Analysis of the enrichment and epigenomic similarity results: GWAS catalog example

Mikhail Dozmorov

February 2, 2014

What will be done

- Disease- and trait-specific sets of SNPs from GWAScatalog will be analyzed for the enrichment in 161 transcription factor binding sites (TFBSs) using GenomeRunner Web;
- The results of the analysis will be loaded into R and pre-processed;
- Although GenomeRunner Web has default visualization strategy, we will explore and tweak visualization of the epigenomic similarity results;
- We will answer a question which epigenomic elements are differentially enriched in different clusters defined by the epigenomic similarity analysis;
- We will identify most strongly epigenomically correlated and anticorrelated sets of SNPs;
- We will visualize and tweak the enrichment analysis results.

Data analysis and preparation

The results used in this tutorial have been obtained with GenomeRunner Web. The details of the analysis and the BED files are available on github https://github.com/mdozmorov/gwas2bed. Briefly, BED files having at least 15 disease- or trait-associated SNP were tested for enrichment in 161 experimentally obtained transcription factor binding sites (tfbsEncode). Download and extract the .tar.gz results file, or use them from the `/data/more15_vs_tfbsEncode` folder.

Load the libraries

```r
source("utils.R")
suppressMessages(library(Hmisc))  # For rcorr function
suppressMessages(library(gplots))
suppressMessages(library(Biobase))
suppressMessages(library(limma))
```

The enrichment analysis results are outputted in the `matrix.txt` file, stored as the enrichment p-values with a ”-” sign added to denote depletion. We transform the raw p-values using -log10 transformation, and keep the ”-” sign for the case of depletion.

```r
# Define data subfolder to use, change to analyze different data
dname <- "data//more15_vs_tfbsEncode//"
mtx <- as.matrix(read.table(paste(dname, "matrix.txt", sep = ""),
    sep = "\t", header = T, row.names = 1))
mtx <- mtx.transform(mtx)  # -log10 transform p-values
```

Our matrix now contains the transformed p-values, negative in case of depletion. Rows are the TFBSs names, columns are the names of disease- or trait-associated SNP sets. We filter remove the row if a TFBS does not show enrichment in any of the SNP sets. The `cutoff<-2` on -log10-transformed scale corresponds to the 0.01 p-value cutoff. If a SNP set shown no enrichments in any of the TFBS, we remove the column as well. Check the final dimensions before actually trim the matrix.

```r
dim(mtx)  # Check original dimensions

## [1] 161 251

cutoff <- 2  # raw p-value significance cutoff is 0.01
# What remains if we remove rows/cols with nothing significant
dim(mtx[apply(mtx, 1, function(x) {
    sum(abs(x) > cutoff)
}) > 0, apply(mtx, 2, function(x) {
    sum(abs(x) > cutoff)
}) > 0])

## [1] 119  68

# Trim the matrix
mtx <- mtx[apply(mtx, 1, function(x) {
    sum(abs(x) > cutoff)
}) > 0, apply(mtx, 2, function(x) {
    sum(abs(x) > cutoff)
}) > 0]
```

Visualizing epigenomic similarity results

Epigenomic similarity analysis groups SNP sets by similarity of their enrichemt profiles, or sets of the transformed p-values. In our matrix, SNP set-specific enrichment profiles are columns. We obtain a NxN square matrix of Spearman correlation coefficients, where N is the number of SNP sets.

```
# rcorr returns a list, [[1]] - correl coeffs, [[3]] - p-values.
# Type - pearson/spearman
mtx.cor <- rcorr(as.matrix(mtx), type = "spearman")
```

We visualzie the matrix returned by *rcorr* as a clustered heatmap (Figure 1). Tweak distance and clustering parameters, or use a code snippet from `01_heatmap_corr.R` script for automated testing of combinations of them.

```
# Adjust clustering parameters.  Distance: 'euclidean',
# 'maximum','manhattan', 'canberra', 'binary' or 'minkowski'.
# Clustering: 'ward', 'single', 'complete', 'average', 'mcquitty',
# 'median' or 'centroid'
dist.method <- "euclidean"
hclust.method <- "ward"
h <- heatmap.2(as.matrix(mtx.cor[[1]]), trace = "none", density.info = "none",
    col = color, distfun = function(x) {
        dist(x, method = dist.method)
    }, hclustfun = function(x) {
        hclust(x, method = hclust.method)
    }, cexRow = 0.5, cexCol = 0.5)
```

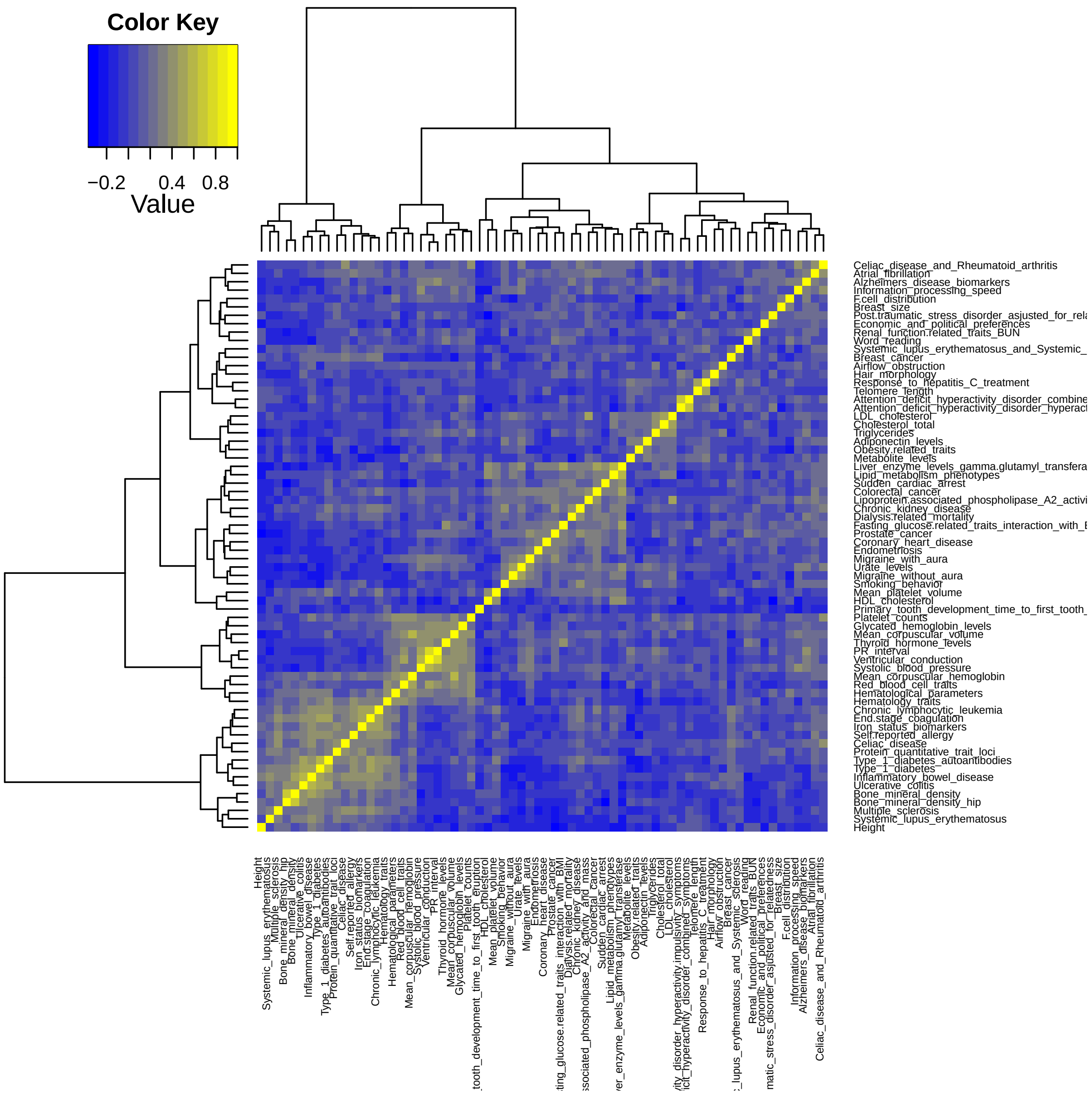
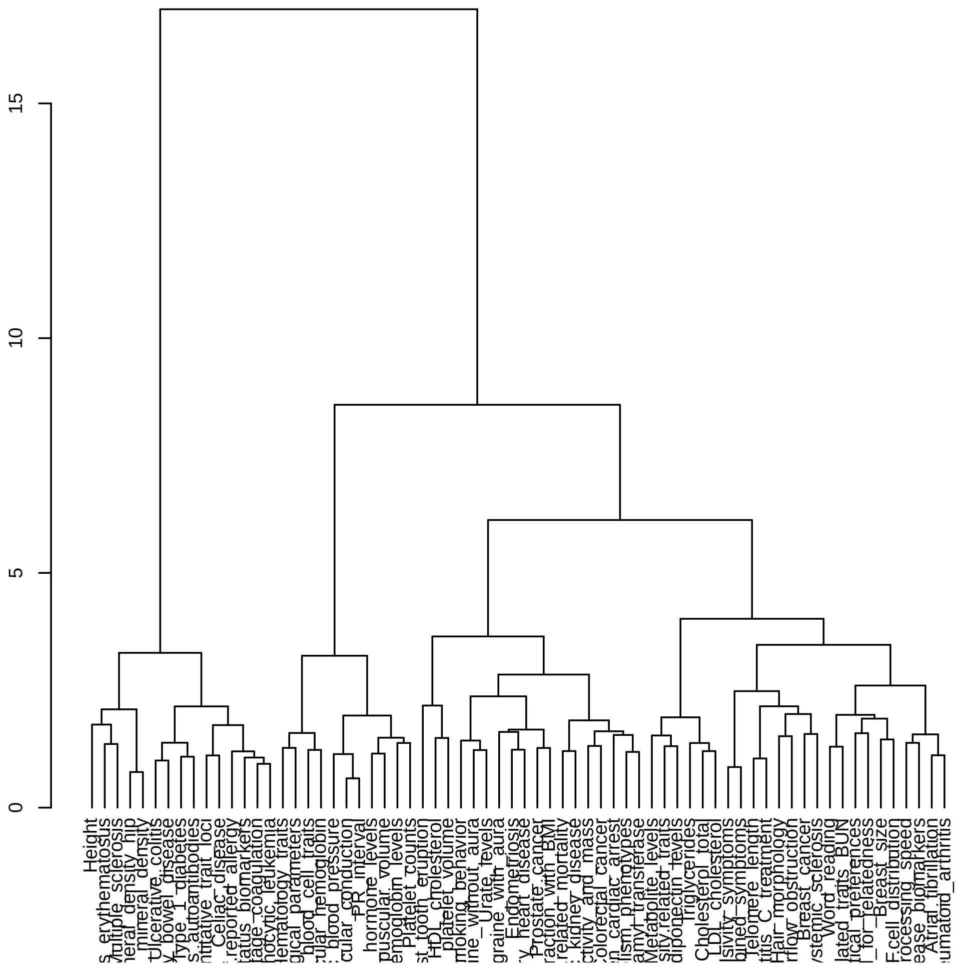


Figure 1: Epigenomic similarity heatmap

Epigenomic differences among the clusters

The heatmap object contains information about the order of clustering. We visualize it as a dendrogram, cut into separate clusters defined by the cut height, and output the cluster ordering in a file.

```r
plot(h$colDendrogram)  # Plot the dendrogram only
```



```r
# Cut the dentrogram into separate clusters. Tweak the height
c <- cut(h$colDendrogram, h = 2.35)
c  # Check the number of clusters, and the number of members
unlink(paste(dname, "clustering.txt", sep = ""))
for (i in 1:length(c$lower)) {
    write.table(paste(i, t(labels(c$lower[[i]])), sep = "\t"), paste(dname,
        "clustering.txt", sep = ""), sep = "\t", col.names = F, row.names = F,
        append = T)
}
```

Remember, our p-values are -log10-transformed. We will test whether these transformed p-values are different between the clusters using the Bioconductor *limma* package.

First, we define the cluster groups and labels. Clusters that contain less than 3 elements are not considered, as we cannot test statistically significant differences with them.

```
eset.labels <- character()  # Empty vector to hold cluster labels
eset.groups <- numeric()  # Empty vector to hold cluster groups
for (i in 1:length(c$lower)) {
    # Go through each cluster If the number of members is more than 2
    if (attr(c$lower[[i]], "members") > 2) {
        eset.labels <- append(eset.labels, labels(c$lower[[i]]))
        eset.groups <- append(eset.groups, rep(i, length(labels(c$lower[[i]]))))
    }
}
```

Then, we perform standard *limma* analysis. That is, we construct an ExpressionSet, define a design matrix, the thresholds, and test each cluster combination for differential expression. We also define the `degs.matrix` matrix to containing just the numbers of differentially expressed epigenomic elements.

The results of *limma* analysis are writtein into the `degs.txt` file.

```
# Make eset out of eset.labels
eset <- new("ExpressionSet", exprs = as.matrix(mtx[, eset.labels]))
# Make model matrix
design <- model.matrix(~0 + factor(eset.groups))
colnames(design) <- paste("c", unique(eset.groups), sep = "")
# Create a square matrix of counts of DEGs
degs.matrix <- matrix(0, length(c$lower), length(c$lower))
# Name rows and cols by cluster name
colnames(degs.matrix) <- paste("c", seq(1, length(c$lower)))
rownames(degs.matrix) <- paste("c", seq(1, length(c$lower)))
cutoff.pval <- 0.05  # p-value cutoff. Tweak
cutoff.lfc <- log(2)  # log2 fold change cutoff. Tweak
for (i in colnames(design)) {
    for (j in colnames(design)) {
        # Contrasts between two clusters
        contrast.matrix <- makeContrasts(contrasts = paste(i, j, sep = "-"),
            levels = design)
        fit <- lmFit(eset, design)
        fit2 <- contrasts.fit(fit, contrast.matrix)
        fit2 <- eBayes(fit2)
    degs <- topTable(fit2, number = dim(exprs(eset))[[1]], adjust.method = "BH",
      p.value = cutoff.pval, lfc = cutoff.lfc)
    print(paste(i, "vs.", j, ", number of degs:", dim(degs)[[1]]))
    # Keep the number of DEGs in the matrix
    degs.matrix[as.numeric(sub("c", "", i)), as.numeric(sub("c",
      "", j))] <- dim(degs)[[1]]
    if (dim(degs)[[1]] > 0) {
      # Average values in clusters i and j
      i.av <- rowMeans(matrix(exprs(eset)[rownames(degs), eset.groups ==
        as.numeric(sub("c", "", i))], nrow = dim(degs)[[1]]))
      j.av <- rowMeans(matrix(exprs(eset)[rownames(degs), eset.groups ==
        as.numeric(sub("c", "", j))], nrow = dim(degs)[[1]]))
      i.vs.j <- rep(paste(i, "vs.", j), dim(degs)[[1]])
      # Put it all together in a file
      write.table(cbind(degs, i.vs.j, i.av, j.av), paste(dname,
        "degs.txt", sep = ""), sep = "\t", col.names = NA,
        append = T)
    }
  }
}
```

Most epigenomically (anti)correlated sets of SNPs

Out of all epigenomic similarity results, the natural question may be: "What pair ofsets of SNPs show the strongest epigenomic similarity?", or "Which disease-specific SNP set most strongly correlates with Multiple Sclerosis SNP set?".

We simply scan each column of the correlation matrix and detect rows corresponding to minimum and maximum correlation coefficients. We want to ignore perfect self-correlations, therefore, we set the diagonal of the matrix containing such self-correlations to 0. We output the results in the `maxmin_correlations.txt` file.

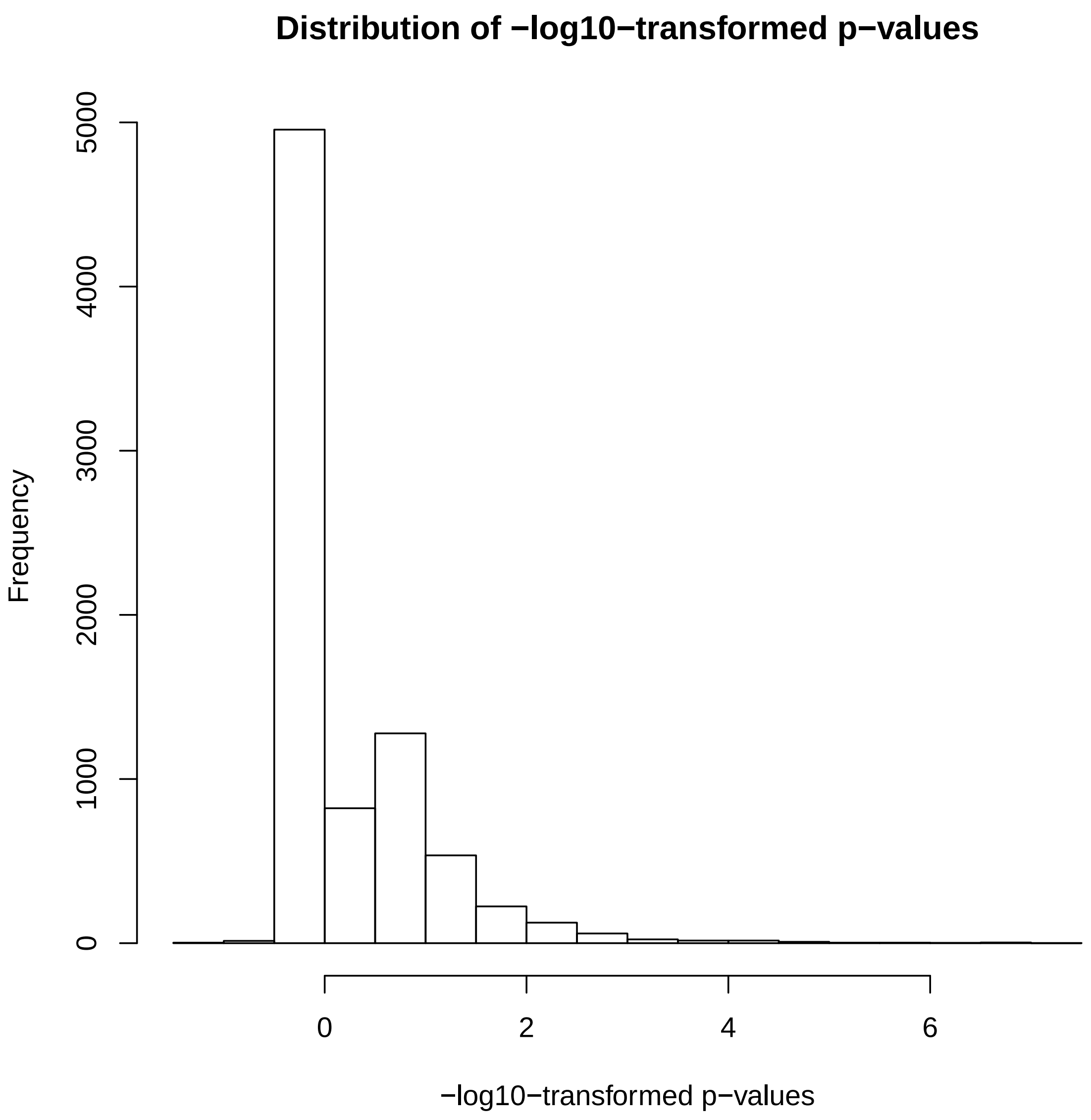
```
mtx.cor1 <- mtx.cor[[1]]
# We don't need to consider perfect correlations, zero them out
diag(mtx.cor1) <- 0
# Print top correlated parameters on screen
for (i in head(unique(mtx.cor1[order(mtx.cor1, decreasing = T)]))) {
  print(which(mtx.cor1 == i, arr.ind = T))
}
unlink(paste(dname, "maxmin_correlations.txt", sep = ""))
for (i in 1:ncol(mtx.cor1)) write.table(paste(colnames(mtx.cor1)[i],
  "correlates with", colnames(mtx.cor1)[which(mtx.cor1[i, ] == max(mtx.cor1[i,
    ]))], "at corr. coeff.", formatC(mtx.cor1[i, which(mtx.cor1[i,
    ] == max(mtx.cor1[i, ]))]), "anticorrelates with", colnames(mtx.cor1)[which(m
    ] == min(mtx.cor1[i, ]))], "at corr. coeff.", formatC(mtx.cor1[i,
    which(mtx.cor1[i, ] == min(mtx.cor1[i, ]))]), sep = "|"),
  paste(dname, "maxmin_correlations.txt", sep = ""), append = T,
  sep = "\t", col.names = F, row.names = F)
```

Enrichment results visualization

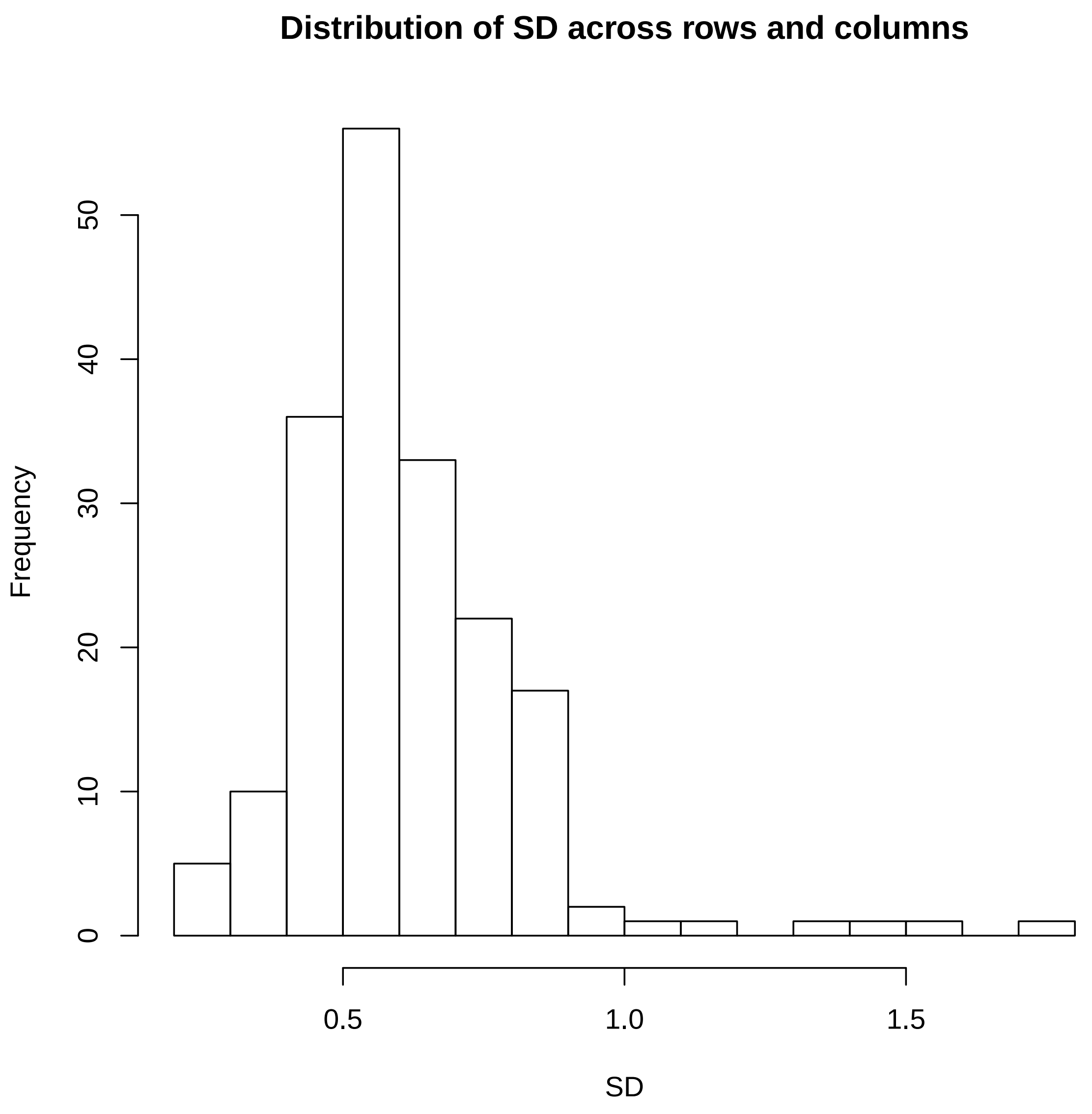
Clstering and visualization of the enrichment results is the main heatmap generated by GenomeRunner. It is the fastest way to overview which epigenomic elements enriched where, and how strong. Due to large number of epigenomic elements typically used for the analysis, GenomeRunner filters those that do not show enrichment in any of the sets of SNPs. We will tweak the default filtering parameters to visualize the most significant enrichments.

First, we define the minimum number of times an epigenomic element should show statistically significant associations - at least once in this tutorial. Then, we investigate and set the transformed p-value and SD cutoffs. The higher the p-value cutoff - the more epigenomic elements will be filtered. The SD cutoff is used to filter out "stably" enriched epigenomic elements and bring up the "most variable" to visualize "differentially enriched" epigenomic elements.

```
# Define minimum number of times a row/col should have values
# above the cutoffs
numofsig <- 1
dim(mtx)  # Original dimensions
# Check summary and set p-value and variability cutoffs as 3rd
# quantiles of their distributions
summary(as.vector(abs(mtx)))
cutoff.p <- summary(as.vector(abs(mtx)))[[5]]
summary(as.vector(apply(abs(mtx), 1, sd)))
cutoff.sd <- summary(as.vector(apply(abs(mtx), 1, sd)))[[5]]
# Check visual distributions and set p-value and variability
# cutoffs manually
hist(as.vector(mtx), breaks = 20, main = "Distribution of -log10-transformed p-values
    xlab = "-log10-transformed p-values")
hist(c(as.vector(apply(mtx, 1, sd)), as.vector(apply(mtx, 2, sd))),
    breaks = 20, main = "Distribution of SD across rows and columns",
    xlab = "SD")
cutoff.p <- 4
cutoff.sd <- 0.5
```



Distribution of −log10−transformed p−values
Frequency
0
1000
2000
3000
4000
5000
0
2
4
6
−log10−transformed p−values



We trim both rows and columns.

```
mtx.gf <- mtx
# Remove rows/cols that do not show significant p-values and
# variability less than numofsig times
mtx.gf <- mtx.gf[apply(mtx.gf, 1, function(row) {
    sum(abs(row) > cutoff.p) >= numofsig
}), apply(mtx.gf, 2, function(col) {
    sum(abs(col) > cutoff) >= numofsig
})]
mtx.gf <- mtx.gf[apply(mtx.gf, 1, sd) > cutoff.sd, apply(mtx.gf, 2,
    sd) > cutoff.sd]
dim(mtx.gf)  # Dimensions after trimming
```

We define clustering parameters, and symmetrical color breaks. The symmetry of color gradient helps to visually judge absolute significance. Note that the range between positive and negative cutoffs is gray, to indicate less significant enrichments. However, our p-value cutoff is more stringent `(cutoff.p<-4)` than commonly used 0.01 (2 after -log10-transformation). Therefore it helps to set the "gray" range from -2 to 2. If we are interested in `relative` comparison of the enrichments, we can simply use the whole range of the transformed p-values without defining breaks, as shown below.

```r
# Adjust clustering parameters.  Distance: 'euclidean',
# 'maximum','manhattan', 'canberra', 'binary' or 'minkowski'.
# Clustering: 'ward', 'single', 'complete', 'average', 'mcquitty',
# 'median' or 'centroid'
dist.method <- "maximum"
hclust.method <- "ward"
granularity <- 7
my.breaks <- c(seq(min(mtx.gf), max(mtx.gf)))
color <- colorRampPalette(c("blue", "yellow"))
h <- heatmap.2(as.matrix(mtx.gf), distfun = function(x) {
    dist(x, method = dist.method)
}, hclustfun = function(x) {
    hclust(x, method = hclust.method)
}, dendrogram = "none", breaks = my.breaks, col = color, lwid = c(1.5,
    3), lhei = c(1.5, 4), key = T, symkey = T, keysize = 0.01, density.info = "none",
    trace = "none", cexCol = 1, cexRow = 0.8)
```

This tutorial has been denerated on February 2, 2014with R version R version 3.0.2 (2013-09-25) on a x86$_6 4 - w64 - mingw32platform$.

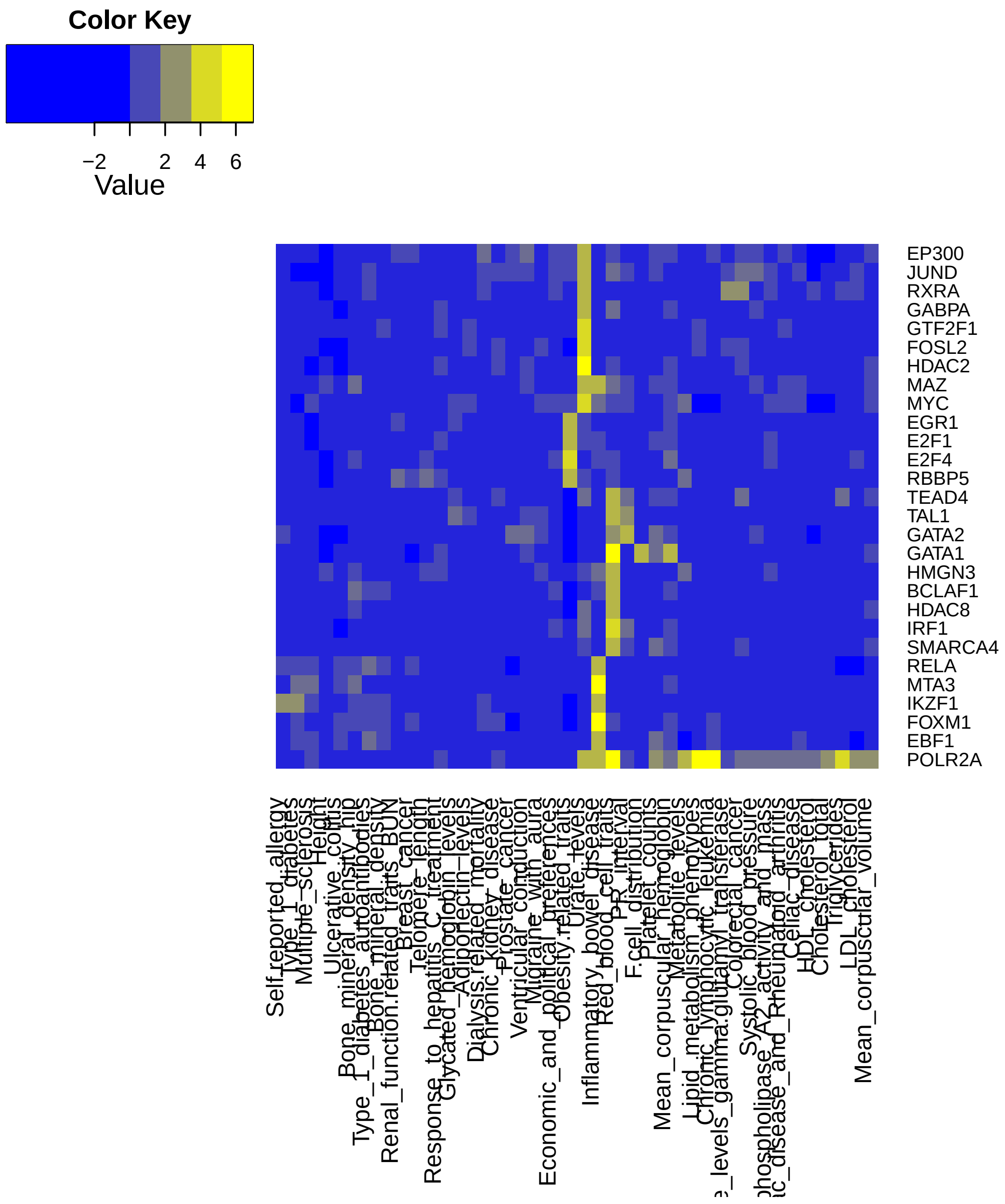


Figure 2: Heatmap of the enrichment results