EMA_Microarray_Analysis

Tzu L. Phang

April 9, 2015

Experimental Design

Read in descriptive data file and run summary statistics to examine the distribution of the data prior to importing CEL files

```
options(width=500)
## Load EMA and oligo libraries
library(EMA)
library(oligo)

## Read in descriptive data file
PBHI.file.info = read.table(file = "./data/PHBI cel annotation BIO6660 v2.txt", header = T, sep = "\t")
## print(PBHI.file.info)
knitr::kable(PBHI.file.info)
```

PHBI.cel.file	PHBI.number	Clinical.Category	Age	Race	Sex	Scan.Date	Batch
PHBI_001.cel	PHBI_001	IPAH	62	White	Female	10/1/09	1
PHBI_002.cel	PHBI_002	IPAH	49	White	Female	10/1/09	1
PHBI_003.cel	PHBI_003	IPAH	64	White	Male	10/1/09	1
PHBI_019.cel	PHBI_019	Failed Donor	60	White	Male	10/1/09	1
PHBI_027.cel	PHBI_027	Failed Donor	49	White	Male	10/1/09	1
PHBI_042.cel	PHBI_042	IPAH	44	White	Female	10/1/10	2
PHBI_044.cel	PHBI_044	IPAH	40	White	Female	10/1/10	2
PHBI_053.cel	PHBI_053	Failed Donor	25	White	Male	10/1/10	2
PHBI_059.cel	PHBI_059	Failed Donor	55	White	Male	10/1/10	2
PHBI_062.cel	PHBI_062	IPAH	45	White	Male	10/1/10	2
PHBI_066.cel	PHBI_066	Failed Donor	19	White	Male	10/1/10	2
PHBI_067.cel	PHBI_067	Failed Donor	62	White	Male	10/1/11	3
PHBI_068.cel	PHBI_068	Failed Donor	21	White	Female	10/1/11	3
PHBI_069.cel	PHBI_069	Failed Donor	28	White	Female	10/1/11	3
PHBI_073.cel	PHBI_073	IPAH	71	White	Female	10/1/11	3
PHBI_075.cel	PHBI_075	IPAH	56	White	Male	10/1/11	3
PHBI_078.cel	PHBI_078	IPAH	61	White	Male	10/1/11	3

We will use the "Clinical.Category" column to define the 2 comparing groups

```
## Extract groups: IPAH and Failed Donor
PBHI.type.cl = as.character(PBHI.file.info$Clinical.Category)
PBHI.type.cl
```

```
##  [1] "IPAH"         "IPAH"         "IPAH"         "Failed Donor"
##  [5] "Failed Donor" "IPAH"         "IPAH"         "Failed Donor"
##  [9] "Failed Donor" "IPAH"         "Failed Donor" "Failed Donor"
## [13] "Failed Donor" "Failed Donor" "IPAH"         "IPAH"
## [17] "IPAH"
```

We suspect there might be batch-effect. We will use the "Batch" column to extract potential batch-effect information

```
PBHI.batch.cl = PBHI.file.info$Batch
PBHI.batch.cl
```

```
##  [1] 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3
```

What is the average age for the 2 groups of patients

```
## Run basic summary statistics
tapply(PBHI.file.info$Age, PBHI.file.info$Clinical.Category, mean)
```

```
## Failed Donor         IPAH
##     39.87500     54.66667
```

What is the gender distribution

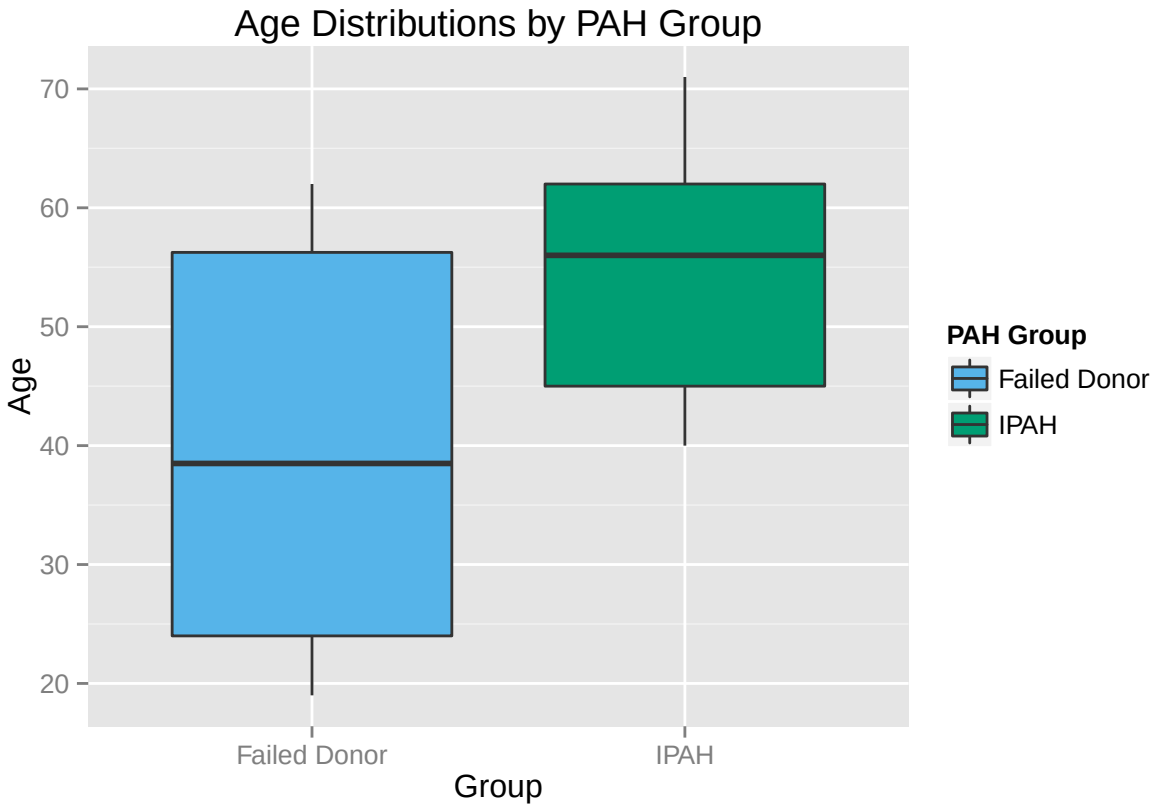
```
tapply(PBHI.file.info$Sex, PBHI.file.info$Clinical.Category, summary)
```

```
## $`Failed Donor`
## Female   Male
##      2      6
##
## $IPAH
## Female   Male
##      5      4
```

Visualize Age Distribution by Group

```
library(ggplot2)

plot = ggplot(PBHI.file.info, aes(factor(PBHI.file.info$Clinical.Category),
    PBHI.file.info$Age)) + geom_boxplot(aes(fill = factor(PBHI.file.info$Clinical.Category))) +
    labs(title = "Age Distributions by PAH Group") + xlab("Group") + ylab("Age")
plot + scale_fill_manual("PAH Group", values = c("#56B4E9", "#009E73"))
```



Preprocessing and Quality Assessemnt

Import CEL files

Import CEL files, read in data, and normalize data using the RMA function

```
## Import CEL files
PBHI.CEL = PBHI.file.info$PBHI.CEL
PBHI.CEL = list.celfiles('./data')
PBHI.CEL = paste('./data/', PBHI.CEL, sep = '')

## Read CEL files to directory
PHBI.data = read.celfiles(PBHI.CEL, verbose = F)
```

```
## Reading in : ./data/PHBI_001.CEL
## Reading in : ./data/PHBI_002.CEL
## Reading in : ./data/PHBI_003.CEL
## Reading in : ./data/PHBI_019.CEL
## Reading in : ./data/PHBI_027.CEL
## Reading in : ./data/PHBI_042.CEL
## Reading in : ./data/PHBI_044.CEL
## Reading in : ./data/PHBI_053.CEL
## Reading in : ./data/PHBI_059.CEL
## Reading in : ./data/PHBI_062.CEL
## Reading in : ./data/PHBI_066.CEL
## Reading in : ./data/PHBI_067.CEL
## Reading in : ./data/PHBI_068.CEL
```

```
## Reading in : ./data/PHBI_069.CEL
## Reading in : ./data/PHBI_073.CEL
## Reading in : ./data/PHBI_075.CEL
## Reading in : ./data/PHBI_078.CEL
```

```
## Normalize the data
PHBI.norm = rma(PHBI.data)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

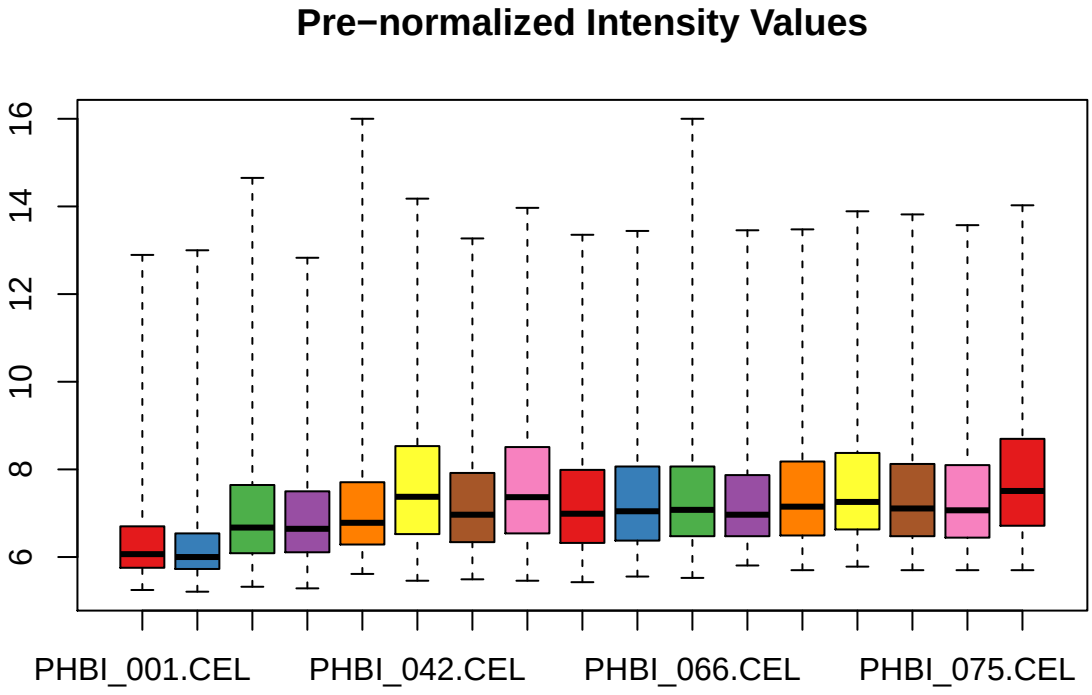
Visual Inspection: Before and After Data Normalization

Create boxplots of log-intensity distribution to visualize data pre and post normalization

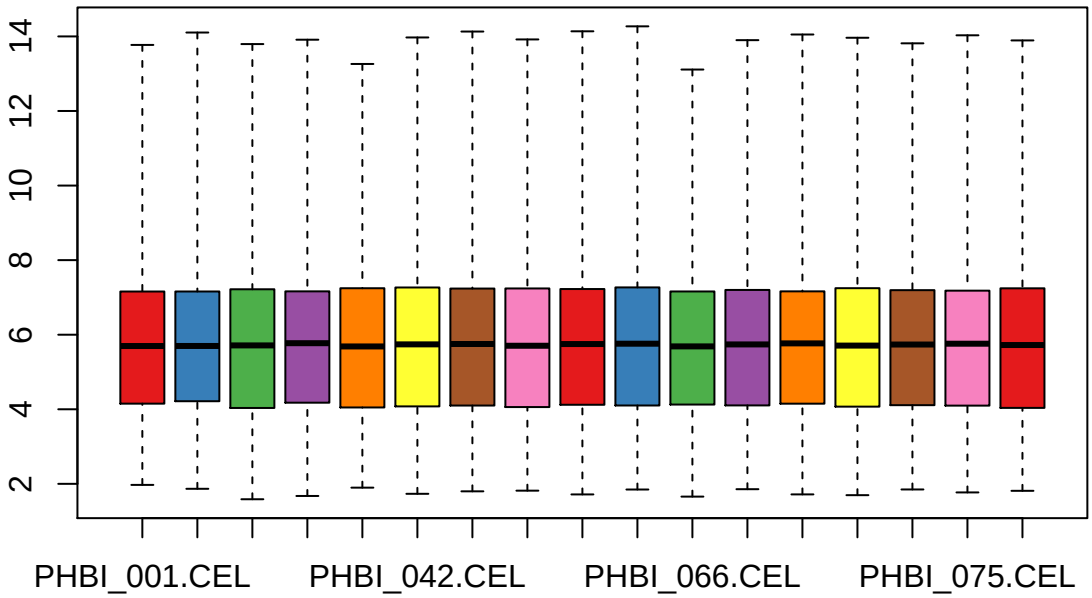
```
## Load color libraries
library(RColorBrewer)

## Set color palette
color.palette = brewer.pal(8, "Set1")


## Pre-normalized intensity values boxplot
boxplot(PHBI.data, col = color.palette, main = "Pre-normalized Intensity Values")
```

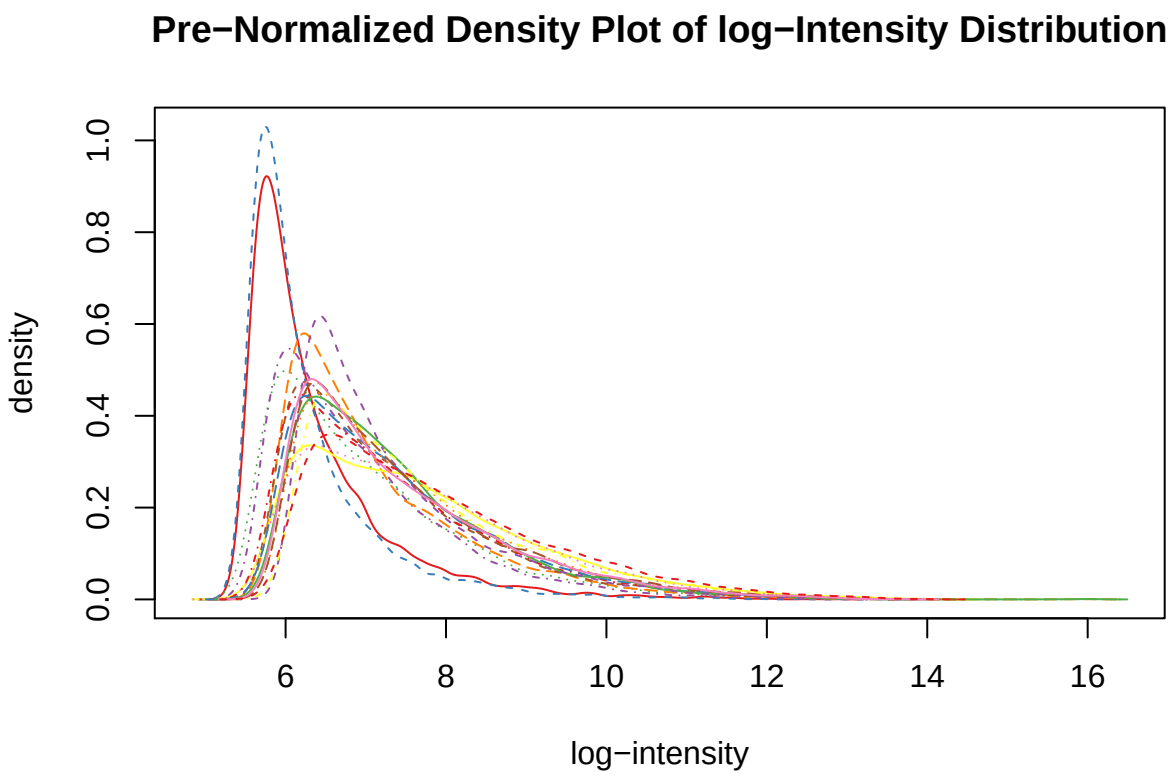



```
## Normalized intensity values boxplot
boxplot(PHBI.norm, col = color.palette, main = "Normalized Intensity Values")
```

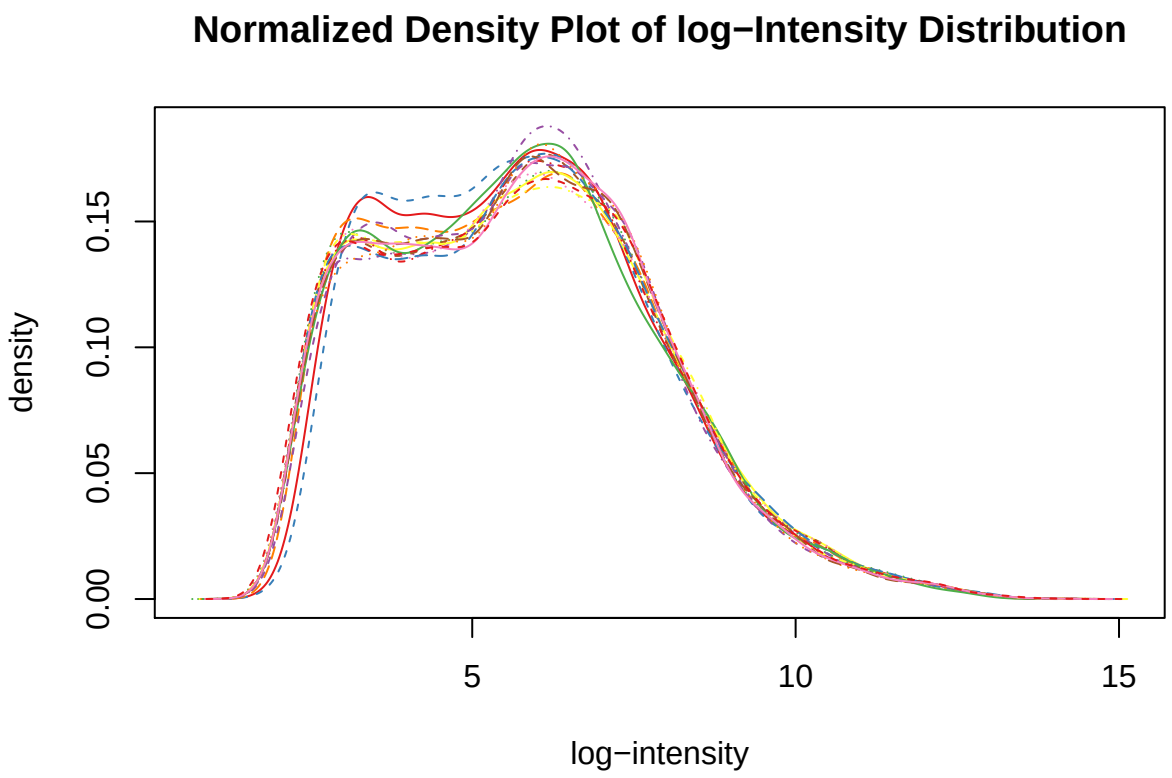


Create density plots of log-intensity distribution to visualize data pre and post normalizatio

```
## Pre-normalized density plot of log-intensity distribution
hist(PHBI.data, col = color.palette, main = "Pre-Normalized Density Plot of log-Intensity Distribution")
```



```
## Normalized density plot of log-intensity distribution
hist(PHBI.norm, col = color.palette, main = "Normalized Density Plot of log-Intensity Distribution")
```



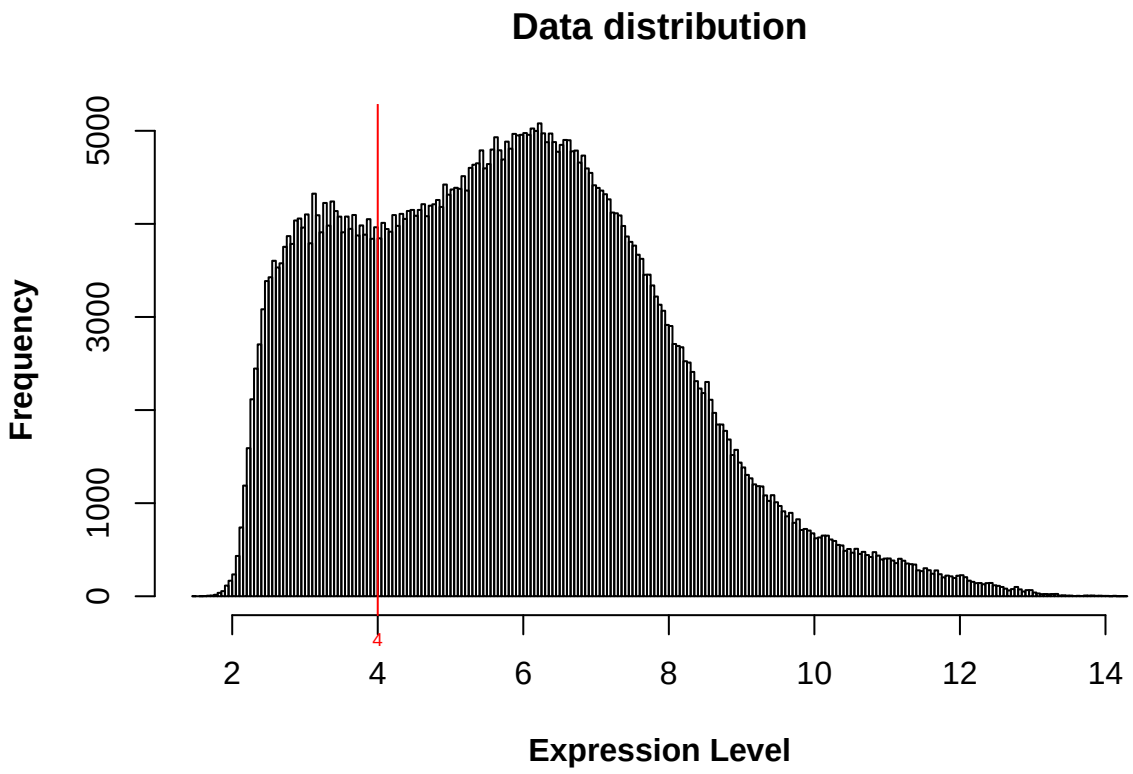
Filtering Process

Extract normalized expression values and perform filtering to discard probesets below a specified threshold

```
## Extract expression values after normalization, verify dimensions
PHBI.exprs = exprs(PHBI.norm)
dim(PHBI.exprs)
```

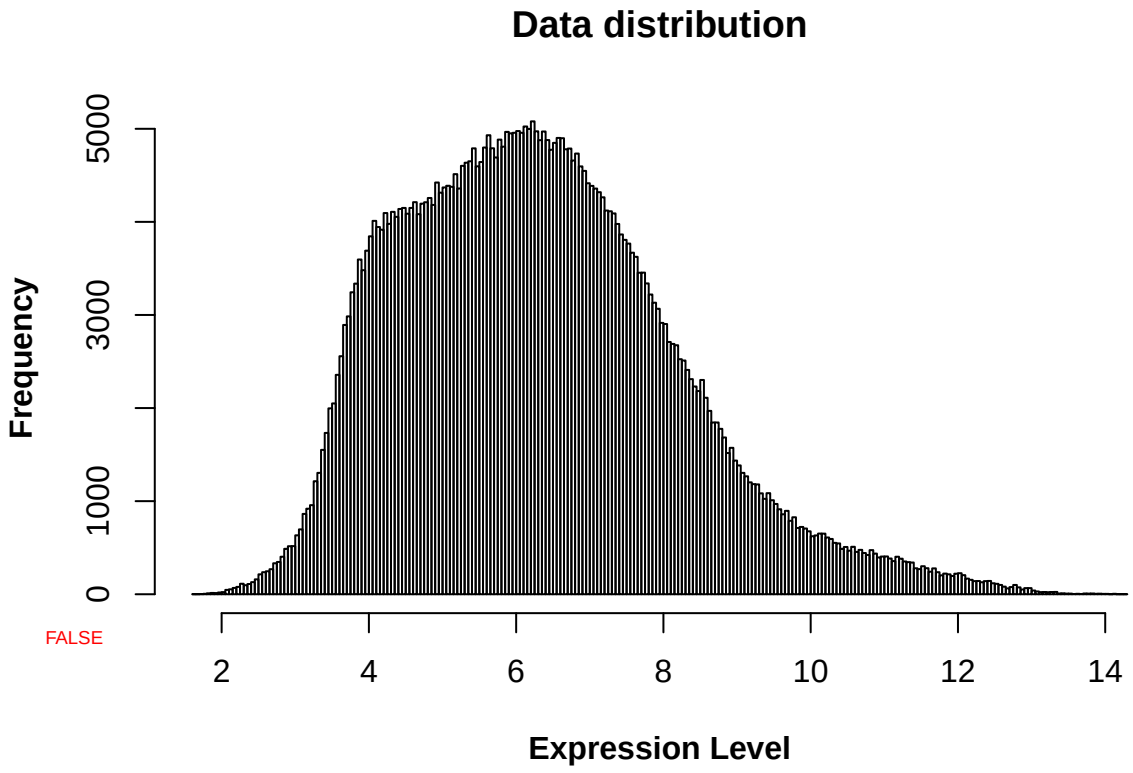
```
## [1] 33297    17
```

```
## Filter and discard probesets with a maximum log2 expression value below 4,
## p=0.01
PUBHI.f = expFilter(PHBI.exprs, threshold = 4)
```




```
## Keep probes with at least 1 sample(s) with an expression level higher than 4
```

```
## View data ditribution after filtering; remove threshold line
PUBHI.A = expFilter(PUBHI.f, threshold = F)
```

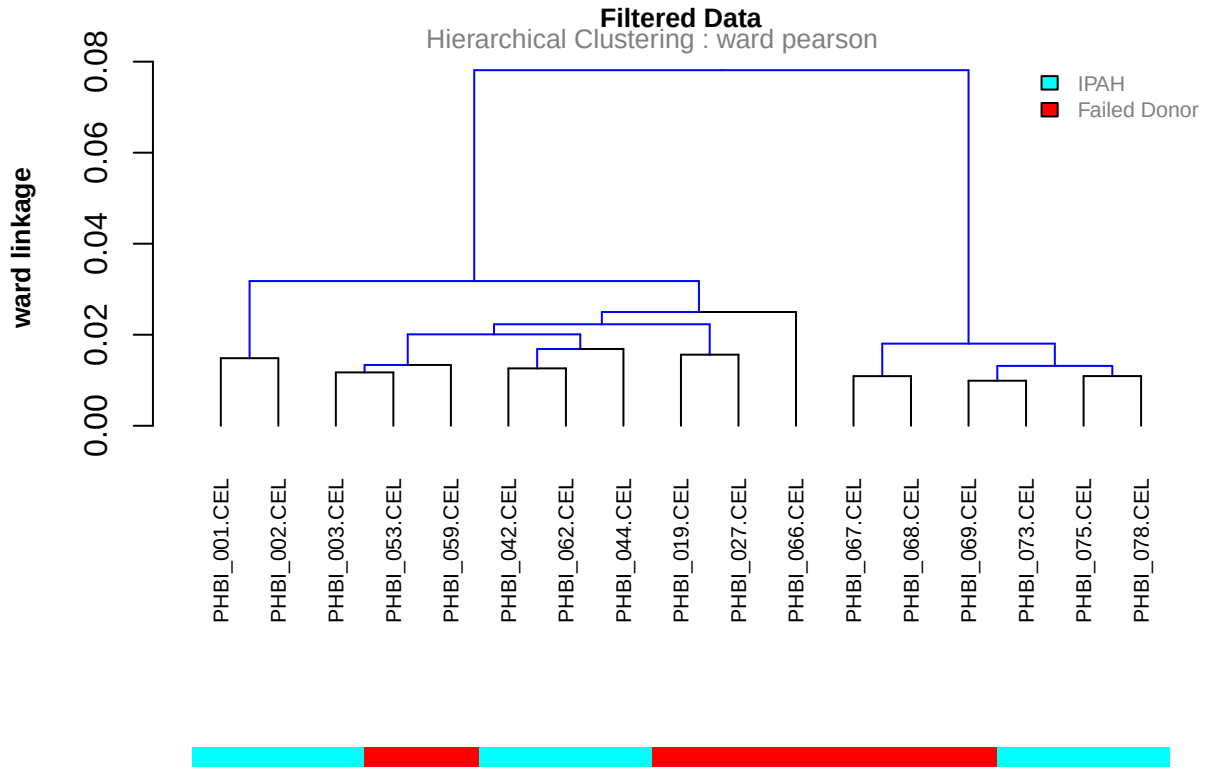


```
## Keep probes with at least 1 sample(s) with an expression level higher than FALSE
```

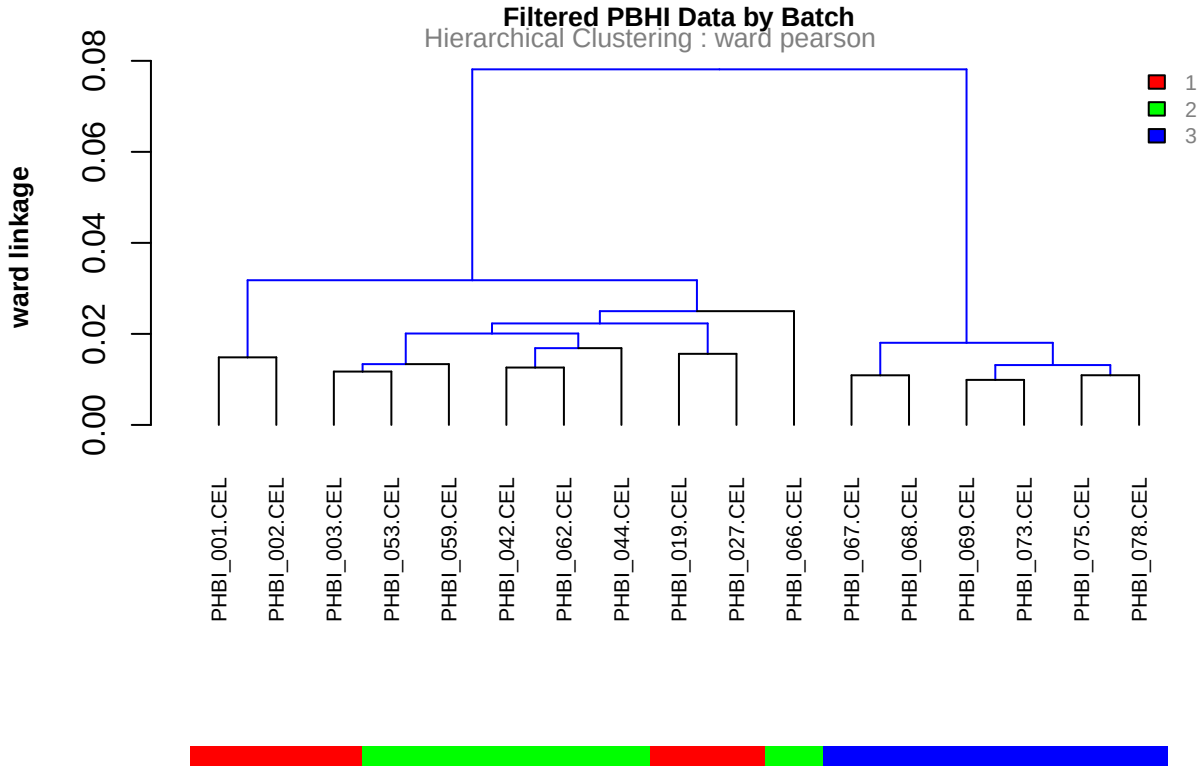
Hierarchical Clustering

```
PUBHI.sample = clustering(data = PUBHI.f, metric = "pearson", method = "ward")

## Visualize data pre and post filtering
clustering.plot(tree = PUBHI.sample, lab = PBHI.type.cl, title = "Filtered Data")
```



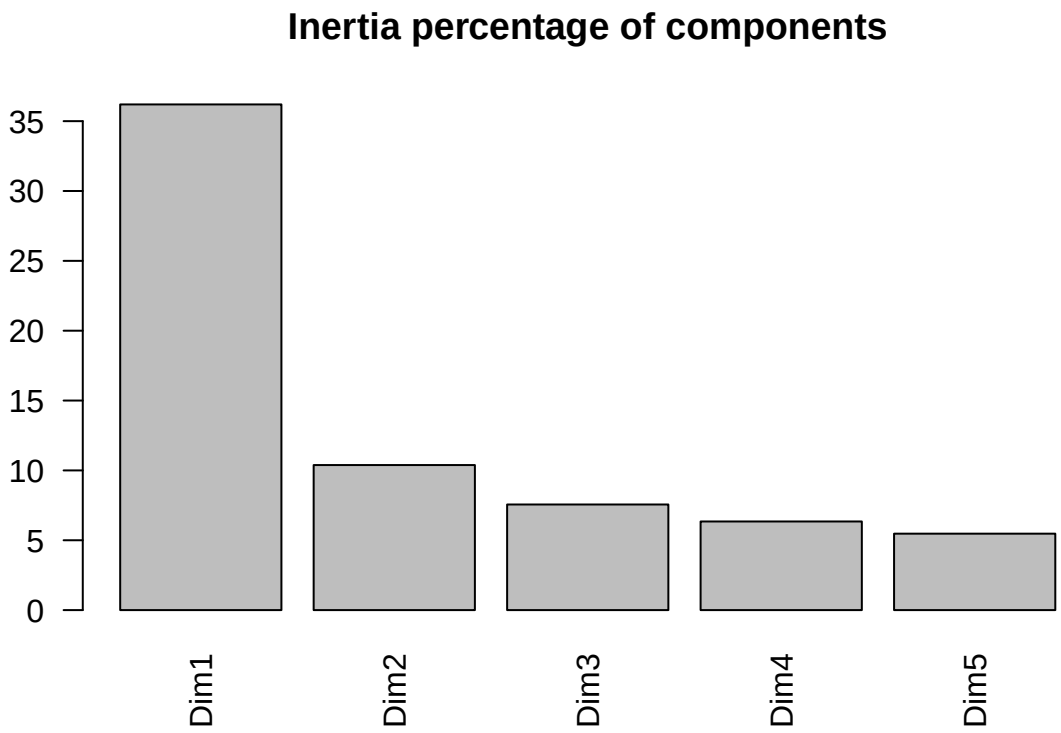

```
clustering.plot(tree = PUBHI.sample, lab = PBHI.batch.cl, title = "Filtered PBHI Data by Batch")
```



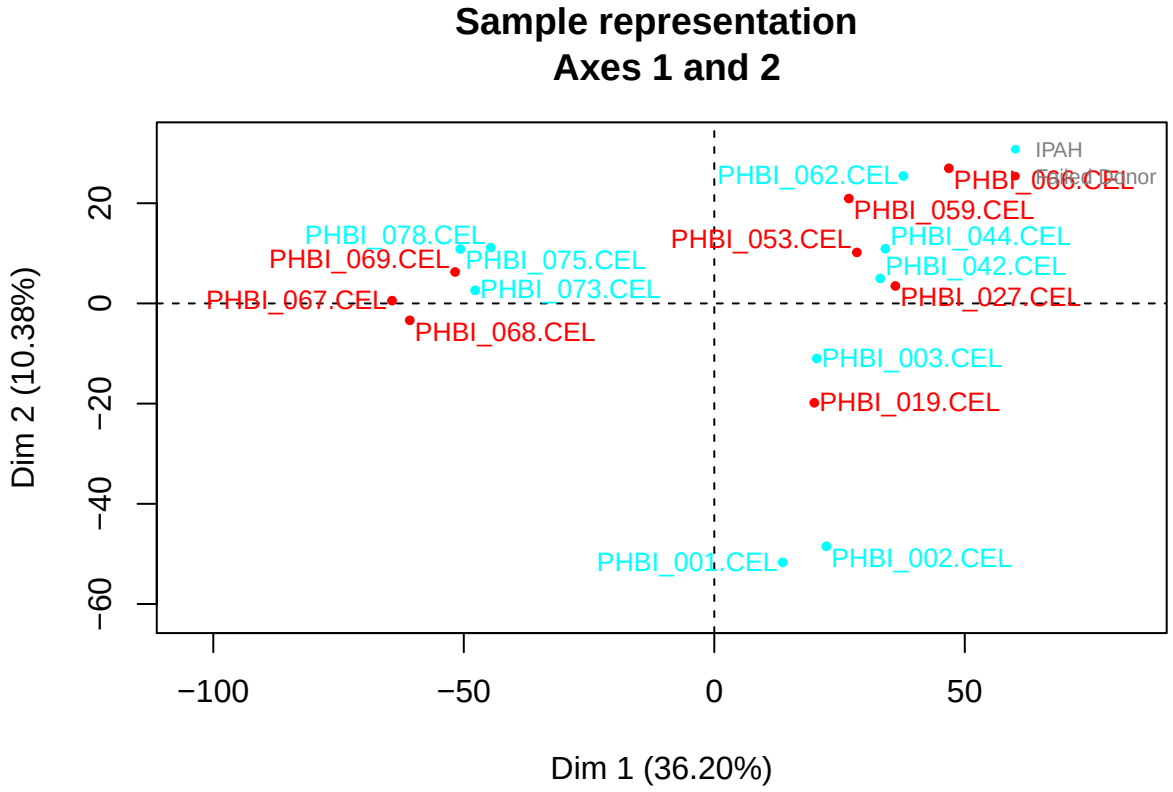
Principal Component Analysis

Run Principle Component Analysis (PCA) on the normalized and filtered data

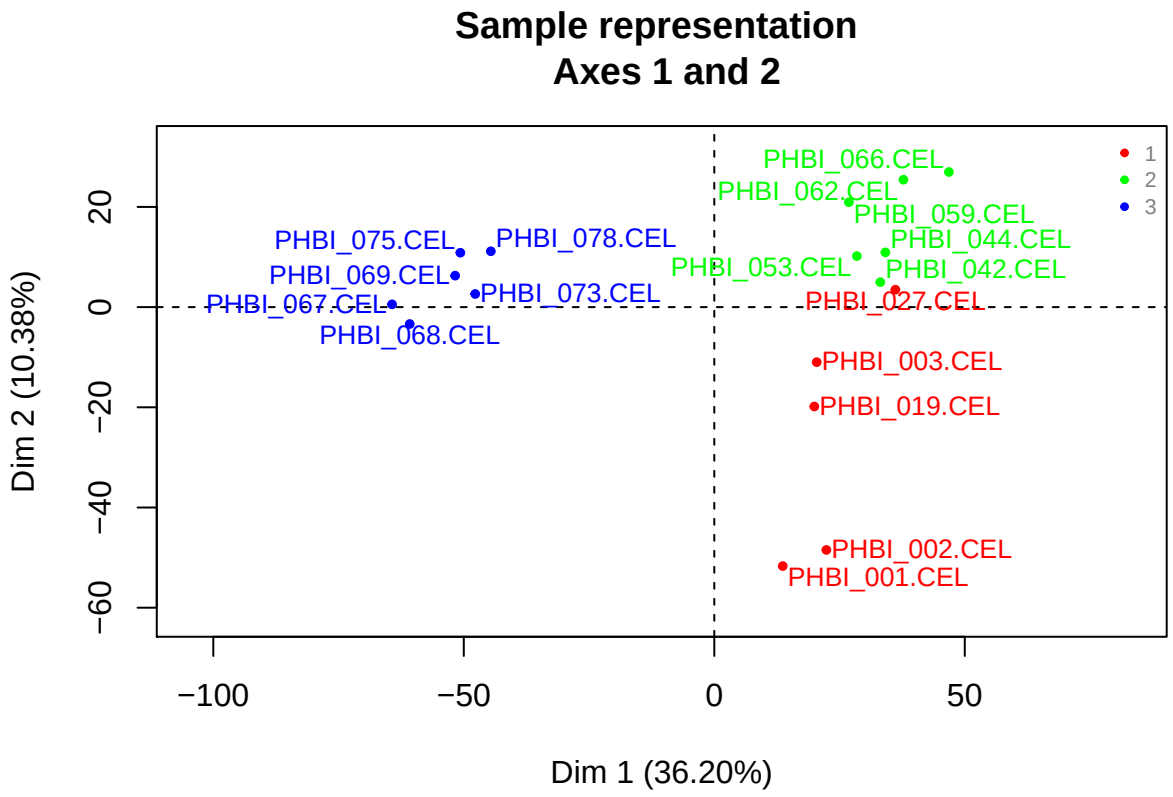
```
## Segment variation into different components
acp = runPCA(t(PUBHI.f), scale = FALSE, lab.sample = PBHI.type.cl, plotSample = FALSE,
    plotInertia = FALSE)
plotInertia(acp)
```




```
## View pre-batch correction individual maps (axes 1 and 2) to look at the
## variation between groups and batches
plotSample(acp, axes = c(1, 2), lab = PBHI.type.cl)
```



```
plotSample(acp, axes = c(1, 2), lab = as.character(PBHI.batch.cl))
```




```
## Create pdf report of PCA with selected plots
acp = runPCA(t(PUBHI.f), scale = FALSE, pdfname = "PCA.pdf", lab.sample = PBHI.type.cl)
```

Batch Correction

Perform batch correction on the normalized data and re-run PCA to show post-batch correction improvements

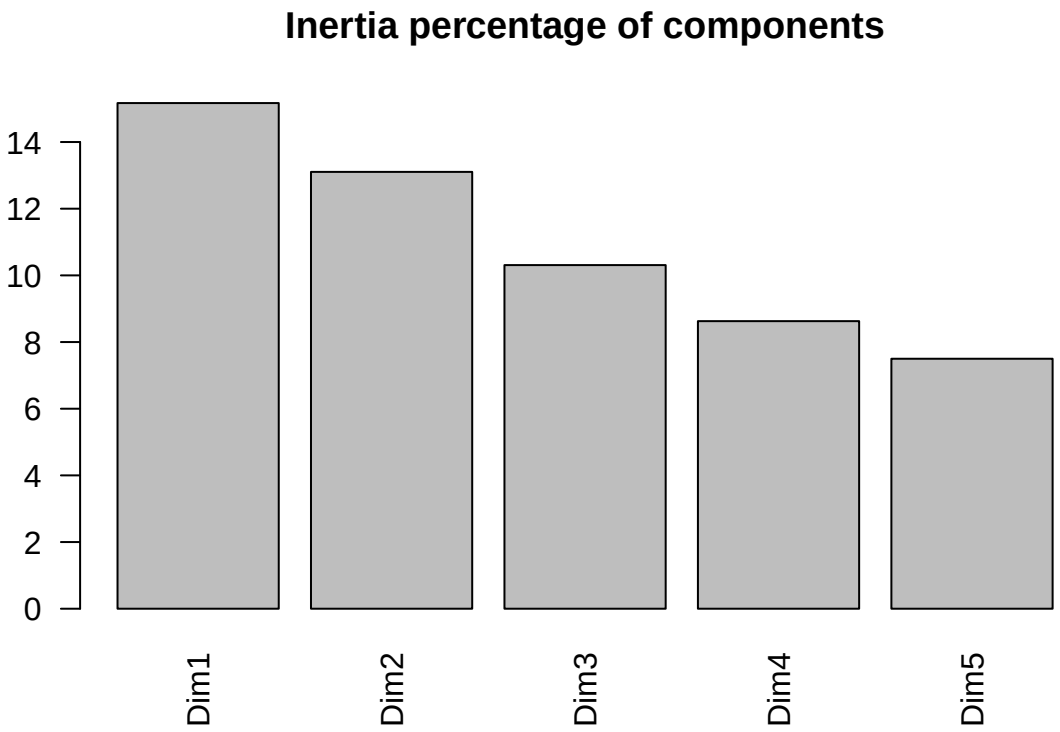
```
# Load sva library
library(sva)

## Create experimental design object and perform batch correction
mod = model.matrix(~as.factor(PBHI.type.cl))
combat_edata = ComBat(dat = PUBHI.f, batch = PBHI.batch.cl, mod = mod, numCovs = NULL,
    par.prior = TRUE, prior.plots = FALSE)
```

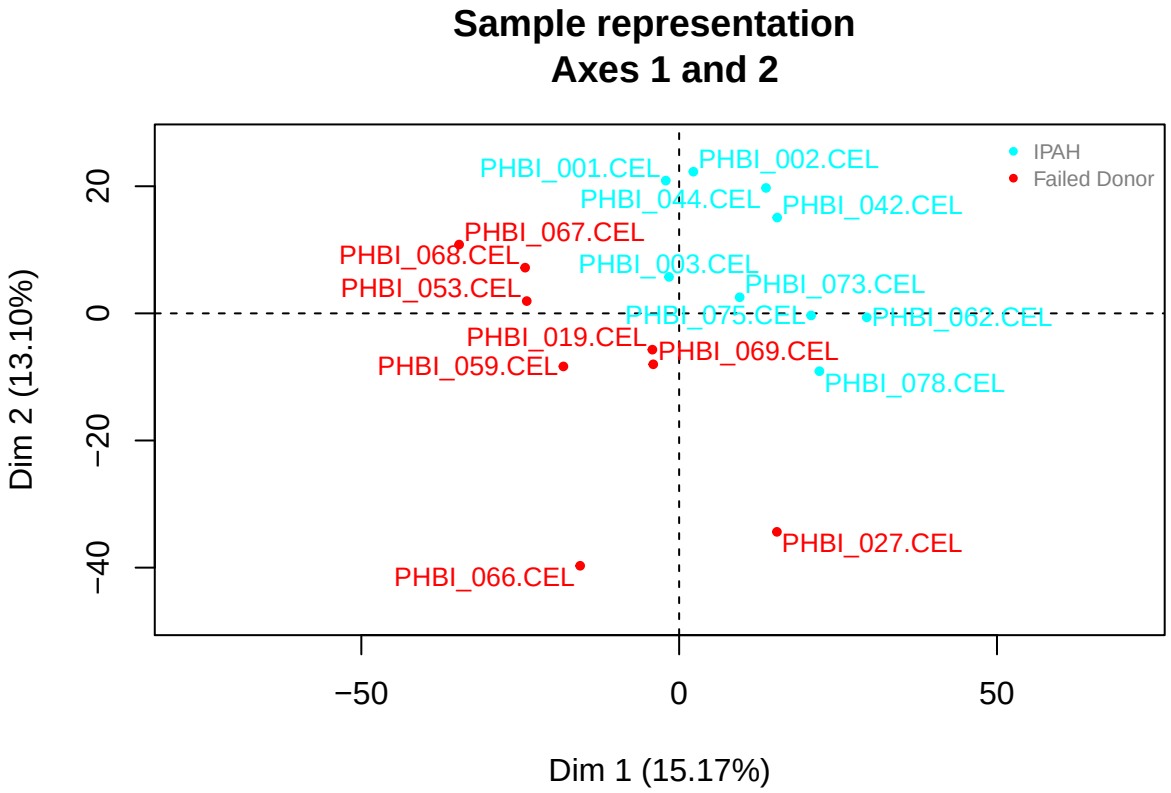
```
## Found 3 batches
## Found 1  categorical covariate(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
```

Run PCA and Hierarchical Clustering again after batch adjustment

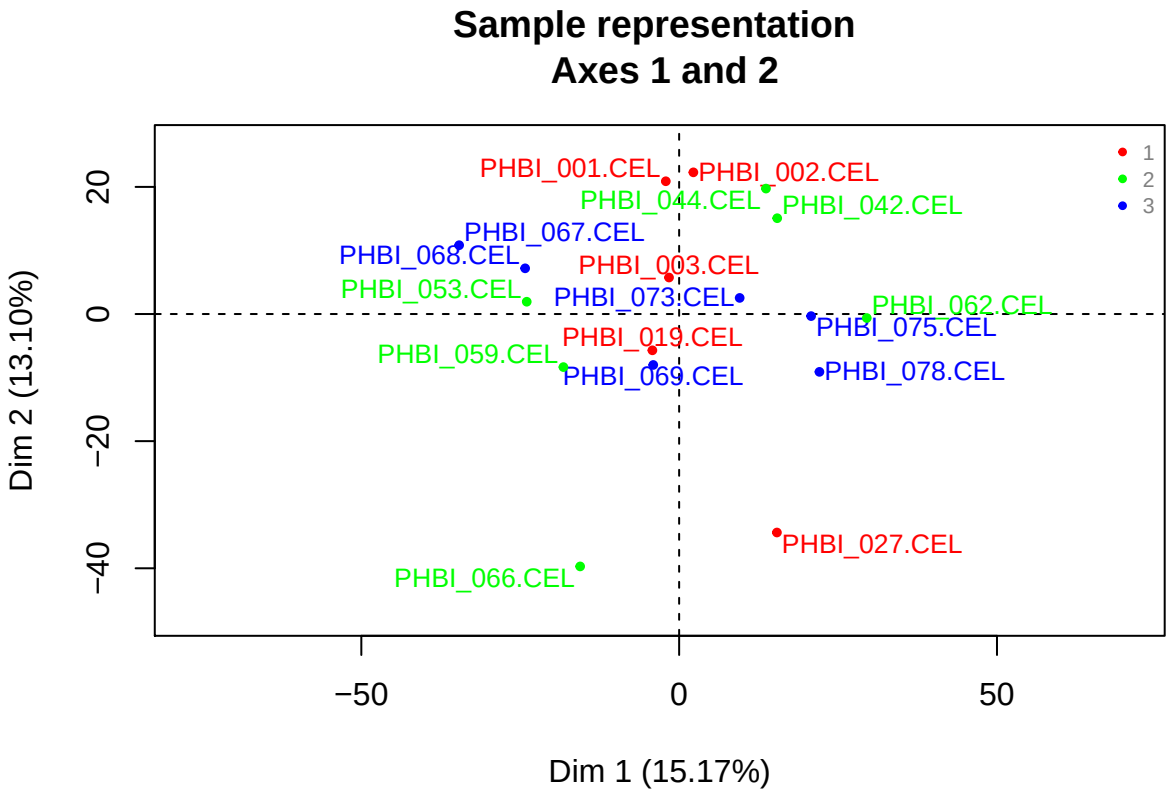
```
## Run PCA after batch correction
acp = runPCA(t(combat_edata), scale = FALSE, lab.sample = PBHI.type.cl, plotSample = FALSE, plotInertia
plotInertia(acp)
```



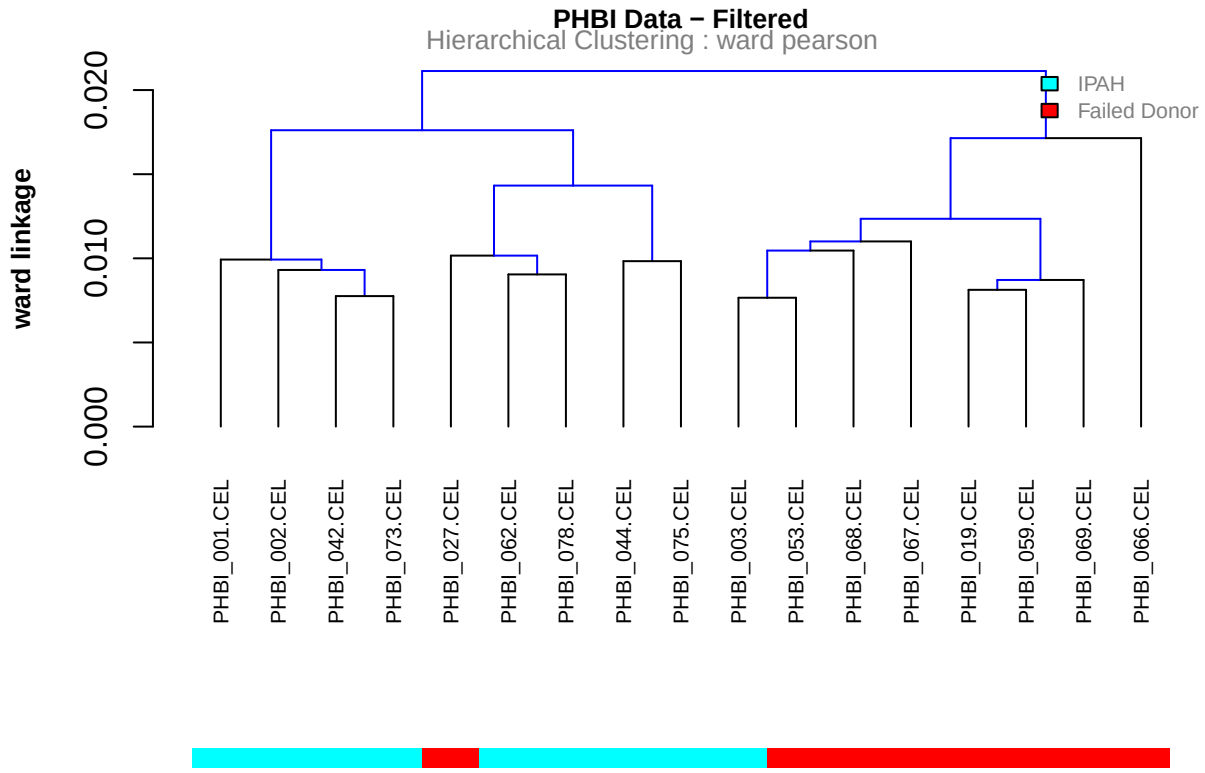

```
## View post-batch correction individual maps (axes 1 and 2) to look at the
## variation between groups and batches
plotSample(acp, axes = c(1, 2), lab = PBHI.type.cl)
```



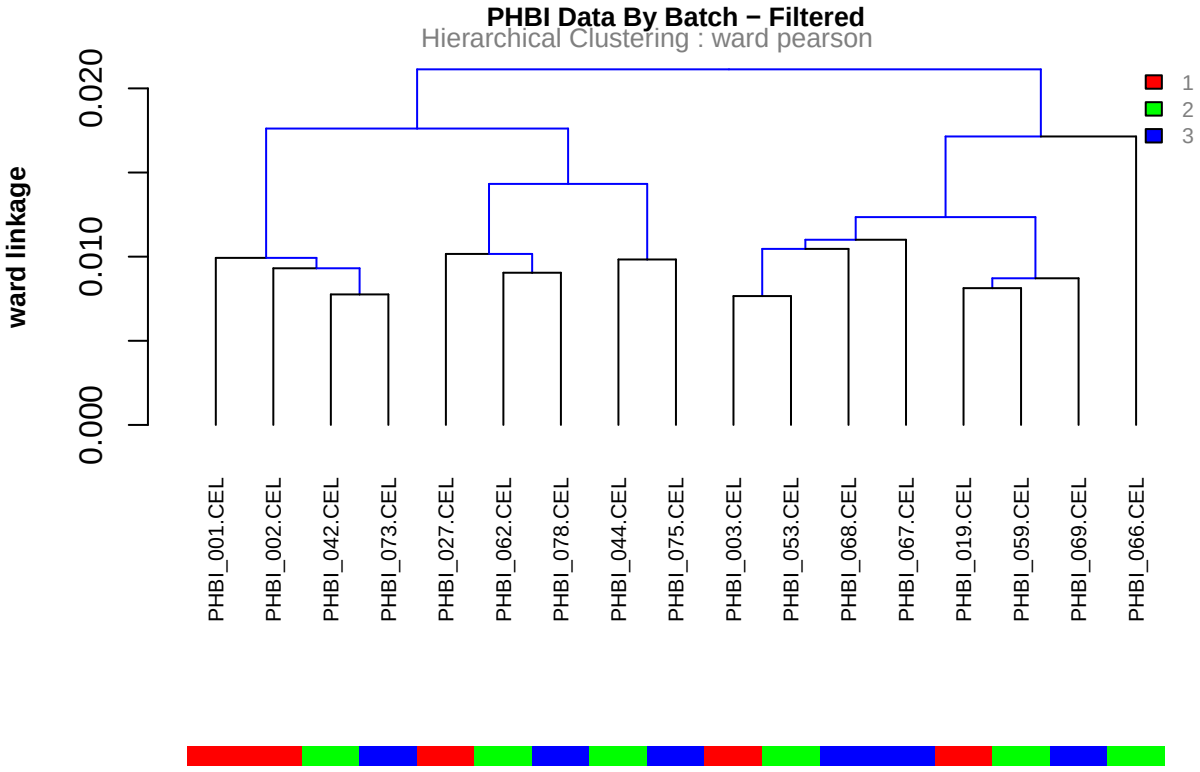

```
plotSample(acp, axes = c(1, 2), lab = PBHI.batch.cl)
```



```
## Sample Hierarchical Clustering (post-batch adjustment)
PUBHI.sample2 = clustering(data = combat_edata, metric = "pearson", method = "ward")
clustering.plot(tree = PUBHI.sample2, lab = PBHI.type.cl, title = "PHBI Data - Filtered")
```




```
clustering.plot(tree = PUBHI.sample2, lab = PBHI.batch.cl, title = "PHBI Data By Batch - Filtered")
```



Statistical Analysis

Run Student's t-test and Significance of Analysis of Microarrays (SAM) test on normalized data. Merge the test results into one data set to produce a heatmap

```
## Set-up Student's t-test comparison factor
PUBHI.type.num = ifelse(PBHI.type.cl == "IPAH", 0, 1)

## Run Student's t-test with batch corrected data
PUBHI.ttest = runTtest(combat_edata, labels = PUBHI.type.num, algo = "t.equalvar",
    q = 0.2, plot = FALSE)
```

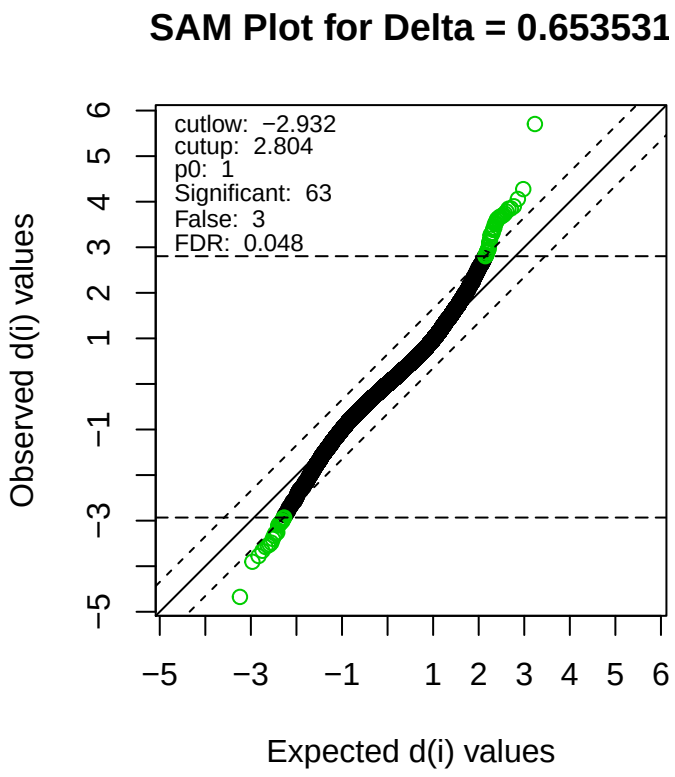
```
## [1] "typeFDR= FDR-BH"
```

```
## head(PUBHI.ttest)
knitr::kable(head(PUBHI.ttest))
```

probeID	Stat	RawpValue	AdjpValue
7892501	-0.8986333	0.3830467	0.7424598
7892502	3.8840718	0.0014682	0.0844791
7892503	0.0945693	0.9259087	0.9800601
7892504	-0.5491785	0.5909654	0.8574125
7892505	0.1255285	0.9017721	0.9729966
7892506	-0.7403385	0.4705250	0.7958579

```
## Run the SAM test with batch corrected data
PUBHI.SAM = runSAM(PUBHI.f, labels = PUBHI.type.num)
```

```
##
## We're doing 24310 complete permutations
## and randomly select 500 of them.
##
##    Delta p0 False Called        FDR
## 1   0.10  1 820.5   1473 0.55702648
## 2   0.15  1 458.5    988 0.46406883
## 3   0.20  1 253.0    677 0.37370753
## 4   0.25  1 154.0    513 0.30019493
## 5   0.30  1  92.5    381 0.24278215
## 6   0.35  1  61.0    301 0.20265781
## 7   0.40  1  38.0    229 0.16593886
## 8   0.45  1  22.0    163 0.13496933
## 9   0.50  1  16.0    140 0.11428571
## 10  0.55  1   9.0    104 0.08653846
## 11  0.60  1   5.0     80 0.06250000
## 12  0.65  1   4.0     67 0.05970149
## 13  0.70  1   3.0     52 0.05769231
## 14  0.75  1   1.0     43 0.02325581
## 15  0.80  1   1.0     42 0.02380952
## 16  0.85  1   1.0     38 0.02631579
## The threshold seems to be at
##      Delta Called      FDR
## 5 0.653530     67 0.059701
## 6 0.653531     63 0.047619
## [1] "Delta : 0.653531"
```



```
## [1] "Find 63 significant genes ..."
```

```
knitr::kable(head(PUBHI.SAM))
```

	probeID	Stat	RawpValue	FoldChange	Significant
7892501	7892501	-0.2580086	0.7042239	0.9007307	FALSE
7892502	7892502	1.4187845	0.0481135	1.2360180	FALSE
7892503	7892503	0.0547860	0.9355328	1.0114994	FALSE
7892504	7892504	-0.1685861	0.8038497	0.9657320	FALSE
7892505	7892505	-0.1104603	0.8705780	0.9752623	FALSE
7892506	7892506	-0.6230634	0.3644616	0.8565688	FALSE
Reorganize	the resul	t			

```
## Sort genes in both datasets by the 'probeID' variable
PUBHI.ttest.sig = PUBHI.ttest[order(PUBHI.ttest$probeID), ]
PUBHI.SAM.sig = PUBHI.SAM[order(PUBHI.SAM$probeID), ]

## Extract the 'probeID' and 'FoldChange' variables from the PBHI.SAM.sig
## data set
FC.raw = PUBHI.SAM.sig[, c("probeID", "FoldChange")]

## Merge the 'FoldChange' Variable into PUBHI.ttest.sig database by probeID
PUBHI.test.merge = merge(PUBHI.ttest.sig, FC.raw, by = "probeID", all.y = TRUE,
    all.x = TRUE)
knitr::kable(head(PUBHI.test.merge))
```

probeID	Stat	RawpValue	AdjpValue	FoldChange
7892501	-0.8986333	0.3830467	0.7424598	0.9007307
7892502	3.8840718	0.0014682	0.0844791	1.2360180
7892503	0.0945693	0.9259087	0.9800601	1.0114994
7892504	-0.5491785	0.5909654	0.8574125	0.9657320
7892505	0.1255285	0.9017721	0.9729966	0.9752623
7892506	-0.7403385	0.4705250	0.7958579	0.8565688

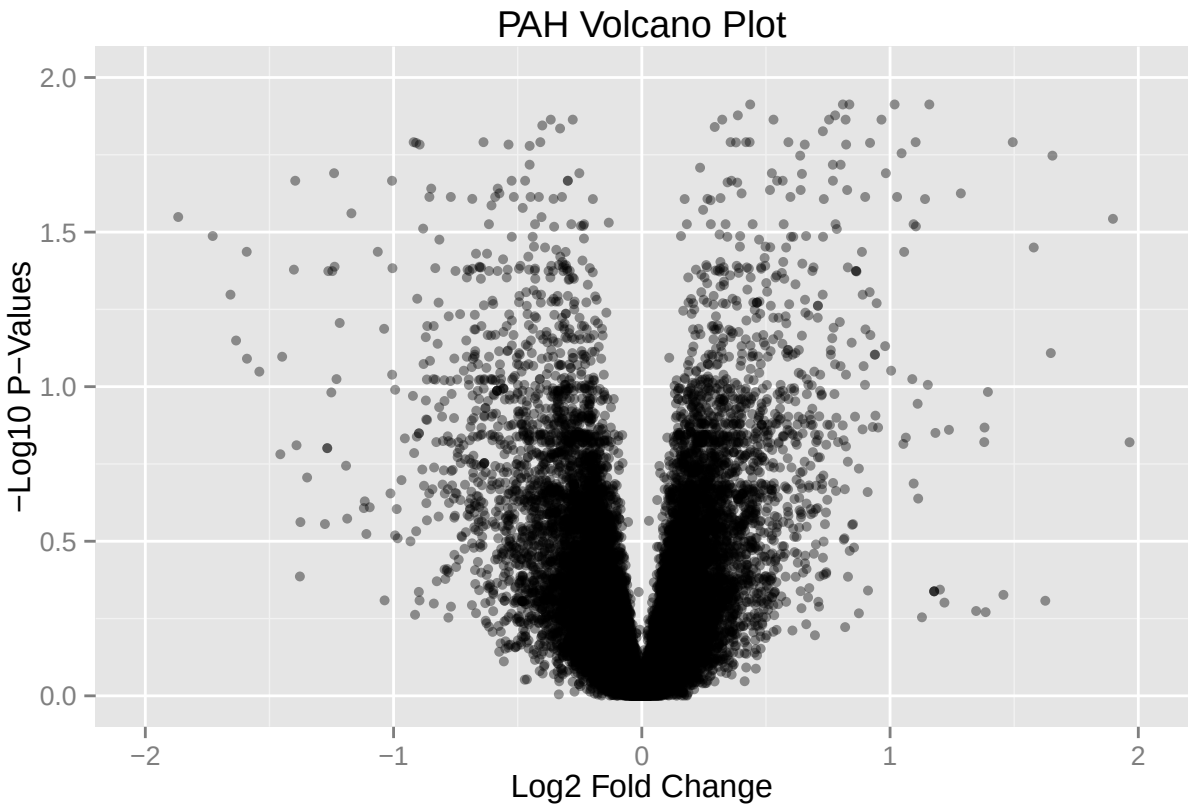
```
## Re-sort merged data by the 'AdjpValue' variable
PUBHI.test.merge = PUBHI.test.merge[order(PUBHI.test.merge$AdjpValue), ]
knitr::kable(head(PUBHI.test.merge))
```

	probeID	Stat	RawpValue	AdjpValue	FoldChange
5913	7922174	8.691899	3.0e-07	0.0060968	1.5718386
5973	7922793	8.181010	7.0e-07	0.0060968	1.2188873

	probeID	Stat	RawpValue	AdjpValue	FoldChange
26312	8163908	-8.285346	6.0e-07	0.0060968	0.5022629
10129	7973110	7.784296	1.2e-06	0.0084292	4.4227513
8924	7958174	7.402996	2.2e-06	0.0088352	1.7526347
15130	8031207	7.464315	2.0e-06	0.0088352	2.4424002

Volcano Plot

```
## Produce a volcano Plot to verify SAM findings
volcano = ggplot(data = PUBHI.test.merge, aes(x = log2(PUBHI.test.merge$FoldChange),
    y = -log10(PUBHI.test.merge$AdjpValue)), colour = none) + geom_point(alpha = 0.4,
    size = 1.75) + labs(title = "PAH Volcano Plot") + xlim(c(-2, 2)) + ylim(c(0,
    2)) + xlab("Log2 Fold Change") + ylab("-Log10 P-Values")
volcano
```

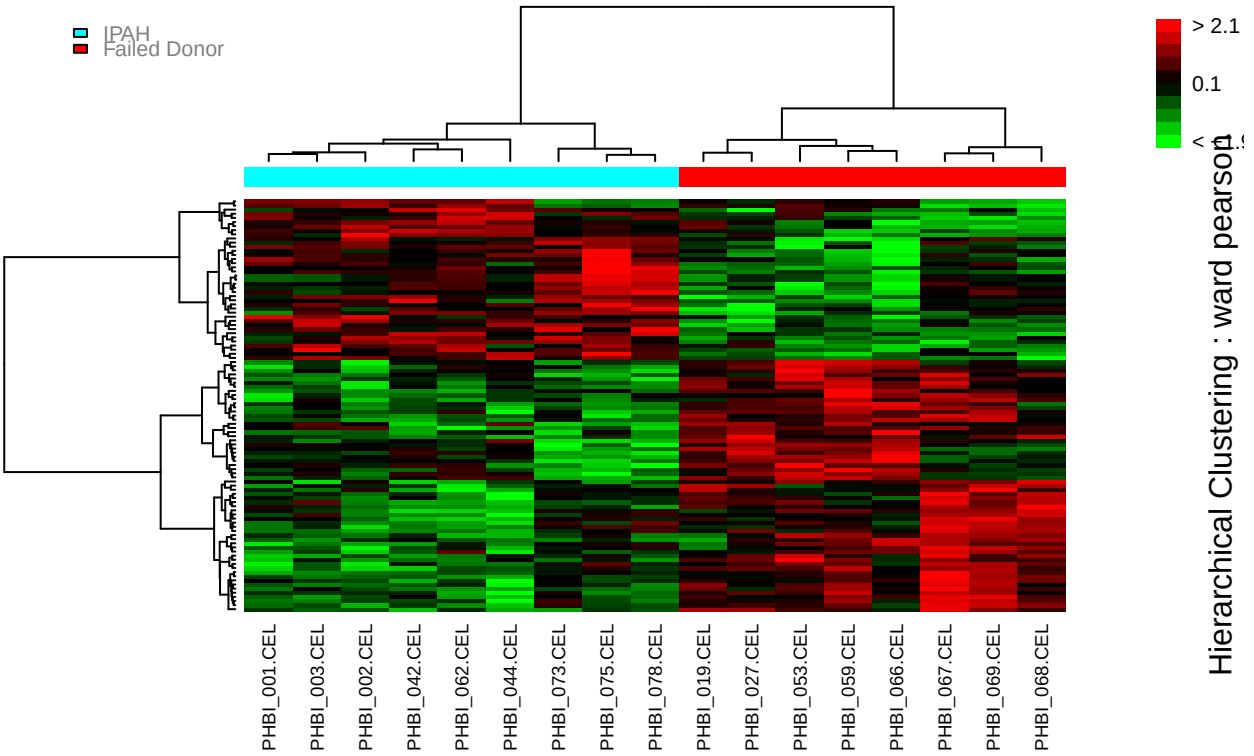


Output Analysis Result

Produce a heatmap of the top 100 significant genes

```
## Produce Heatmap
mvgenes = as.character(PUBHI.test.merge$probeID[1:100])
c.sample <- clustering(data = PUBHI.f[mvgenes, ], metric = "pearson", method = "ward")
```

```
c.gene <- clustering(data = t(PUBHI.f[mvgenes, ]), metric = "pearson", method = "ward")
clustering.plot(tree = c.sample, tree.sup = c.gene, data = PUBHI.f[mvgenes,
    ], names.sup = FALSE, lab = PBHI.type.cl, trim.heatmap = 0.99)
```



Hierarchical Clustering : ward pearson

Annotate Significant Genes

Annotate the top 250 significant genes, output to a text file. Input cluster analysis output from DAVID and print the first three rows

```
options(width = 500)
## Load annaffy and hugene10sttranscriptcluster.db libraries
library(annaffy)
library(hugene10sttranscriptcluster.db)

## Annotate the top 250 significant genes
anntable = aafTableAnn(as.character(PUBHI.test.merge$probeID[1:250]), "hugene10sttranscriptcluster.db")

## Add the 'AdjpValue' and 'FoldChange' variables to the annotation table
atable = aafTable(`P-Value` = PUBHI.test.merge$AdjpValue[1:250], signed = TRUE)
FCtable = aafTable(`Fold Change` = PUBHI.test.merge$FoldChange[1:250], signed = TRUE)
table = merge(anntable, atable)
table2 = merge(table, FCtable)

## Export results to an HTML and text file
saveHTML(table2, file = "PBHI.psig.genes.htm")
```

```
saveText(table2, file = "PBHI.psig.genes.txt")

## Print top 7 significant genes at p  0.01 level
#annot.output = read.table(file = "annot.sig.txt", header = T, sep = "\t", nrows = 7)
#annot.outputs = annot.output[order(annot.output$P.Value), ]
#knitr::kable(annot.outputs)
```

R session info

```
sessionInfo()
```

```
## R version 3.1.2 (2014-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] hugene10sttranscriptcluster.db_8.2.0
##  [2] org.Hs.eg.db_3.0.0
##  [3] annaffy_1.38.0
##  [4] KEGG.db_3.0.0
##  [5] GO.db_3.0.0
##  [6] AnnotationDbi_1.28.2
##  [7] GenomeInfoDb_1.2.5
##  [8] sva_3.12.0
##  [9] genefilter_1.48.1
## [10] mgcv_1.8-6
## [11] nlme_3.1-120
## [12] RColorBrewer_1.1-2
## [13] pd.hugene.1.0.st.v1_3.10.0
## [14] RSQLite_1.0.0
## [15] DBI_0.3.1
## [16] ggplot2_1.0.1
## [17] oligo_1.30.0
## [18] Biostrings_2.34.1
## [19] XVector_0.6.0
## [20] IRanges_2.0.1
## [21] S4Vectors_0.4.0
## [22] Biobase_2.26.0
## [23] oligoClasses_1.28.0
## [24] BiocGenerics_0.12.1
## [25] EMA_1.4.4
##
## loaded via a namespace (and not attached):
##  [1] affxparser_1.38.0     affy_1.44.0           affyio_1.34.0
##  [4] annotate_1.44.0       BiocInstaller_1.16.2  biomaRt_2.22.0
##  [7] bit_1.1-12            bitops_1.0-6          car_2.0-25
## [10] cluster_2.0.1         codetools_0.2-11      colorspace_1.2-6
```

```
## [13] digest_0.6.8          evaluate_0.5.5        FactoMineR_1.29
## [16] ff_2.2-13             flashClust_1.01-2     foreach_1.4.2
## [19] formatR_1.1           gcrma_2.38.0          GenomicRanges_1.18.4
## [22] grid_3.1.2            GSA_1.03              gtable_0.1.2
## [25] heatmap.plus_1.3      htmltools_0.2.6       iterators_1.0.7
## [28] knitr_1.9             lattice_0.20-31       leaps_2.9
## [31] lme4_1.1-7            MASS_7.3-40           Matrix_1.2-0
## [34] minqa_1.2.4           multtest_2.22.0       munsell_0.4.2
## [37] nloptr_1.0.4          nnet_7.3-9            pbkrtest_0.4-2
## [40] plyr_1.8.1            preprocessCore_1.28.0 proto_0.3-10
## [43] quantreg_5.11         Rcpp_0.11.5           RCurl_1.95-4.5
## [46] reshape2_1.4.1        rmarkdown_0.5.1       scales_0.2.4
## [49] scatterplot3d_0.3-35  siggenes_1.40.0       SparseM_1.6
## [52] splines_3.1.2         stringr_0.6.2         survival_2.38-1
## [55] tools_3.1.2           XML_3.98-1.1          xtable_1.7-4
## [58] yaml_2.1.13           zlibbioc_1.12.0
```