Microarray Analysis - Running ANOVA Using Limma for Time-Series Data

Varun Dwaraka

This tutorial caters to individuals who are would like to implement limma for differential gene expression analysis on time-series microarray datasets. Limma is very useful in conducting this analysis as it takes in RMA normalized datasets and identifies genes which are significant based on fitting the data to a linear model, and controlling for family wise error via an empirical bayes method, smoothing the data for stable analyses even within experiments with small number of arrays. For more information please read: http://nar.oxfordjournals.org/content/43/7/e47 (Ritchie et al. 2015).

The data that I will be using is from a time-series experiment detailing tail regeneration in stage 42 axolotls. All timepoints are in hours post amputation (hpa) with C0 relevant to time exactly at amputation.

Read in the dataset (CSV file)

Here we read the CSV file to create a data.frame. I use the read.table() function, but if you want to use the read.csv() function, I believe that works well as well. Here I just make sure that everything is set up so that the row names are the probe identifiers while the column names are ones that I have specified in the names variable.

NOTE: If you are inputting a .txt file, just specify that file with the .txt extension and change the sep=',' option to reflect tab (https://stat.ethz.ch/R-manual/R-devel/library/utils/html/write.table.html)

```
setwd("C:/Users/Varun Dwaraka/Documents/Tutorials-Rmd/")
wnt <- read.table("Wnt_data.csv",header=TRUE,sep=',')
rownames(wnt) <- wnt[,1]
wnt <- wnt[,-1]
wnt <- wnt[,c(60:69,11:19,30:59,1:10,20:29)]
names <-
c("C0_1","C0_2","C0_3","C0_4","C0_5","T0_1","T0_2","T0_3","T0_4","T0_5",

"C12_1","C12_2","C12_3","C12_4","T12_1","T12_2","T12_3","T12_4","T12_5",

"C24_1","C24_2","C24_3","C24_4","C24_5","T24_1","T24_2","T24_3","T24_4","T24_
5",

"C48_1","C48_2","C48_3","C48_4","C48_5","T48_1","T48_2","T48_3","T48_4","T48_
5",

"C72_1","C72_2","C72_3","C72_4","C72_5","T72_1","T72_2","T72_3","T72_4","T72_
5",
```

```
"C120_1","C120_2","C120_3","C120_4","C120_5","T120_1","T120_2","T120_3","T120
_4","T120_5",

"C168_1","C168_2","C168_3","C168_4","C168_5","T168_1","T168_2","T168_3","T168
_4","T168_5")
colnames(wnt)<-names
wnt_control <-
cbind(wnt[,1:5],wnt[,11:14],wnt[,20:24],wnt[,30:34],wnt[,40:44],wnt[,50:54],w
nt[,60:64])
head(wnt_control)
```

```
##                      C0_1      C0_2      C0_3      C0_4      C0_5
## AFFX-BioB-3_at   9.363175  9.630548  9.205536  9.816648  9.739097
## AFFX-BioB-5_at   9.078323  9.358224  9.055940  9.411005  9.395720
## AFFX-BioB-M_at   9.651438 10.035080  9.583186 10.133930 10.031660
## AFFX-BioC-3_at  10.703700 10.923690 10.585140 11.048790 10.983250
## AFFX-BioC-5_at  10.555990 10.673920 10.360250 10.829830 10.774490
## AFFX-BioDn-3_at 12.793760 12.971380 12.688530 13.082650 13.004070
##                     C12_1     C12_2     C12_3     C12_4     C24_1
## AFFX-BioB-3_at   9.310589  9.777759  9.737213  9.637977  9.086430
## AFFX-BioB-5_at   8.984932  9.366185  9.365263  9.347719  8.769448
## AFFX-BioB-M_at   9.676533 10.109480 10.012150  9.954139  9.404696
## AFFX-BioC-3_at  10.560730 11.014500 10.893580 10.910040 10.418320
## AFFX-BioC-5_at  10.439860 10.806880 10.719830 10.705090 10.232130
## AFFX-BioDn-3_at 12.657840 13.031350 12.965450 12.955040 12.452320
##                     C24_2     C24_3     C24_4     C24_5     C48_1
## AFFX-BioB-3_at   9.435314  8.852466  9.666026  9.572707  9.358328
## AFFX-BioB-5_at   8.936174  8.537787  9.335927  9.523444  9.082760
## AFFX-BioB-M_at   9.795684  9.191022 10.002210  9.981407  9.704735
## AFFX-BioC-3_at  10.642550 10.312130 10.805060 10.914710 10.626610
## AFFX-BioC-5_at  10.399210 10.046260 10.627080 10.713880 10.370740
## AFFX-BioDn-3_at 12.722370 12.353600 12.724570 12.909240 12.811800
##                     C48_2     C48_3     C48_4     C48_5     C72_1
## AFFX-BioB-3_at   8.999336  9.512229  9.360888  9.001040  9.373201
## AFFX-BioB-5_at   8.793926  9.186524  9.278390  8.753245  9.098090
## AFFX-BioB-M_at   9.346910  9.811908  9.749407  9.357752  9.790163
## AFFX-BioC-3_at  10.417620 10.716490 10.771310 10.363540 10.684870
## AFFX-BioC-5_at  10.250900 10.583190 10.558950 10.122750 10.527970
## AFFX-BioDn-3_at 12.566980 12.825890 12.864630 12.419160 12.839750
##                     C72_2     C72_3     C72_4     C72_5    C120_1
## AFFX-BioB-3_at   9.005775  9.420070  9.393159  8.871319  8.938770
## AFFX-BioB-5_at   8.694841  8.963427  9.066567  8.590433  8.645839
## AFFX-BioB-M_at   9.260710  9.658865  9.692024  9.211786  9.214333
## AFFX-BioC-3_at  10.360520 10.633400 10.689590 10.250530 10.211340
## AFFX-BioC-5_at  10.125200 10.417870 10.427480  9.999205  9.997602
## AFFX-BioDn-3_at 12.478230 12.735750 12.754280 12.390720 12.318770
##                    C120_2    C120_3    C120_4    C120_5    C168_1
## AFFX-BioB-3_at   9.533326  8.839933  9.315142  9.325258  8.977916
## AFFX-BioB-5_at   9.151310  8.506100  8.907226  9.079068  8.654383
```

```
## AFFX-BioB-M_at   9.841401  9.105494  9.609594  9.662539  9.243385
## AFFX-BioC-3_at  10.756970 10.306310 10.539210 10.579860 10.299300
## AFFX-BioC-5_at  10.561730 10.032700 10.341420 10.409050 10.116540
## AFFX-BioDn-3_at 12.836470 12.407480 12.651830 12.756200 12.430000
##                    C168_2    C168_3    C168_4    C168_5
## AFFX-BioB-3_at   9.337351  8.846651  9.300838  9.233148
## AFFX-BioB-5_at   8.979328  8.574668  8.949969  8.983064
## AFFX-BioB-M_at   9.580468  9.187052  9.688663  9.551884
## AFFX-BioC-3_at  10.578230 10.274860 10.510640 10.514950
## AFFX-BioC-5_at  10.383260 10.029340 10.311340 10.343270
## AFFX-BioDn-3_at 12.715690 12.434330 12.742820 12.700620

nrow(wnt_control)

## [1] 20080

ncol(wnt_control)

## [1] 34
```

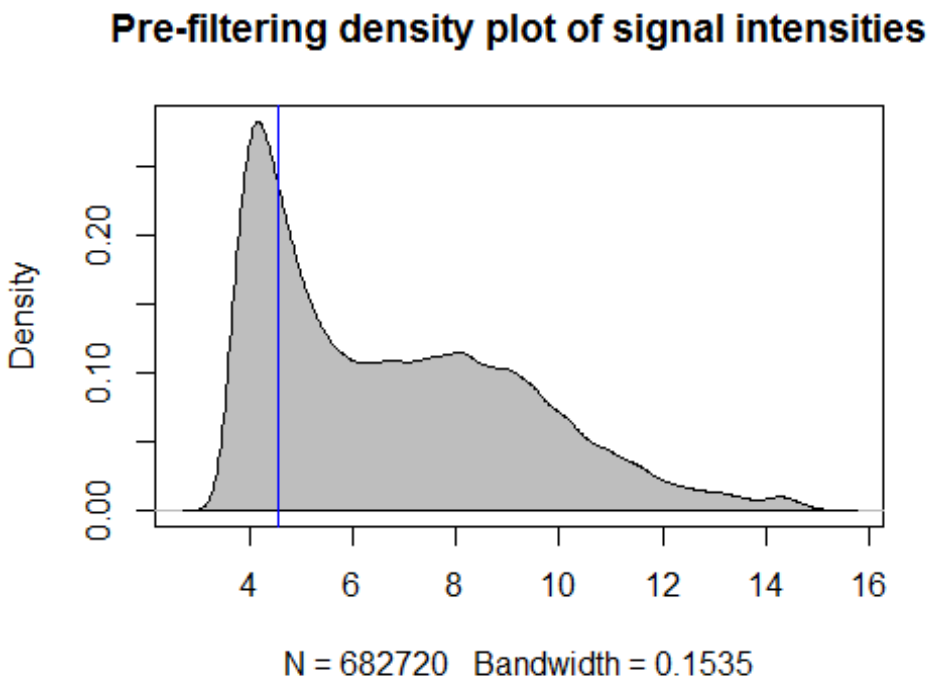
At this point you will notice that the matrix is set up so that the row names are probe ID's and the column headers are all on line 1. The overall matrix consists of 20,080 rows (or genes in the microarray dataset) and 34 samples (for each timepoint and its replicates).

Filtering out probes that are lowly expressed

Filtering version 1

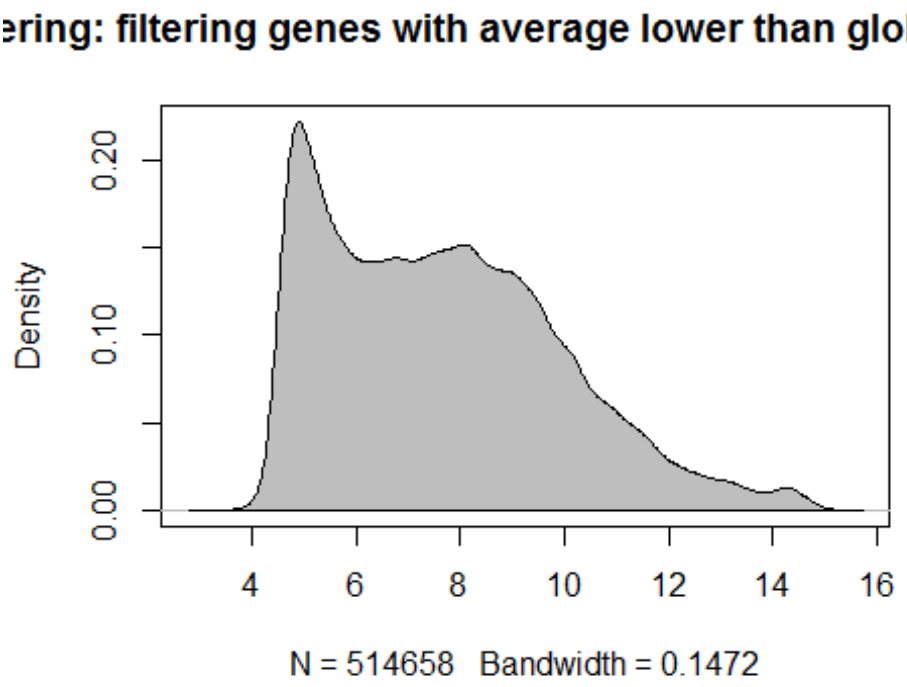
First look at the density of all RNA normalized expression values.

```
pre.filter <- density(as.matrix(wnt_control))
plot(pre.filter, "Pre-filtering density plot of signal intensities")
polygon(pre.filter,col="grey")
quantile(as.matrix(wnt_control))

##        0%       25%       50%       75%      100%
##  3.163384  4.563939  6.310062  8.589735 15.292720

globalfq <- quantile(as.matrix(wnt_control))[[2]]
abline(v=globalfq,col="blue")
```



Filter out samples which have a probe mean that is less than the first quartile value of the global data (represented by the blue line in the density plot). This is a method I use based on a previous study done in our lab. To show the effects of the filtering procedure, density plots are produced; the goal is to lessen the initial peak in the subtly bimodal distribution (which usually represents signal intensities that are below the detectable limit for microarrays and therefore not very confident).

```
wnt_filtered <- matrix(ncol=ncol(wnt_control),nrow=nrow(wnt_control))
rownames(wnt_filtered) <- rownames(wnt_control)
colnames(wnt_filtered) <- colnames(wnt_control)

for(i in 1:nrow(wnt_control)){
  if(mean(as.numeric(wnt_control[i,]))>=globalfq){
    wnt_filtered[i,]<-unlist(c(wnt_control[i,]))
  }
}
wnt_filtered <- na.omit(wnt_filtered)
plot(density(wnt_filtered),main="Post-filtering: filtering genes with average
lower than global first quartile")
polygon(density(wnt_filtered),col="grey")
```



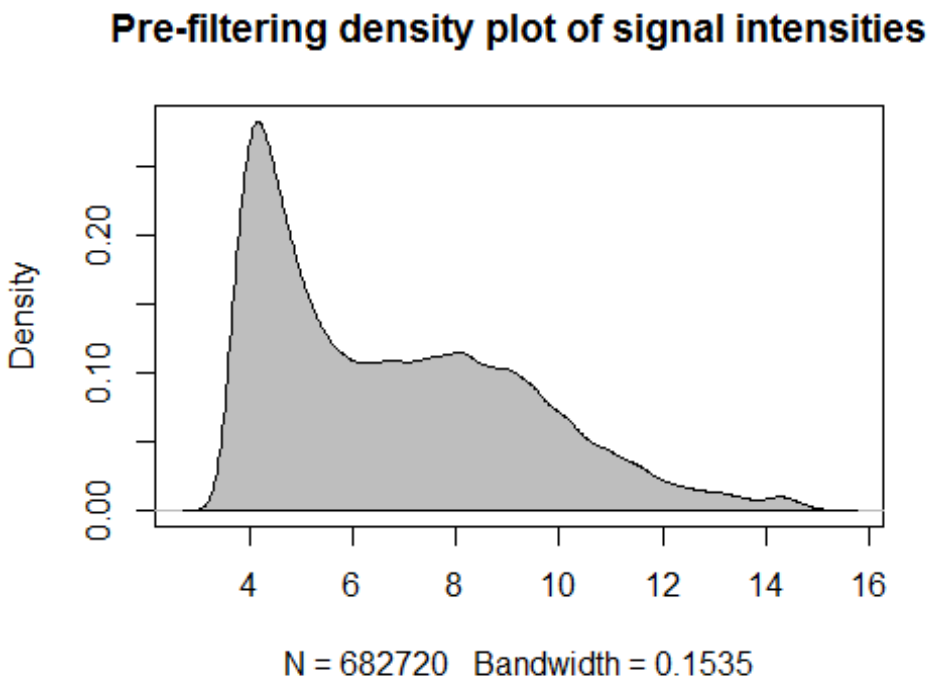

```
nrow(wnt_filtered)

## [1] 15137

ncol(wnt_filtered)

## [1] 34
```

However, if you would rather use a tested package such as geneFilter, you may do so as well. The documentation for those packages can be found: geneFilter - ftp://ftp2.uib.no/pub/bioconductor/2.7/bioc/html/genefilter.html

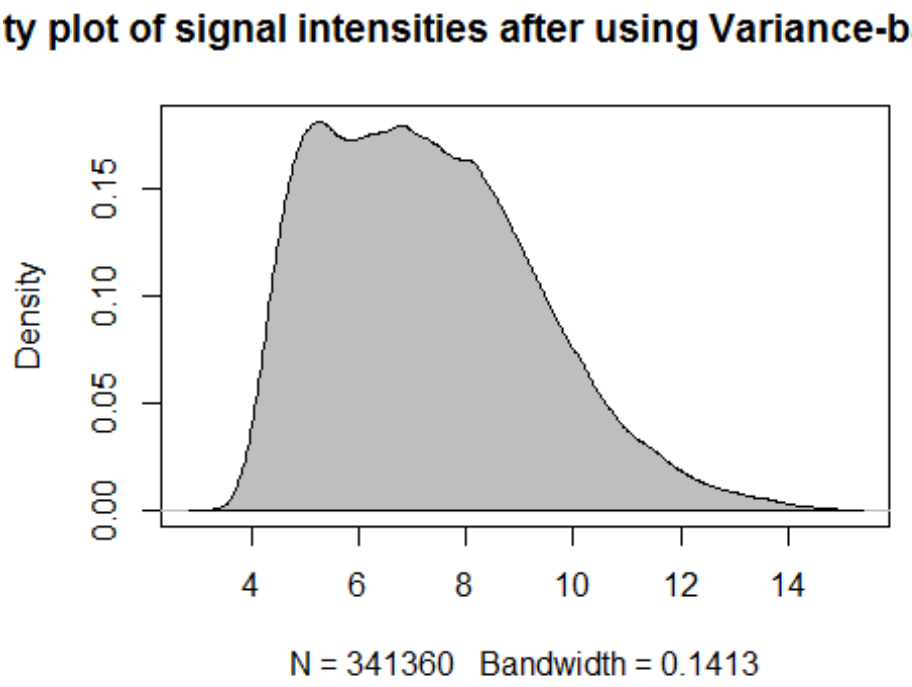
Filtering version 2

Here is geneFilter, utilizing the variance based filtering. Hackstadt and Hess, 2009 explain that variance based filtering along with RMA normalization (which is was used for pre-processing) return pretty good results. I just wanted to see if that is representative with our dataset as well...

```
#biocLite("genefilter","HTSFilter")
library(genefilter)
plot(pre.filter, "Pre-filtering density plot of signal intensities")
polygon(pre.filter,col="grey")
```




```
wnt_var_filter <- varFilter(as.matrix(wnt_control), var.func = IQR,
var.cutoff = 0.5, filterByQuantile = TRUE)
plot(density(wnt_var_filter), main="Density plot of signal intensities after
using Variance-based filtering")
polygon(density(wnt_var_filter),col="grey")
```




```
nrow(wnt_var_filter)

## [1] 10040

ncol(wnt_var_filter)

## [1] 34
```

Now compare the two different post-filtering density plots. Even though the genefilter method had less probes filtered out, the distribution looks relatively "normal". This is also because the threshold was more restrictive than the method I wanted to use. For the rest of the analysis, I will use the dataset used by the first quartile filtering because I want to input as many genes as possible; this method might skew the adjusted p-value calculation but I will take my chances. I would highly recommend running the analysis each dataset that was filtering by both methods and do a comparison on how well they work.

Starting Differential Expression analysis

Setting up the design matrix

Creating a design matrix will effectively tells limma which comparisons to do.

NOTE: The first two lines are required to download and install limma, make sure you do this before you run limma the first time. After that, it's not required, go ahead and comment it out.

```r
#source("https://www.bioconductor.org/biocLite.R")
#biocLite("limma")
library(limma)

## Warning: package 'limma' was built under R version 3.3.1

lev <-
as.factor(c(rep("HPA0",5),rep("HPA12",4),rep("HPA24",5),rep("HPA48",5),rep("H
PA72",5),rep("HPA120",5),rep("HPA168",5)))
design<-model.matrix(~0+lev)
colnames(design)<-c("HPA0","HPA12","HPA24","HPA48","HPA72","HPA120","HPA168")
rownames(design)<-colnames(wnt_control)
design

##        HPA0 HPA12 HPA24 HPA48 HPA72 HPA120 HPA168
## C0_1      1     0     0     0     0      0      0
## C0_2      1     0     0     0     0      0      0
## C0_3      1     0     0     0     0      0      0
## C0_4      1     0     0     0     0      0      0
## C0_5      1     0     0     0     0      0      0
## C12_1     0     1     0     0     0      0      0
## C12_2     0     1     0     0     0      0      0
## C12_3     0     1     0     0     0      0      0
## C12_4     0     1     0     0     0      0      0
## C24_1     0     0     0     0     1      0      0
## C24_2     0     0     0     0     1      0      0
## C24_3     0     0     0     0     1      0      0
## C24_4     0     0     0     0     1      0      0
## C24_5     0     0     0     0     1      0      0
## C48_1     0     0     0     0     0      1      0
## C48_2     0     0     0     0     0      1      0
## C48_3     0     0     0     0     0      1      0
## C48_4     0     0     0     0     0      1      0
## C48_5     0     0     0     0     0      1      0
## C72_1     0     0     0     0     0      0      1
## C72_2     0     0     0     0     0      0      1
## C72_3     0     0     0     0     0      0      1
## C72_4     0     0     0     0     0      0      1
## C72_5     0     0     0     0     0      0      1
## C120_1    0     0     1     0     0      0      0
## C120_2    0     0     1     0     0      0      0
## C120_3    0     0     1     0     0      0      0
## C120_4    0     0     1     0     0      0      0
## C120_5    0     0     1     0     0      0      0
## C168_1    0     0     0     1     0      0      0
## C168_2    0     0     0     1     0      0      0
## C168_3    0     0     0     1     0      0      0
## C168_4    0     0     0     1     0      0      0
## C168_5    0     0     0     1     0      0      0
## attr(,"assign")
```

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

This design matrix will also reflect the comparisons that will do in the later chunks; specifically this is the matrix that you will be reference when you run the makeContrasts() function. That will be two code chunks below.

Fit to linear model and run empirical bayes method

At this point you will fit each gene to a linear model and identify which genes are significantly differentially expressed based on an ANOVA method (done using the topTableF() function). Prior to running anova, the data is sent through the emperical bayes method to smooth out family-wise error to

```
fit <- lmFit(wnt_filtered,design)
topTableF(eBayes(fit))
```

```
##                      HPA0    HPA12    HPA24    HPA48    HPA72   HPA120
## axo07215-3_at   14.74261 14.74292 14.77562 14.81698 14.73431 14.76133
## axo24277-r_at   15.27410 15.26513 15.25779 15.24306 15.23902 15.26988
## axo14608-f_at   14.69949 14.71970 14.77412 14.76007 14.71483 14.75333
## axo12246-f_at   14.56815 14.57590 14.68195 14.70570 14.56620 14.64421
## axo18011-f_at   14.77418 14.74974 14.88232 14.90407 14.75082 14.82009
## axo22545-f_s_at 14.92084 14.94332 15.02036 15.06546 14.95116 14.96877
## axo08711-r_at   14.72132 14.73284 14.78536 14.78993 14.72552 14.77089
## axo04085-f_s_at 14.53720 14.56833 14.61512 14.59739 14.51111 14.58437
## axo18210-f_at   14.68645 14.70637 14.75541 14.73996 14.67896 14.73311
## axo10488-f_at   14.28866 14.31471 14.42631 14.39085 14.25533 14.36222
##                   HPA168  AveExpr        F      P.Value    adj.P.Val
## axo07215-3_at   14.79780 14.76809 441407.9 5.305132e-74 3.909434e-70
## axo24277-r_at   15.25581 15.25761 440887.9 5.400643e-74 3.909434e-70
## axo14608-f_at   14.75948 14.74075 420811.3 1.093547e-73 3.909434e-70
## axo12246-f_at   14.68503 14.63411 420040.4 1.124324e-73 3.909434e-70
## axo18011-f_at   14.84537 14.82010 411861.1 1.514154e-73 3.909434e-70
## axo22545-f_s_at 14.98703 14.98063 411231.6 1.549620e-73 3.909434e-70
## axo08711-r_at   14.79004 14.76020 401061.5 2.263871e-73 4.895460e-70
## axo04085-f_s_at 14.58513 14.57132 384549.3 4.278027e-73 8.094561e-70
## axo18210-f_at   14.75362 14.72244 372769.0 6.851468e-73 1.094682e-69
## axo10488-f_at   14.38747 14.34744 371440.8 7.231830e-73 1.094682e-69
```

Set up the contrasts

Setting up the contrasts means that you are telling limma which comparisons you want to do. This is referring back to the design matrix that you created previously (in my case called 'design')

Contrast 1: Each Time post amputation vs control (HPA 0)

NOTE: I will be running two separate comparisons: The following chunk is for all comparisons between each timepoint against the initial Day 0 timepoint (which I am considering the Control)

```
# contrast 1: between each HPA vs HPA0
contrasts <- makeContrasts("HPA12-HPA0","HPA24-HPA0","HPA48-HPA0","HPA72-
HPA0","HPA120-HPA0","HPA168-HPA0",levels=design)
fit2 <- contrasts.fit(fit,contrasts)
fit2 <- eBayes(fit2,trend=TRUE)
x<-topTableF(fit2,number=nrow(wnt_filtered))
control_limma_sig <- x[which(x$adj.P.Val<0.001),]
head(control_limma_sig)
```

```
##               HPA12.HPA0 HPA24.HPA0 HPA48.HPA0 HPA72.HPA0 HPA120.HPA0
## axo24112-r_at 1.63902840   3.586278   3.188112  3.0661104   3.4040004
## axo23311-r_at 1.02100145   3.026198   3.374082  2.1972400   2.9505428
## axo07467-f_at 2.41307765   4.991300   4.511834  4.2018618   4.9852404
## axo31448-f_at 0.46158425   4.471089   4.835175  0.2531646   3.0343306
## axo25197-f_at 0.00545105   2.892218   3.278520  0.0345946   1.6400540
## axo24605-f_at 0.11128905   1.571235   1.724349 -0.4287870   0.5199438
##               HPA168.HPA0   AveExpr        F      P.Value    adj.P.Val
## axo24112-r_at    3.745710 12.570649 342.8125 3.506177e-27 5.307300e-23
## axo23311-r_at    2.919506  8.218021 313.5476 1.387947e-26 1.050468e-22
## axo07467-f_at    5.190172  9.794088 289.0157 4.864942e-26 2.454688e-22
## axo31448-f_at    3.564081  8.237106 281.6139 7.250281e-26 2.743688e-22
## axo25197-f_at    2.402343  6.369445 263.2932 2.037994e-25 6.169823e-22
## axo24605-f_at    1.254382  9.405362 221.4250 2.894843e-24 7.303207e-21
```

```
nrow(control_limma_sig)
```

```
## [1] 9306
```

Based on this comparison, there are approximately 9306 probes that pass the threshold; these are considered to be differentially expressed genes (DEG).

Contrast 2: between adjacent HPA

This comparison is between all adjacent timepoints. My reason for running this contrast is so that I can identify genes that are changing dynamically between each timepoint.

```
contrasts2 <-makeContrasts("HPA12-HPA0","HPA24-HPA12","HPA48-HPA24","HPA72-
HPA48","HPA120-HPA72","HPA168-HPA120",levels=design)
fit2.2 <- contrasts.fit(fit,contrasts2)
fit2.2 <- eBayes(fit2.2,trend=TRUE)
x.2<-topTableF(fit2.2,number=nrow(wnt_filtered))
control_limma_sig.2 <- x.2[which(x.2$adj.P.Val<0.001),]
head(control_limma_sig.2)
```

```
##               HPA12.HPA0 HPA24.HPA12 HPA48.HPA24 HPA72.HPA48 HPA120.HPA72
## axo24112-r_at 1.63902840    1.947250  -0.3981660  -0.1220020    0.3378900
## axo23311-r_at 1.02100145    2.005197   0.3478834  -1.1768418    0.7533028
## axo07467-f_at 2.41307765    2.578223  -0.4794668  -0.3099718    0.7833786
## axo31448-f_at 0.46158425    4.009505   0.3640860  -4.5820104    2.7811660
## axo25197-f_at 0.00545105    2.886767   0.3863018  -3.2439252    1.6054594
## axo24605-f_at 0.11128905    1.459946   0.1531140  -2.1531358    0.9487308
##               HPA168.HPA120   AveExpr        F      P.Value    adj.P.Val
## axo24112-r_at     0.3417100 12.570649 342.8125 3.506177e-27 5.307300e-23
## axo23311-r_at    -0.0310370  8.218021 313.5476 1.387947e-26 1.050468e-22
## axo07467-f_at     0.2049320  9.794088 289.0157 4.864942e-26 2.454688e-22
## axo31448-f_at     0.5297500  8.237106 281.6139 7.250281e-26 2.743688e-22
## axo25197-f_at     0.7622888  6.369445 263.2932 2.037994e-25 6.169823e-22
## axo24605-f_at     0.7344386  9.405362 221.4250 2.894843e-24 7.303207e-21
```

```
nrow(control_limma_sig.2)
```

```
## [1] 9306
```

There are 9306 probes that pass the 0.001 threshold; these can be considered to be DEGs.

At this point you have two data.frames that represent all significant changed genes (depending on comparison). Notice that in the data.frame there are the first 6 columns which represent log2FC values for each comparison, Average expression, F value, unadjust p-value, and adjusted p-value (FDR). At this point you can also separate out which genes are up-regulated vs down-regulated by adding another chunk of code to separate out the genes based on the log2FC where anything that is > 0.58 is upregulated (equivalent to 1.5 log-fold change) and < -0.58 is downregulated:

```
upregulated = control_limma_sig.2[which(control_limma_sig.2$HPA12.HPA0 >
0.58),]
downregulated = control_limma_sig.2[which(control_limma_sig.2$HPA12.HPA0 < -
0.58),]

nrow(upregulated)
```

```
## [1] 334
```

```
nrow(downregulated)
```

```
## [1] 137
```

The above chunk of code finds that 334 DEGs are upregulated while 137 DEGs are downregulated.

I just focused on one of my significant lists from my second comparison. The timepoint focus I cared about 12 hpa vs 0 hpa; the comparison was structured where it was all the 12 hpa signal intensities were compared to against 0 hpa. This allows anything positive to be considered as upregulated at 12 hpa while the opposite is downregulated at 12 hpa (and therefore upregulated at 12 hpa).

Exporting the data.frames to your computer

I will export the three data.frames that document: 1) All the probes that are significant (p.adjust < 0.001); 2) DEGs that are significant and upregulated (p.adjust < 0.001 & HPA12.HPA0 > 0.58); and 3) DEGs that are significant and downregulated (p.adjust < 0.001 & HPA12.HPA0 < -0.58) s

```
write.table(control_limma_sig.2, "Tail-regeneration_adjacent-
significant.txt",row.names=TRUE, col.names=TRUE,sep='\t')
write.table(upregulated, "Tail-regeneration_12hpa-
upregulated.txt",row.names=TRUE, col.names=TRUE,sep='\t')
write.table(downregulated, "Tail-regeneration_12hpa-
downregulated.txt",row.names=TRUE, col.names=TRUE,sep='\t')
```

After exporting the lists, you can match probes to annotations related to your microarray to identify what gene each microarray probe matches to. Or you can run each probe sequence through BLAST to identify what gene it has the closest identity to. This is necessary as these identifiers will help with downstream analysis of each gene, especially for Gene Ontology analysis (which you can use DAVID for).

You were given a taste of using limma. Please refer to the R vignette (http://bioconductor.org/packages/release/bioc/html/limma.html) and the user's manual (http://www.bioconductor.org/packages/devel/bioc/vignettes/limma/inst/doc/usersguide.pdf) if you want to learn more about the utilities available in limma. Also refer to the Ritchie et al. 2015 paper to understand the underlying theories implemented in limma.