Computer lab, Bioinformatics: Omics data and pathway analysis

Perry Moerland

Thursday, February 25, 2021

We will continue with the differential expression analysis of the dataset analyzed last time. **If you didn't do so yet**, let us first install the different packages that we'll need. Remember that in order to execute R code from within RStudio, just click the green arrow head in the chunk of code below or put the cursor somewhere in the chunk and select Run - Run Current Chunk from the menu. You can also execute the code line by line using Ctrl-Enter. If you get a message whether packages should be compiled from source or whether old packages should be updated, just type "n" in the console.

```
# Installing packages should now be a piece of cake ...
# The commented line is needed for the LO desktops, but can be skipped on other systems
#.libPaths("C:/Scratch")
install.packages("BiocManager")
BiocManager::install(c("Biobase","annotate","arrayQualityMetrics","caret","e1071","genefilter",
           "hgu133a.db","lattice","limma","NMF","pamr","RColorBrewer"))
```

Now load the libraries so that you can use the functions defined in them:

```
library(annotate)
library(arrayQualityMetrics)
library(Biobase)
library(caret)
library(genefilter)
library(hgu133a.db)
library(lattice)
library(limma)
library(NMF)
```

Then retrieve the RMA-normalized data for the experiment of (Winn et al., 2009) that we generated last time, also including the information about the scan dates:

```
download.file("http://bioinformatics.amc.nl/wp-content/uploads/gs-bioinformatics/PathwaysNetworks/eset.
unzip("eset.zip")
load("eset.RData")
# Sample annotation:
pData(eset)
```

```
#                     condition sample scandate
# GSM367781.CEL.gz      preterm      1 02/11/05
# GSM367782.CEL.gz      preterm      2 02/26/04
# GSM367783.CEL.gz      preterm      3 11/10/04
# GSM367784.CEL.gz      preterm      4 02/11/05
# GSM367785.CEL.gz      preterm      5 08/25/04
# GSM367786.CEL.gz      preterm      6 02/26/04
# GSM367787.CEL.gz      preterm      7 03/10/05
# GSM367788.CEL.gz      preterm      8 03/10/05
# GSM367789.CEL.gz      preterm      9 02/11/05
```

```
# GSM367790.CEL.gz      preterm     10 11/10/04
# GSM367791.CEL.gz      preterm     11 11/10/04
# GSM367792.CEL.gz preeclampsia     12 11/10/04
# GSM367793.CEL.gz preeclampsia     13 02/26/04
# GSM367794.CEL.gz preeclampsia     14 02/11/05
# GSM367795.CEL.gz preeclampsia     15 11/10/04
# GSM367796.CEL.gz preeclampsia     16 11/10/04
# GSM367797.CEL.gz preeclampsia     17 11/10/04
# GSM367798.CEL.gz preeclampsia     18 08/25/04
# GSM367799.CEL.gz preeclampsia     19 11/10/04
# GSM367800.CEL.gz preeclampsia     20 02/26/04
# GSM367801.CEL.gz preeclampsia     21 03/10/05
# GSM367802.CEL.gz preeclampsia     22 02/11/05
# GSM367803.CEL.gz preeclampsia     23 02/11/05
```

1 Differential expression

A flexible way of analyzing microarray experiments uses *linear models* (Smyth, 2005). The idea is that an experiment can be concisely described using matrices and simple linear algebra. A key ingredient of a linear model is the *design matrix*. Each row of the design matrix corresponds to an array in the experiment and each column corresponds to a coefficient which is used to describe the RNA sources in the experiment. In this case the design matrix can be specified as follows:

```
design <- model.matrix(~0+pData(eset)$condition)
colnames(design) <- c("preeclampsia","preterm")
design
```

```
#    preeclampsia preterm
# 1             0       1
# 2             0       1
# 3             0       1
# 4             0       1
# 5             0       1
# 6             0       1
# 7             0       1
# 8             0       1
# 9             0       1
# 10            0       1
# 11            0       1
# 12            1       0
# 13            1       0
# 14            1       0
# 15            1       0
# 16            1       0
# 17            1       0
# 18            1       0
# 19            1       0
# 20            1       0
# 21            1       0
# 22            1       0
# 23            1       0
# attr(,"assign")
# [1] 1 1
# attr(,"contrasts")
```

```
# attr(,"contrasts")$`pData(eset)$condition`
# [1] "contr.treatment"
```

The first column of design matrix contains only 1's for the preeclampsia arrays and the second column contains only 1's for the preterm samples.

The following command fits the actual linear model:

```
fit <- lmFit(eset,design)
```

The `fit` object you just made contains coefficients. The first coefficient corresponds to the mean log-intensity for the preeclampsia samples and the second coeffcient to the mean log-intensity for the preterm samples:

```
# Only show this for the first probeset. First the coefficient in the linear
# model for the preeclampsia samples
fit$coef[1,1]
```

```
# [1] 10.3089
```

```
# Then calculate the mean 'by hand'
mean(exprs(eset)[1,as.character(pData(eset)$condition)=="preeclampsia"])
```

```
# [1] 10.3089
```

```
# Idem for the preterm samples
fit$coef[1,2]
```

```
# [1] 9.866928
```

```
mean(exprs(eset)[1,as.character(pData(eset)$condition)=="preterm"])
```

```
# [1] 9.866928
```

To find differentially expressed genes, we have to make a *contrast matrix*, which allows the coefficients defined by the design matrix to be combined into contrasts or comparisons of interest. The contrasts are arithmetic combinations of the parameters estimated in the model. The contrast matrix must have a number of rows equal to the number of coefficients in the linear model. Each column in the contrast matrix corresponds to a different contrast of interest where the rows correspond to the parameters estimated by the linear model fit. A contrast,in the contrast matrix, consists of a linear combination of the effects (coefficients) in the linear model. For our dataset we are interested in the difference between preeclampsia and preterm samples as a contrast.Note that since we always work on a log2 scale and because log2(preeclampsia) - log2(preterm) = log2(preeclampsia/preterm) this represents the log-ratio:

```
# Specify the comparison of interest
cont.matrix <- makeContrasts(PreeclampsiaVsPreterm = preeclampsia - preterm, levels = design)
cont.matrix
```

```
#              Contrasts
# Levels         PreeclampsiaVsPreterm
#   preeclampsia                     1
#   preterm                         -1
```

```
fit2 <- contrasts.fit(fit, cont.matrix)
# Indeed this gives the difference of the mean log-intensities (only show this for
# the first probeset)
fit2$coef[1,1]
```

```
# [1] 0.4419701
```

```
fit$coef[1,1] - fit$coef[1,2]
```

```
# [1] 0.4419701
```

Above we used the classical t-test for assessing statistical significance. However, since a typical microarray experiment is noisy and the number of arrays per group tends to be small, the standard error (=denominator of t-statistic) is hard to estimate reliably for each gene individually. The consequence is that some genes have small p-values only because, by chance, the denominator of the t-statistic was very small. Several researchers have proposed alternative statistics to obtain a more stable estimate of the gene-specific variance. In the **limma** package this has been implemented via a moderated t-statistic, details are described in (Smyth, 2005). The moderated t-statistic is simply calculated from `fit2` using the function `eBayes`:

```
fit2.eb <- eBayes(fit2)
# Extract a table from a linear model fit ordered by nominal p-value
tt.limma <- topTable(fit2.eb,n=Inf)
# Number of differentially expressed probesets with an FDR < 0.05
sum(tt.limma$adj<0.05)
```

```
# [1] 33
```

```
# Select the ten most differentially expressed genes
tt.limma[1:10,]
```

```
#                 logFC   AveExpr         t      P.Value   adj.P.Val
# 203140_at    1.3786542  8.739861  7.946999 2.960059e-08 0.000659590
# 205629_s_at  2.6749554 10.238140  6.635252 6.423190e-07 0.007156397
# 210511_s_at  1.9680280 11.308989  6.315939 1.402754e-06 0.010419187
# 207092_at    3.5642576  8.649346  6.197742 1.878425e-06 0.010463605
# 205829_at   -0.8819307 11.088605 -6.063261 2.623307e-06 0.010463605
# 215812_s_at  0.7366654  8.341534  5.941725 3.553181e-06 0.010463605
# 222033_s_at  1.5352577  8.720886  5.920269 3.749248e-06 0.010463605
# 219888_at    0.9509402  6.675417  5.896714 3.977170e-06 0.010463605
# 210141_s_at  1.3810884  8.680865  5.840764 4.576597e-06 0.010463605
# 219542_at    0.5536100  7.755475  5.830530 4.695779e-06 0.010463605
#                   B
# 203140_at   7.859050
# 205629_s_at 5.448030
# 210511_s_at 4.817328
# 207092_at   4.579826
# 205829_at   4.307058
# 215812_s_at 4.058289
# 222033_s_at 4.014155
# 219888_at   3.965630
# 210141_s_at 3.850065
# 219542_at   3.828882
```

Question 19 Compare results with the results obtained using a classical t-test and with GEO2R results. Can you explain the differences?

Answer

Just looking at the top-10, we observe only very minor differences between the limma results (*tt.limma*) and the results from GEO2R. It is actually strange that there are differences at all, since both differential expression analyses are identical. You can inspect the R code used by GEO2R by clicking on the *R script* tab.You'll notice that it is indeed similar to the code you used above (but more complicated since more generic). However, it turns out that the RMA-normalized data avaiable on GEO as submitted by the authors is slightly different from our RMA-normalized data. See the code below for a comparison of the expression values for probeset 31637_s_at in our data with those listed for the different GSMs for GSE14722. This is probably due to small changes in the RMA implementation over the past few years. Comparing the limma results (*tt.limma*) and the results

with a classical t-test shows that the log fold-changes are (of course) identical (columns *logFC* and *dm*) respectively (see code below). However, p-values (and therefore also the corresponding FDRs or adjusted p-values) are generally smaller in the limma results. In general the moderated t-statistics from limma will be more reliable than the classical ones, especially for a relatively small number of samples per group as is the case here.

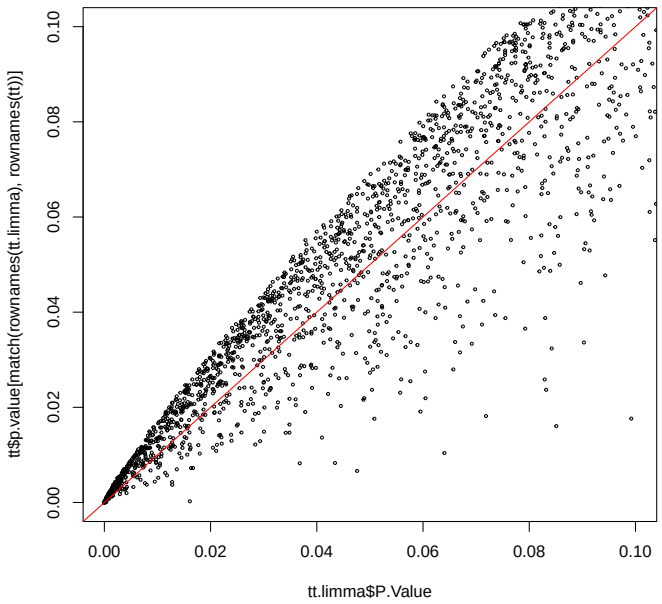
R code

```
# Our expression values for 31637_s_at
exprs(eset)["31637_s_at",]
```

```
# GSM367781.CEL.gz GSM367782.CEL.gz GSM367783.CEL.gz GSM367784.CEL.gz
#        10.350205         9.735635         9.573338        10.414047
# GSM367785.CEL.gz GSM367786.CEL.gz GSM367787.CEL.gz GSM367788.CEL.gz
#         9.478606         9.526482        10.062220        10.218725
# GSM367789.CEL.gz GSM367790.CEL.gz GSM367791.CEL.gz GSM367792.CEL.gz
#        10.522329        10.265910         9.759412        10.141612
# GSM367793.CEL.gz GSM367794.CEL.gz GSM367795.CEL.gz GSM367796.CEL.gz
#         9.520392        10.289127         9.691382        10.222393
# GSM367797.CEL.gz GSM367798.CEL.gz GSM367799.CEL.gz GSM367800.CEL.gz
#         9.785971         9.396659         9.975078         9.724093
# GSM367801.CEL.gz GSM367802.CEL.gz GSM367803.CEL.gz
#        10.382650        10.200475        10.430617
```

R code

```
# Results for a classical t-test
tt <- rowttests(exprs(eset),as.factor(pData(eset)$condition))
# Use 'match' to have the probesets in the same order
plot(tt.limma$P.Value,tt$p.value[match(rownames(tt.limma),rownames(tt))],xlim=c(0,0.1),
     ylim=c(0,0.1),cex=0.4)
abline(0,1,col="red")
```



Let us now try to take the batch effects observed between the different scan dates into account and see what we gain by doing so. In a linear modeling framework this is easily done by extending our previous linear model with the scan dates as a *blocking* variable:

```
# Switch order from (preeclampsia,preterm) to (preterm,preeclampsia)
pData(eset)$condition <- relevel(pData(eset)$condition, "preterm")
# Include scan date as a blocking variable
design <- model.matrix(~pData(eset)$condition + pData(eset)$scandate)
# This is a more complicated design than the previous one. I will explain this some time
# this afternoon
head(design,n=4)
```

```
#   (Intercept) pData(eset)$conditionpreeclampsia
# 1           1                                 0
# 2           1                                 0
# 3           1                                 0
# 4           1                                 0
#   pData(eset)$scandate02/26/04 pData(eset)$scandate03/10/05
# 1                            0                            0
# 2                            1                            0
# 3                            0                            0
# 4                            0                            0
#   pData(eset)$scandate08/25/04 pData(eset)$scandate11/10/04
# 1                            0                            0
# 2                            0                            0
# 3                            0                            1
# 4                            0                            0
```

```
tail(design,n=4)
```

```
#    (Intercept) pData(eset)$conditionpreeclampsia
# 20           1                                 1
```

```
# 21           1                                 1
# 22           1                                 1
# 23           1                                 1
#    pData(eset)$scandate02/26/04 pData(eset)$scandate03/10/05
# 20                            1                            0
# 21                            0                            1
# 22                            0                            0
# 23                            0                            0
#    pData(eset)$scandate08/25/04 pData(eset)$scandate11/10/04
# 20                            0                            0
# 21                            0                            0
# 22                            0                            0
# 23                            0                            0
```

```
fit <- lmFit(eset,design)
```

We can now estimate differential expression between preeclamptic and preterm samples corrected for batch effects:

```
fit.eb <- eBayes(fit)
# The second coefficient now corresponds to the comparison of interest
tt.pvsp <- topTable(fit.eb,coef=2,n=Inf)
# Number of differentially expressed probesets with an FDR < 0.05
sum(tt.pvsp$adj<0.05)
```

```
# [1] 63
```

```
# Select the ten most differentially expressed genes
tt.pvsp[1:10,]
```

```
#                 logFC   AveExpr         t      P.Value   adj.P.Val
# 203140_at    1.3528666  8.739861  7.388504 2.972018e-07 0.006622548
# 205629_s_at  2.6949601 10.238140  6.475991 2.092320e-06 0.018136039
# 219542_at    0.5509345  7.755475  6.105393 4.766289e-06 0.018136039
# 1007_s_at    0.4078821 10.097521  6.041027 5.508363e-06 0.018136039
# 203184_at   -0.6234173 10.619103 -5.983601 6.270015e-06 0.018136039
# 219137_s_at -0.3431705  6.126684 -5.960324 6.608684e-06 0.018136039
# 91826_at     0.6934606 10.292919  5.901991 7.542042e-06 0.018136039
# 207092_at    3.3590175  8.649346  5.900881 7.561052e-06 0.018136039
# 210511_s_at  1.8873767 11.308989  5.868269 8.142048e-06 0.018136039
# 210141_s_at  1.3676989  8.680865  5.824544 8.993379e-06 0.018136039
#                   B
# 203140_at   6.445439
# 205629_s_at 4.792524
# 219542_at   4.082364
# 1007_s_at   3.956851
# 203184_at   3.844345
# 219137_s_at 3.798602
# 91826_at    3.683617
# 207092_at   3.681424
# 210511_s_at 3.616918
# 210141_s_at 3.530190
```

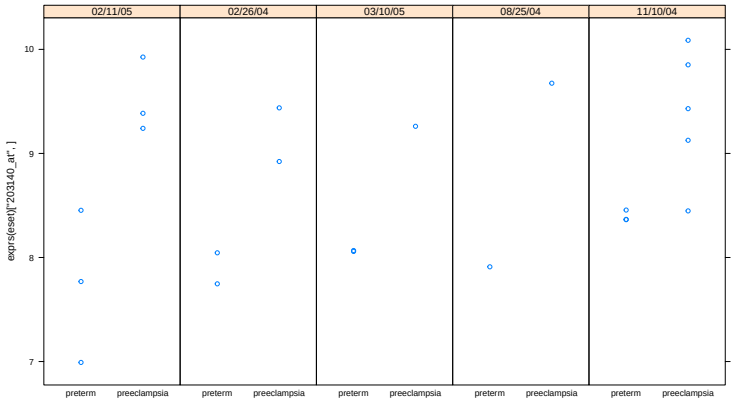
Question 20 Did we gain anything by correcting for the batch effect?

Answer

Yes, the number of genes differentially expressed with an adjusted p-value < 0.05 almost doubled from 33 to 63.

The most differentially expressed gene is 203140_at (BCL6, we'll come back to annotating the probesets below). Let's visualize its expression values:

```
# Create a panel plot
stripplot(exprs(eset)["203140_at",] ~ pData(eset)$condition |
            factor(pData(eset)$scandate), layout = c(5, 1))
```



We can use the same model to estimate differential expression between the five different batches:

```
# Coefficients 3-6 correspond to the pairwise comparisons of batch '02/11/05' with the
# other batches
tt.batch <- topTable(fit.eb,coef=3:6,n=Inf)
sum(tt.batch$adj<0.05)
```

```
# [1] 13278
```

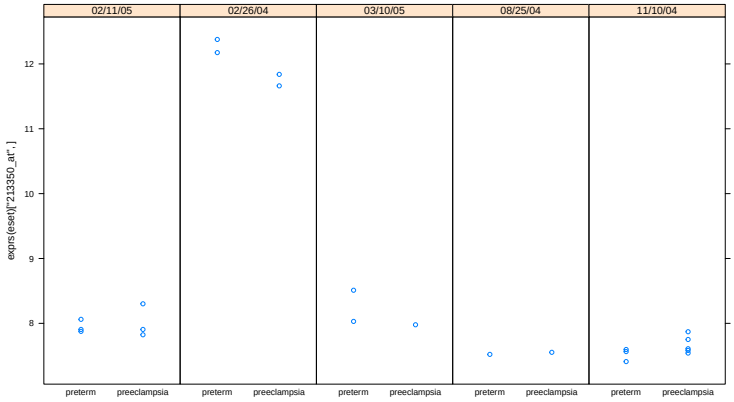
```
# Select the ten most differentially expressed genes
tt.batch[1:10,]
```

```
#                     pData.eset..scandate02.26.04
# AFFX-BioB-5_at                          3.250894
# AFFX-BioB-3_at                          3.455330
# AFFX-r2-Ec-bioC-3_at                    3.074013
# AFFX-r2-Ec-bioC-5_at                    3.532998
# AFFX-r2-Ec-bioB-5_at                    3.612904
# 213350_at                               4.033726
# AFFX-BioB-M_at                          3.633030
# AFFX-BioC-3_at                          2.935646
# AFFX-r2-Ec-bioB-M_at                    4.037140
# AFFX-BioC-5_at                          3.024856
#                     pData.eset..scandate03.10.05
# AFFX-BioB-5_at                         0.3488865
# AFFX-BioB-3_at                         0.4364530
# AFFX-r2-Ec-bioC-3_at                   0.8701823
# AFFX-r2-Ec-bioC-5_at                   0.7348786
# AFFX-r2-Ec-bioB-5_at                   0.2999999
# 213350_at                              0.1835262
# AFFX-BioB-M_at                         0.6201961
```

```
# AFFX-BioC-3_at                          0.6541128
# AFFX-r2-Ec-bioB-M_at                    0.6292786
# AFFX-BioC-5_at                          0.7909894
#                      pData.eset..scandate08.25.04
# AFFX-BioB-5_at                          0.7007151
# AFFX-BioB-3_at                          0.8819665
# AFFX-r2-Ec-bioC-3_at                    0.6223550
# AFFX-r2-Ec-bioC-5_at                    0.5293736
# AFFX-r2-Ec-bioB-5_at                    0.9585527
# 213350_at                              -0.4416965
# AFFX-BioB-M_at                          0.9856034
# AFFX-BioC-3_at                          0.5859292
# AFFX-r2-Ec-bioB-M_at                    1.1012173
# AFFX-BioC-5_at                          0.5645671
#                      pData.eset..scandate11.10.04  AveExpr        F
# AFFX-BioB-5_at                          0.4797429 7.669566 460.0685
# AFFX-BioB-3_at                          0.4733856 7.260402 418.3413
# AFFX-r2-Ec-bioC-3_at                    0.4244627 9.342182 396.6893
# AFFX-r2-Ec-bioC-5_at                    0.2966317 8.960418 383.9480
# AFFX-r2-Ec-bioB-5_at                    0.6730519 7.844852 366.0559
# 213350_at                              -0.3557772 8.541187 365.9181
# AFFX-BioB-M_at                          0.7275697 8.152884 356.2213
# AFFX-BioC-3_at                          0.3078810 8.446862 349.1982
# AFFX-r2-Ec-bioB-M_at                    0.8060223 8.219023 348.8221
# AFFX-BioC-5_at                          0.3691374 8.650347 306.7796
#                           P.Value    adj.P.Val
# AFFX-BioB-5_at       4.882800e-20 1.088034e-15
# AFFX-BioB-3_at       1.301441e-19 1.450001e-15
# AFFX-r2-Ec-bioC-3_at 2.249934e-19 1.671176e-15
# AFFX-r2-Ec-bioC-5_at 3.148666e-19 1.754043e-15
# AFFX-r2-Ec-bioB-5_at 5.144653e-19 1.918051e-15
# 213350_at            5.164614e-19 1.918051e-15
# AFFX-BioB-M_at       6.807292e-19 2.091253e-15
# AFFX-BioC-3_at       8.353443e-19 2.091253e-15
# AFFX-r2-Ec-bioB-M_at 8.446472e-19 2.091253e-15
# AFFX-BioC-5_at       3.157085e-18 7.034932e-15
```

Differences between batches are impressive indeed! Note that many of the top hits are hybridization controls. Let's visualize the expression values of one of the genes most affected by the batch effect (213350_at: RPS11):

```
# Create a panel plot
stripplot(exprs(eset)["213350_at",] ~ pData(eset)$condition |
            factor(pData(eset)$scandate), layout = c(5, 1))
```



Batch corrected expression values corresponding to the linear model-based batch correction above can be generated using the function `removeBatchEffect`:

```
eset.corrected <- eset
exprs(eset.corrected) <- removeBatchEffect(exprs(eset), batch=factor(pData(eset)$scandate),
                                           design=model.matrix(~pData(eset)$condition))
```

We regenerate an arrayQualityMetrics report but now on the batch corrected data:

```
if (!dir.exists("aQ_corrected")){
  arrayQualityMetrics(eset.corrected,outdir="aQ_corrected",
                      intgroup=c("condition","scandate"),force=TRUE)
}
```

Question 21 Do you still observe batch effects and outliers after the correction? What could be a reason of the large remaining within-group heterogeneity (think about the way the study was set up).

Answer

No outliers are detected and no clear batch effects can be seen. Part of the heterogeneity might be explained by the large variation in the number of weeks at which the placenta was obtained (24-36 wk in both groups). You might actually want to include this in your linear model as another covariate in a full-fledged analysis. The information is available at GSE14722.

Take home message: batch effects are pervasive and their effect is often blissfully ignored or underestimated. See Leek et al., 2010 for a good review, illustrated with more examples.See also the lively discussion on a recent comparison of the transcriptional landscapes between human and mouse tissues.

If you want to learn more about linear models, see the Biomedical Data Science course of Rafael Irizarry et al.

The tables that we generated above only contained probeset identifiers. In general you would like to add gene annotation to these tables. For Affymetrix arrays this is easily done as follows:

```
probesets <- as.character(rownames(tt.pvsp))
# Retrieve Entrez Gene IDs for the U133A chip
eg <- getEG(probesets, "hgu133a")
# Retrieve gene symbols for the U133A chip
sym <- getSYMBOL(probesets, "hgu133a")
tt.pvsp$ID <- as.character(rownames(tt.pvsp))
# Generate an HTML table with annotation
htmlpage(as.data.frame(eg), filename = "report.html", title = "HTML report",
```

```
        othernames = data.frame(sym, tt.pvsp),
        table.head = c("GeneID","Symbol", colnames(tt.pvsp)),
        table.center = TRUE)
```

Also save the results in a tab-delmited text file:

```
write.table(data.frame(GeneID=eg,Symbol=sym,tt.pvsp),file="tt_GSE14722.txt",sep="\t",
            row.names=FALSE)
```

2 Geneset enrichment analysis

Question 22 Use the DAVID Functional Annotation Tool to find genesets that are enriched for the genes probed in this experiment. In the lecture three types of gene set analyses approaches were described which of the approaches does DAVID use? Things to keep in mind when using DAVID:

- What selection of genes do you use?
- Think about using an appropriate background set.
- Which identifier to use?

Investigate the KEGG pathway results and the tissue expression results in particular.

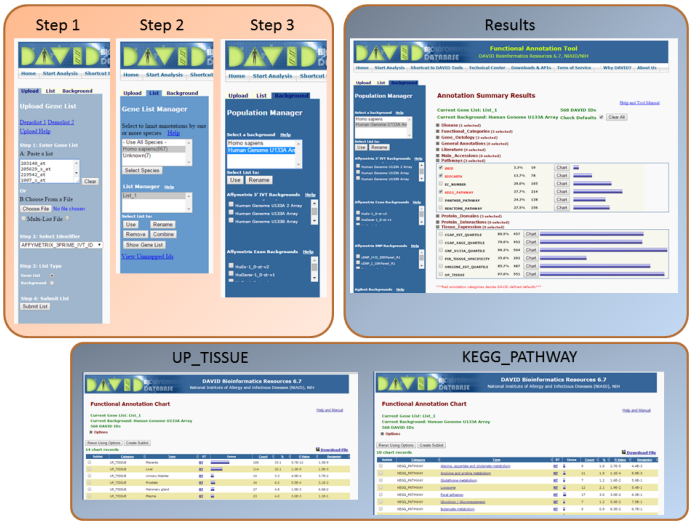
Answer

DAVID takes a cut-off based overrepresentation approach (ORA). I used a selection based on those probesets with a nominal P-value <0.01 in the limma analysis when taking the batch effect into account; see the file tt_GSE14722.txt that you generated above. In general, for an ORA-type analysis the list of selected genes should not be too short (<100 genes) nor too long (>1000 genes). Here, the selection criterion leads to a list of 674 probesets. An ORA-type analysis using DAVID then consists of three steps, illustrated in the workflow below:

- Paste the selected identfiers under 'Upload'. Here I decided to use Affymetrix probeset IDs and therefore selected 'AFFYMETRIX_3PRIME_IVT_ID' and indicated that the list type is 'Gene List';
- Then select 'Homo Sapiens' as species under 'List';
- Finally, under 'Background' select the platform used in the experiment, that is 'Human Genome U133A Array'.

Results for various geneset categories are indicated in the 'Annotation Summary Results'. Here results for the categories 'Tissue_Expression' and 'Pathways' are unfolded. For the 'Tissue_Expression' results it is reassuring to see that the signature is enriched for placenta-specific genes (UP_TISSUE). One of the most enriched KEGG pathways is *Alanine, aspartate and glutamate metabolism*, which has indeed been implied in preeclampsia before (see Pennington et al.).

Workflow

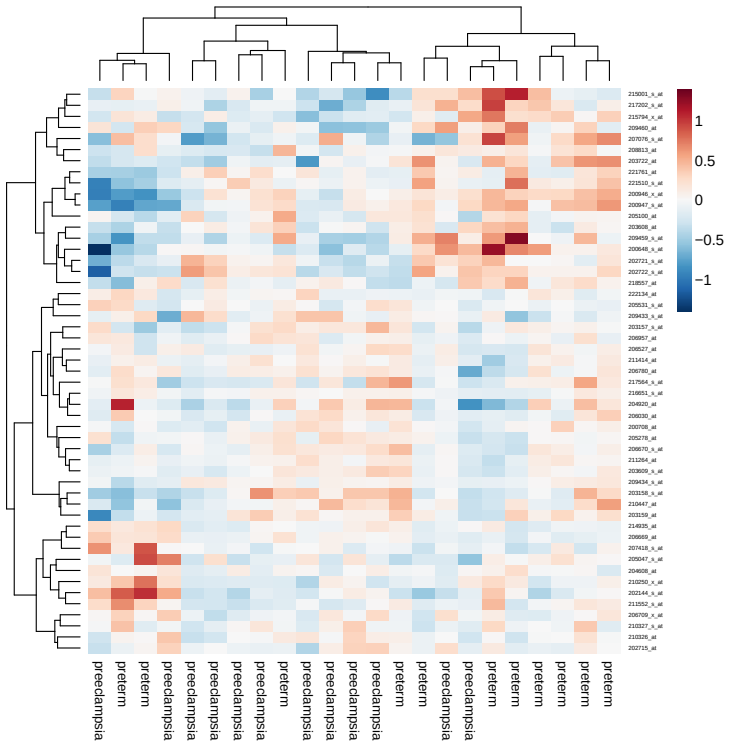


Let us now have a closer look at the alanine, aspartate and glutamate metabolism pathway:

```
# Select the related genes (KEGG ID: 00250)
hsa00250 <- AnnotationDbi::select(hgu133a.db, keys = "00250", keytype = "PATH",
                    columns = c("PROBEID", "ENTREZID", "SYMBOL", "GENENAME"))

# Extract and scale the expression data
E <- exprs(eset[hsa00250$PROBEID, ])
E <- t(scale(t(E), center = TRUE, scale = FALSE))

# Visualize the expression of genes related to alanine, aspartate and glutamate
# metabolism in a heatmap
aheatmap(E, dist = "pearson", col = "-RdBu:256", breaks = 0, labCol = pData(eset)$condition)
```



Restricting clustering to this pathway is still far from perfect, since many genes in the pathway are not differentially expressed.

We could also do a DAVID-type enrichment analysis in R by just constructing the 2x2 contingency table:

```
# Define DE genes based on thresholds
diffExpr <- decideTests(fit.eb, p.value = 0.01, adjust.method = "none")
# Number of DE genes per coefficient
summary(diffExpr)
```

```
#        (Intercept) pData(eset)$conditionpreeclampsia
# Down             0                               338
# NotSig           0                             21608
# Up           22283                               337
#        pData(eset)$scandate02/26/04 pData(eset)$scandate03/10/05
# Down                           3414                          278
# NotSig                        14639                        21579
# Up                             4230                          426
#        pData(eset)$scandate08/25/04 pData(eset)$scandate11/10/04
# Down                           3379                         2194
# NotSig                        15436                        17989
# Up                             3468                         2100
```

```
# Select the comparison of interest: preeclampsia vs.preterm
diffExpr <- diffExpr[,2]
diffExpr <- diffExpr != 0
# Indicate members of the alanine, aspartate and glutamate metabolism pathway
inPathway <- featureNames(eset) %in% hsa00250$PROBEID
# Construct contingency table
tab <- table(diffExpr, inPathway)
tab
```

```
#         inPathway
# diffExpr FALSE  TRUE
#    FALSE 21568    40
#    TRUE    664    11
```

```
# Perform an enrichment test using Fisher's exact test
fisher.test(tab, alternative = "greater")
```

```
#
#   Fisher's Exact Test for Count Data
#
# data:  tab
# p-value = 2.872e-07
# alternative hypothesis: true odds ratio is greater than 1
# 95 percent confidence interval:
#  4.658738      Inf
# sample estimates:
# odds ratio
#   8.930133
```

Indeed, the signature is clearly enriched for genes part of the alanine, aspartate and glutamate metabolism pathway.

Question 23 If there is still time left, you can work with two other enrichment tools:

- Graphite Web
- STRING

3 Classification

We will now have a brief look at how to construct a gene expression-based classifier to predict patient status (preeclampsia/preterm). The **caret** (short for Classification And REgression Training) package offers a versatile set of functions for this purpose. The typical steps in building a classifier are:

First, split the data into a training and test set:

```
# Fix the random seed to make the exercise reproducible
set.seed(13)
# training set: 75%, test set: 25%
inTrain <- createDataPartition(y = pData(eset)$condition,p = .75,list = FALSE)
training <- eset[ ,inTrain]
testing <- eset[,-inTrain]
ncol(training)
```

```
# Samples
#      18
```

```
ncol(testing)
```

```
# Samples
#       5
```

Second, fit a model using the training data (both the gene expression data and the class labels). Here, we use the *nearest shrunken centroids* model also refered to as PAM (Prediction Analysis for Microarrays). This is a simple linear model, proposed in 2002 by Rob Tibshirani (paper). While fitting the model, this approach also identifies subsets of genes that best characterize each class.

```
pamrFit <- train(x=t(exprs(training)), y=pData(training)$condition, method="pam",
                 tuneLength=10,trControl = trainControl(method = "cv"))
```

```
# 12345678910111213141516171819202122232425262728293011111111111
```

```
pamrFit
```

```
# Nearest Shrunken Centroids
#
#    18 samples
# 22283 predictors
#     2 classes: 'preterm', 'preeclampsia'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 17, 16, 17, 16, 16, 16, ...
# Resampling results across tuning parameters:
#
#   threshold  Accuracy  Kappa
#   0.2197355  0.75      0.375
#   0.8789418  0.85      0.625
#   1.5381482  0.90      0.750
#   2.1973546  1.00      1.000
#   2.8565610  0.95      0.875
#   3.5157673  0.95      0.875
#   4.1749737  0.95      0.875
#   4.8341801  0.95      0.875
#   5.4933865  0.95      0.875
#   6.1525928  0.45      0.100
#
# Accuracy was used to select the optimal model using the largest value.
# The final value used for the model was threshold = 2.197355.
```

The optimal number of features is determined by a threshold value and the optimal threshold value is selected using cross-validation.

Third the optimal model is used to predict the class labels for the test data:

```
pamrClasses <- predict(pamrFit, newdata = t(exprs(testing)))
confusionMatrix(data = pamrClasses, pData(testing)$condition)
```

```
# Confusion Matrix and Statistics
#
#               Reference
# Prediction     preterm preeclampsia
#   preterm            1            1
#   preeclampsia       1            2
#
#                Accuracy : 0.6
#                  95% CI : (0.1466, 0.9473)
#     No Information Rate : 0.6
#     P-Value [Acc > NIR] : 0.6826
#
#                   Kappa : 0.1667
#
#  Mcnemar's Test P-Value : 1.0000
#
#             Sensitivity : 0.5000
```

```
#             Specificity : 0.6667
#          Pos Pred Value : 0.5000
#          Neg Pred Value : 0.6667
#              Prevalence : 0.4000
#          Detection Rate : 0.2000
#    Detection Prevalence : 0.4000
#       Balanced Accuracy : 0.5833
#
#        'Positive' Class : preterm
#
```

In this case, this leads to a model that makes one error on the five test samples. Of course, one should split the data into a training and a test set multiple times to obtain more reliable estimates of the accuracy of this model.

Finally, let us have a look at the most important variables in the model:

```
varimps <- varImp(pamrFit)
getSYMBOL(head(featureNames(training)[order(varimps$importance[,1],decreasing=TRUE)]),"hgu133a")
```

```
#   207092_at 210511_s_at 222033_s_at 205629_s_at   203140_at 211918_x_at
#       "LEP"     "INHBA"      "FLT1"       "CRH"      "BCL6"    "PAPPA2"
```

Not surprisingly, the most important variables are those that are strongly differentially expressed.