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

Mikhail Dozmorov

February 7, 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, more visualization options will be explored;
- 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 https://github.com/mdozmorov/gwas2bed. We will use the results from the `/data/more15_vs_tfbsEncode` folder.

Load the libraries

```
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 depletion.

```
# Define output and data subfolders to use, change to analyze
# different data
rname <- "data//results//"  # Output folder
# One or more GenomeRunner Web results data folders.
dname <- "data//more15_vs_tfbsEncode//"
mtx <- do.call("rbind", lapply(dname, function(fn) as.matrix(read.table(paste(fn,
    "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 the case of depletion. Rows are the TFBSs names, columns are the names of disease- or trait-associated SNP sets. We remove the row if a corresponding TFBS does not show enrichment in any of the SNP sets. If a SNP set shows no enrichments in any of the TFBS, we remove the corresponding column as well. Check the final dimensions before actually trim the matrix.

```
dim(mtx)  # Check original dimensions

## [1] 161 243

# Define minimum number of times a row/col should have values
# above the cutoffs
numofsig <- 1
cutoff <- -log10(0.01)  # p-value significance cutoff
# What remains if we remove rows/cols with nothing significant
dim(mtx[apply(mtx, 1, function(x) sum(abs(x) > cutoff)) >= numofsig,
    apply(mtx, 2, function(x) sum(abs(x) > cutoff)) >= numofsig])

## [1] 118  65

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

Visualizing epigenomic similarity results

Epigenomic similarity analysis groups SNP sets by correlation 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 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")
```

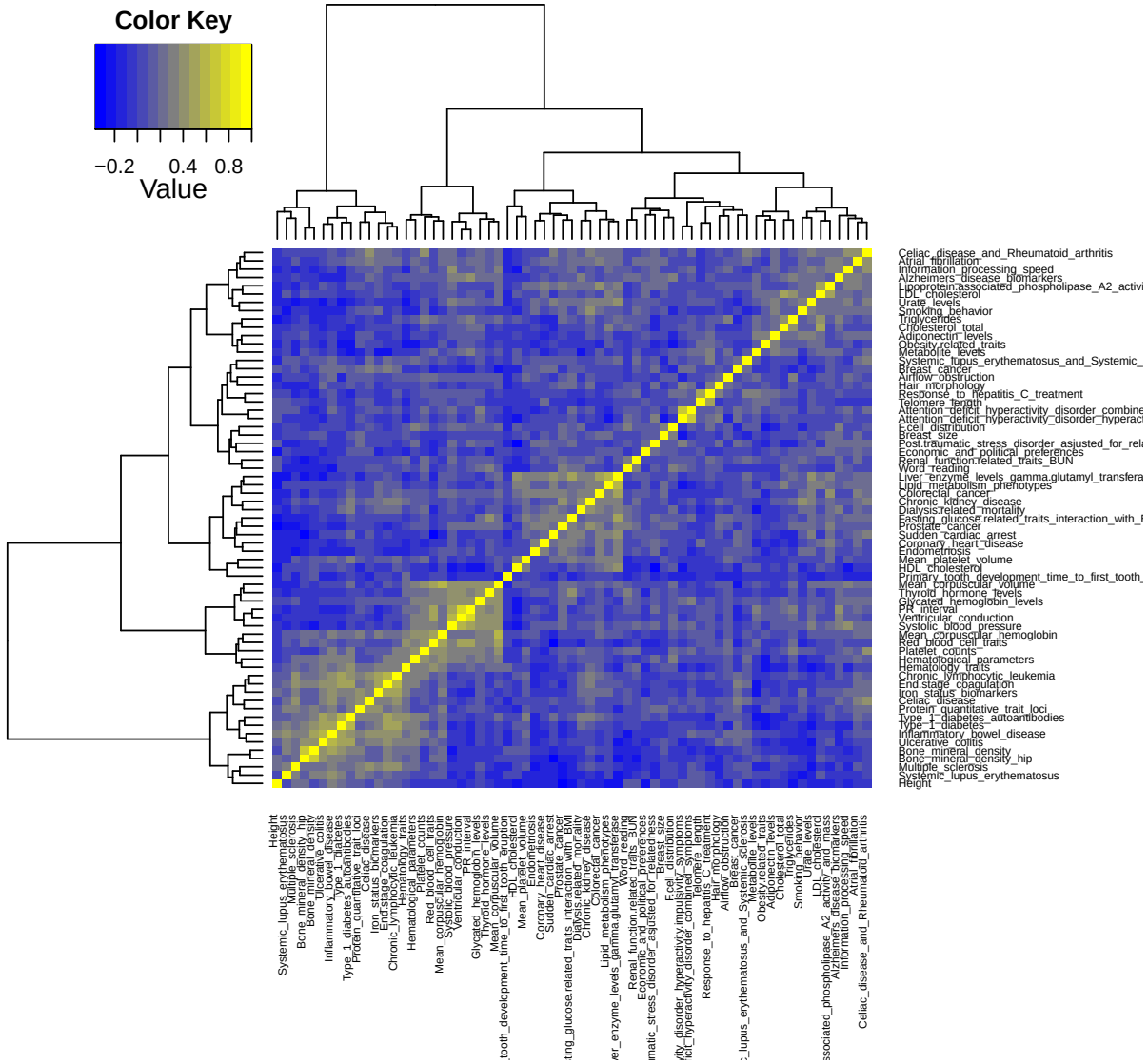


Figure 1: Epigenomic similarity heatmap

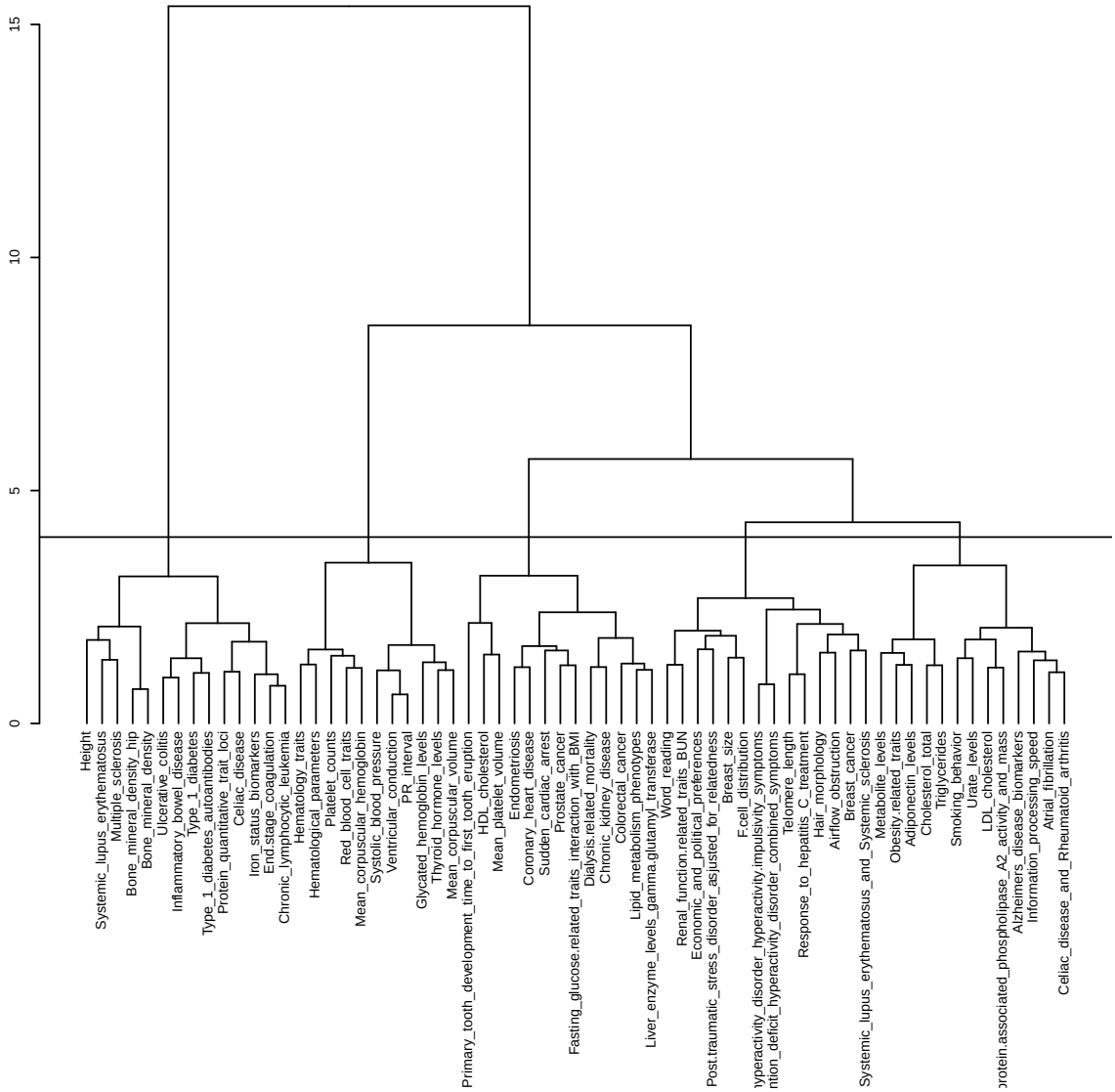
We visualzie the matrix of correlation coefficients as a clustered heatmap (Figure 1). Tweak distance and clustering parameters, or use a code snippet from `01_heatmap_corr.R` script for automated testing of combinations of them.

```
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)
```

Epigenomic differences among the clusters

The heatmap object contains information about the clustering. We visualize it as a dendrogram, cut into separate clusters defined by the cut height (defined empirically), and output the cluster ordering in a file. We also check the number of members in each cluster, and set the minimum number of members for a cluster to be considered for differential enrichment.

```
plot(h$colDendrogram, ylim = c(0, 15))
# Cut the dentrogram into separate clusters. Tweak the height
abline(h = 4)  # Visually evaluate the height where to cut
```



```
c <- cut(h$colDendrogram, h = 4)
# Check the number of clusters, and the number of members.
for (i in 1:length(c$lower)) {
    cat(paste("Cluster", formatC(i, width = 2, flag = "0"), sep = ""),
        "has", formatC(attr(c$lower[[i]], "members"), width = 3),
        "members", "\n")
}

## Cluster01 has  14 members
## Cluster02 has  11 members
## Cluster03 has  13 members
## Cluster04 has  14 members
## Cluster05 has  13 members

# Output the results into a file
unlink(paste(rname, "clustering.txt", sep = ""))
for (i in 1:length(c$lower)) {
    write.table(paste(i, t(labels(c$lower[[i]])), sep = "\t"), paste(rname,
        "clustering.txt", sep = ""), sep = "\t", col.names = F, row.names = F,
        append = T)
}
```

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 a minimum number of elements are not considered.

```
eset.labels <- character()  # Empty vector to hold cluster labels
eset.groups <- numeric()  # Empty vector to hold cluster groups
# Set the minimum number of members to be considered for the
# differential analysis
minmembers <- 9
for (i in 1:length(c$lower)) {
    # Go through each cluster If the number of members is more than a
    # minimum number of members
    if (attr(c$lower[[i]], "members") > minmembers) {
        eset.labels <- append(eset.labels, labels(c$lower[[i]]))
        eset.groups <- append(eset.groups, rep(i, length(labels(c$lower[[i]]))))
    }
}
```

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

The results of *limma* analysis are summarized in the `degs.txt` file.

```
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))
colnames(degs.matrix) <- paste("c", seq(1, length(c$lower)), sep = "")
rownames(degs.matrix) <- paste("c", seq(1, length(c$lower)), sep = "")
# Tweak p-value and log2 fold change cutoffs
cutoff.pval <- 0.05
cutoff.lfc <- log2(2)
unlink(paste(rname, "degs.txt", sep = ""))
for (i in colnames(design)) {
    for (j in colnames(design)) {
        # Test only unique pairs of clusters
        if (as.numeric(sub("c", "", i)) < as.numeric(sub("c", "",
            j))) {
            # 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)
            if (dim(degs)[[1]] > 0) {
                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]]
                # 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, keeping columns with average
                # transformed p-value being significant in at least one condition
                write.table(cbind(degs, i.vs.j, i.av, j.av)[abs(i.av) >
                  -log10(cutoff.pval) || abs(j.av) > -log10(cutoff.pval),
                  ], paste(rname, "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.

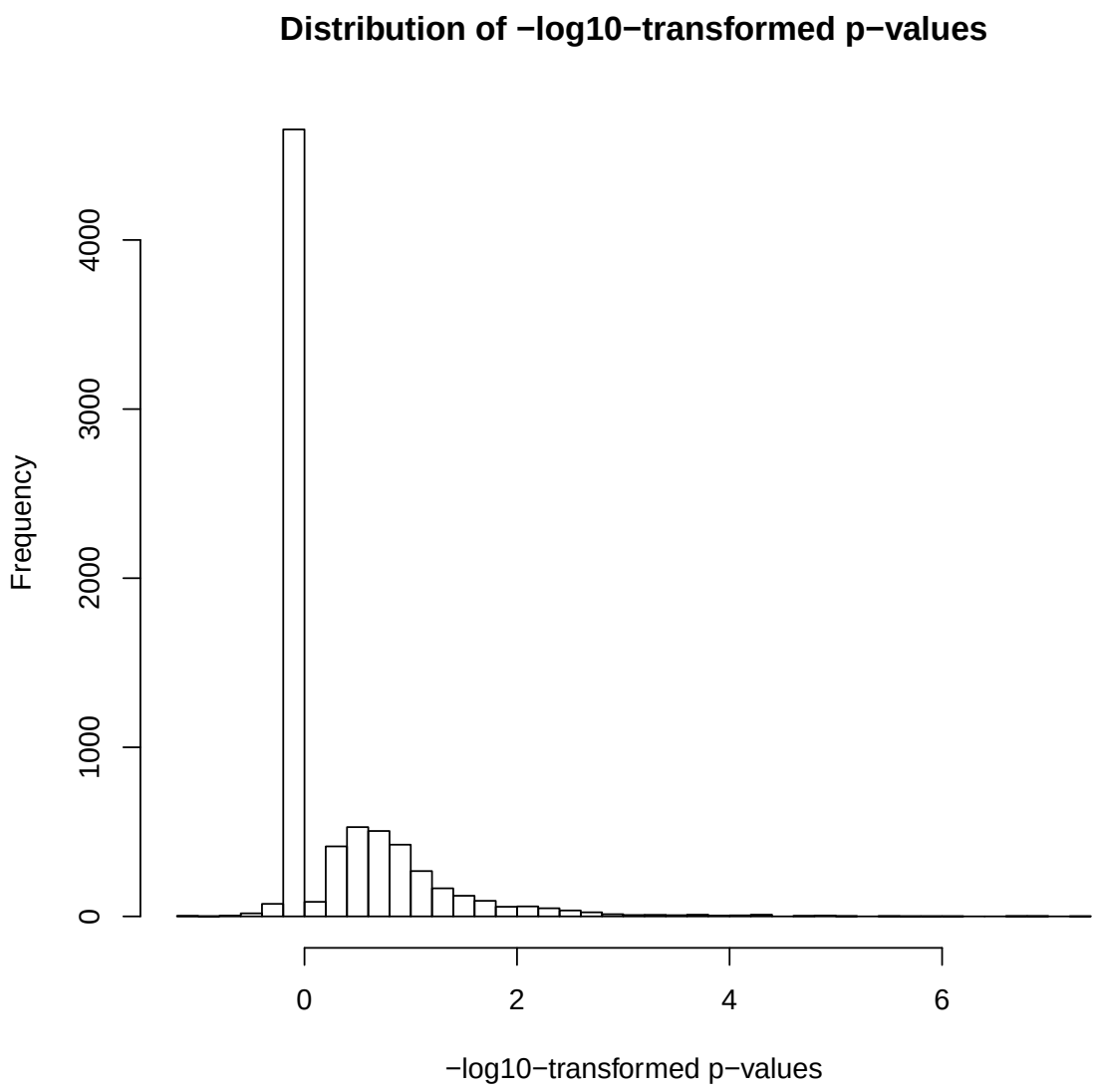
```
mtx.cor1 <- mtx.cor[[1]]
# We don't need to consider perfect correlations, zero them out
diag(mtx.cor1) <- 0
# Print top correlated parameters on screen
for (i in head(unique(mtx.cor1[order(mtx.cor1, decreasing = T)]))) {
    print(which(mtx.cor1 == i, arr.ind = T))
}
for (i in 1:ncol(mtx.cor1)) 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(rname, "maxmin_correlations.csv", sep = ""), append = T,
    sep = ",", col.names = F, row.names = F)
```

Enrichment results visualization

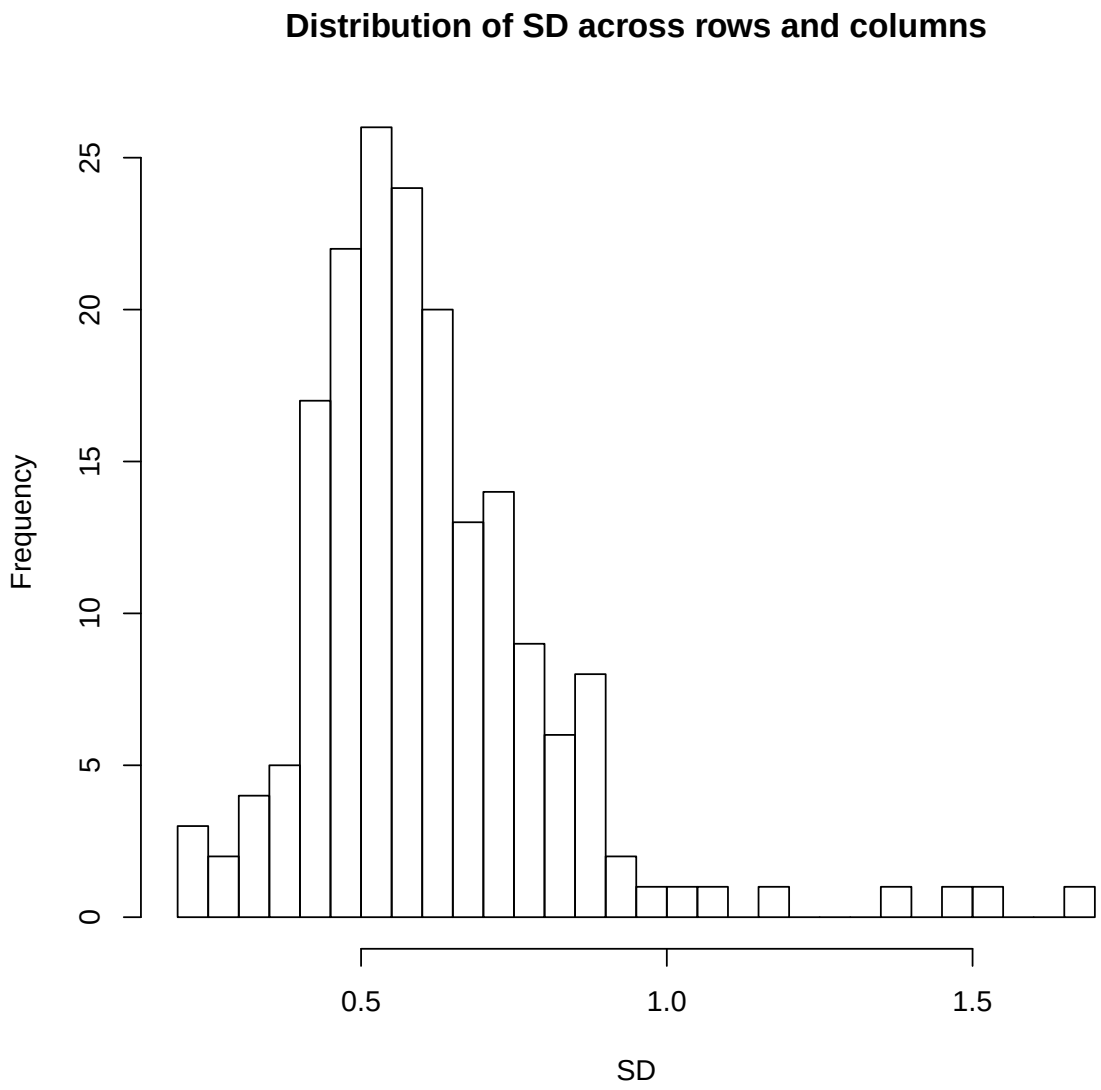
Clstering and visualization of the enrichment results is the main heatmap generated by GenomeRunner. It is the fastest way to overview which epigenomic elements enriched where, and how strong. Due to large number of epigenomic elements typically used for the analysis, GenomeRunner filters those that 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 similarly enriched epigenomic elements and bring up the most differentially enriched epigenomic elements.

```r
# 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 = 50, 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 = 50, main = "Distribution of SD across rows and columns",
    xlab = "SD")
```




```
cutoff.p <- -log10(0.05)
cutoff.sd <- 0.8
```

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.p) >= 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 visualize weak/strong `relative` enrichments using blue/yellow gradient (Figure 2). The `absolute` enrichment results visualization may be achieved by manually setting color breaks.

```
dist.method <- "maximum"
hclust.method <- "ward"
my.breaks <- c(seq(min(mtx.gf), max(mtx.gf)))
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 7, 2014
R version 3.0.2 (2013-09-25) on a $x86_64 - w64 - mingw32 platform$.

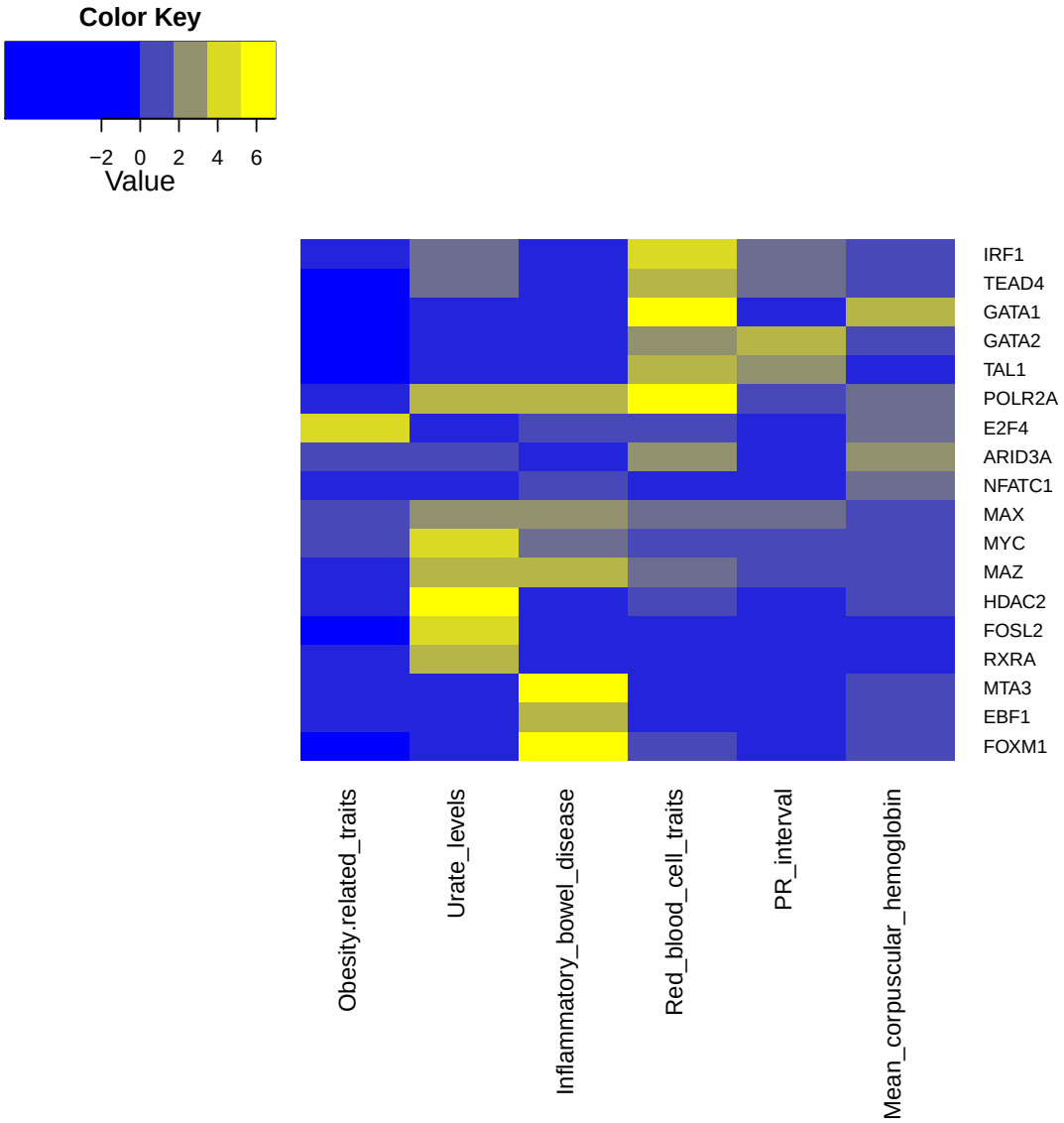


Figure 2: Heatmap of the enrichment results