMBOL	ENTREZID	GENENAME
A1BG	1	alpha-1-B glycoprotein
A1BG-AS1	503538	A1BG antisense RNA 1
A2M	2	alpha-2-macroglobulin
A2M-AS1	144571	A2M antisense RNA 1 (head to head)
A2ML1	144568	alpha-2-macroglobulin like 1
A3GALT2	127550	alpha 1,3-galactosyltransferase 2

```
In [36]: diffexprwithannotation<-merge(etp$table,ann,by.x="row.names",by.y="SYMBOL")
 #merging tables
```

```
In [37]: write.table(diffexprwithannotation, file="DifferentiallyExpressedGenes.csv"
, sep='\t',quote=FALSE)
```

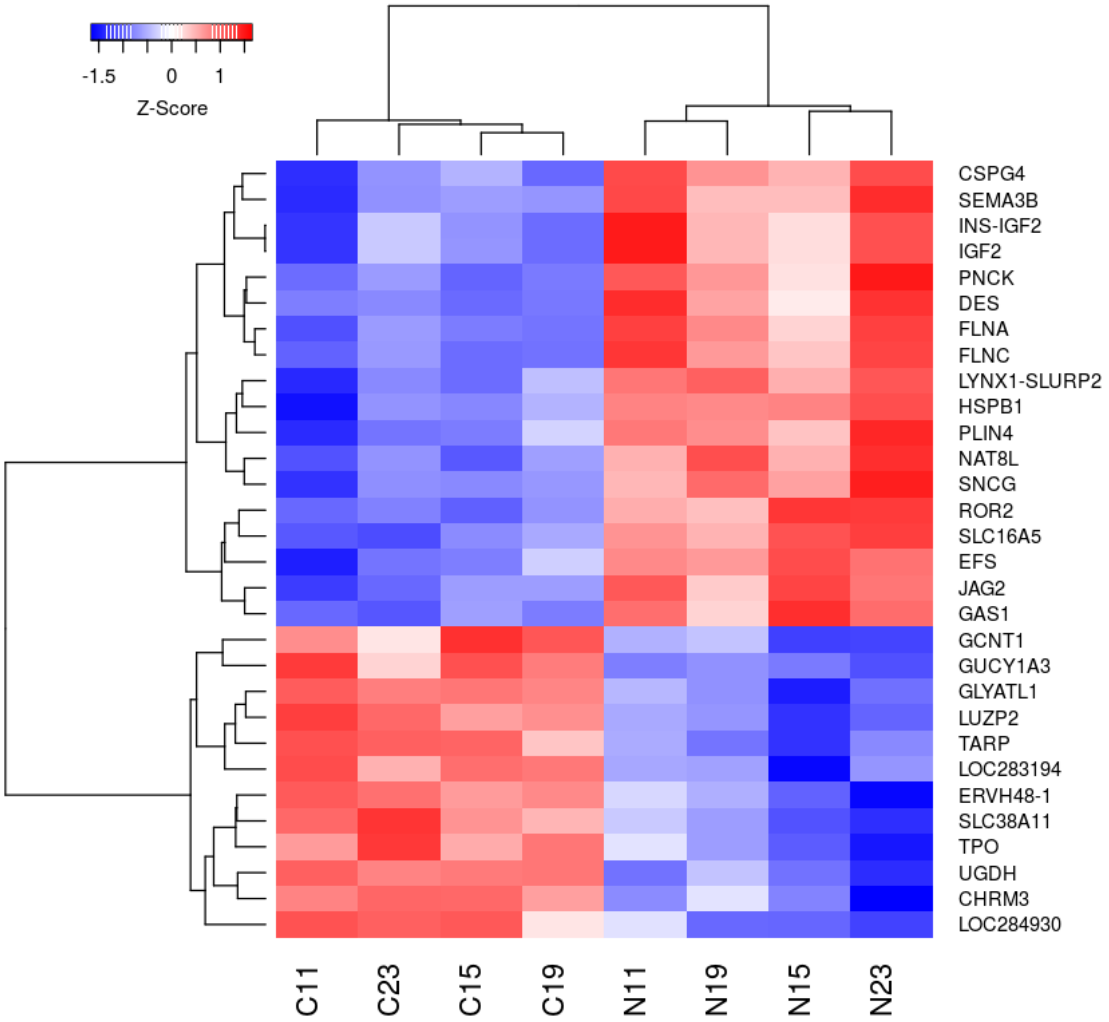
1.3.3 Creating Heatmap of DEGs

```
In [38]: getgeneorder <- order(dgetable$PValue_fdr)
         head(getgeneorder)
         newlogCPM <- normlogcounts[getgeneorder[1:30],]#top 30 DEGs will be plotted
         head(newlogCPM)
```

1. 9532 2. 6116 3. 5584 4. 15060 5. 15538 6. 5582

	C11	C15	C19	C23	N11	N15	N19	N23
LUZP2	6.130939	5.109347	5.288570	5.683359	3.129045	1.854542	2.917693	2.407410
GLYATL1	6.984329	6.756544	6.631154	6.671679	4.879490	3.525789	4.541572	4.270485
FLNC	7.873844	7.968307	8.018243	8.367465	11.144991	9.842670	10.244360	11.023670
SLC16A5	4.202955	4.605717	4.832087	4.105678	6.388928	6.892139	6.125496	7.043273
SNCG	2.344458	3.176274	3.311997	3.248470	5.070778	5.282653	5.817782	6.554064
FLNA	11.054731	11.402067	11.345330	11.642463	13.989881	12.820152	13.414294	13.990034

```
In [39]: coolmap(newlogCPM, margins=c(7,7), lhei=c(1,6), lwid=c(1,3))
```


-1.5
0
1
Z-Score
CSPG4
SEMA3B
INS-IGF2
IGF2
PNCK
DES
FLNA
FLNC
LYNX1-SLURP2
HSPB1
PLIN4
NAT8L
SNCG
ROR2
SLC16A5
EFS
JAG2
GAS1
GCNT1
GUCY1A3
GLYATL1
LUZP2
TARP
LOC283194
ERVH48-1
SLC38A11
TPO
UGDH
CHRM3
LOC284930
C11
C23
C15
C19
N11
N19
N15
N23