Exploratory analysis and explanation of the GenomeRunner results

Mikhail Dozmorov

05 January, 2015

Prerequisites

- A matrix of the enrichment results. Rows are the names of regulatory datasets (aka Genomic Features, GFs), columns are the names of features of interest sets (FOIs), cells are the enrichment p-values.
- R, RStudio (http://www.rstudio.com/) recommended
- Several R packages, listed in the *utils.R* file
- Results of the enrichmend analysis of disease- and trait-specific sets of SNPs from GWAScatalog (FOIs) in 161 transcription factor binding sites (TFBSs, GFs), using GenomeRunner web (http://genomerunner.org) server;

What will be done

- The results of the analysis will be loaded into R environment, and pre-processed;
- Regulatory similarity among the FOIs will be visualized;
- Most regulatory similar sets of SNPs, correlated by their regulatory enrichment profiles, will be identified;
- The clusters of FOIs will be defined;
- Regulatory datasets differentially enriched among the clusters of FOIs will be identified;
- The enrichment analysis results ‘as-is’ will be visualized.

Data analysis and preparation

The results used in this tutorial have been obtained with GenomeRunner Web (http://genomerunner.org/). The BED files for this tutorial, as well as other sets of FOIs, are available on (https://github.com/mdozmorov/gwas2bed)https://github.com/mdozmorov/gwas2bed (https://github.com/mdozmorov/gwas2bed). We will use the results from the */data/more15 vs tfbsEncode* folder.

Prepare the environment

```
suppressMessages(source("utils.R"))  # See the required packages there
# Set up the file names to output the results
fn_maxmin <- "results/maxmin_correlations_all.txt"  # Pair-wise regulatory similar
ity results
fn_clust <- "results/clustering_all.txt"  # Regulatory similar clusters
fn_degs <- "results/degs_all.txt"  # Differential GFs among the clusters
```

The enrichment analysis results are outputted in the *matrix.txt* file, stored as the enrichment p-values with a ”-” sign added to denote depletion. The raw p-values are transformed using -log10 transformation, the ”-” sign is kept for depleted associations.

```
# Define output and data subfolders to use, change to analyze different data
rname <- "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))))  # Combine mult
iple results matrixes
# Optional: filter unused genomic features mtx<-mtx[grep('snp', rownames(mtx), ign
ore.case=T, invert=T), ]
mtx <- mtx.transform(mtx)  # -log10 transform p-values, keeping the '-' sign
# Optional: adjust columns for multiple testing. See utils.R for the function defi
nition.  mtx<-mtx.adjust(mtx)
```

The matrix now contains the transformed p-values, negative in the case of depletion. Rows are the TFBS names, columns are the names of disease- or trait-associated SNP sets. The matrix is filtered by removing the rows 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, the corresponding columns are removed as well. It is a good idea to check the final dimensions before actually trim the matrix.

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

```
[1] 161 224
```

```
# Define minimum number of times a row/col should have values # above the cutoffs
numofsig <- 1
cutoff <- -log10(0.1)  # 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] 98 18
```

```
# 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]
```

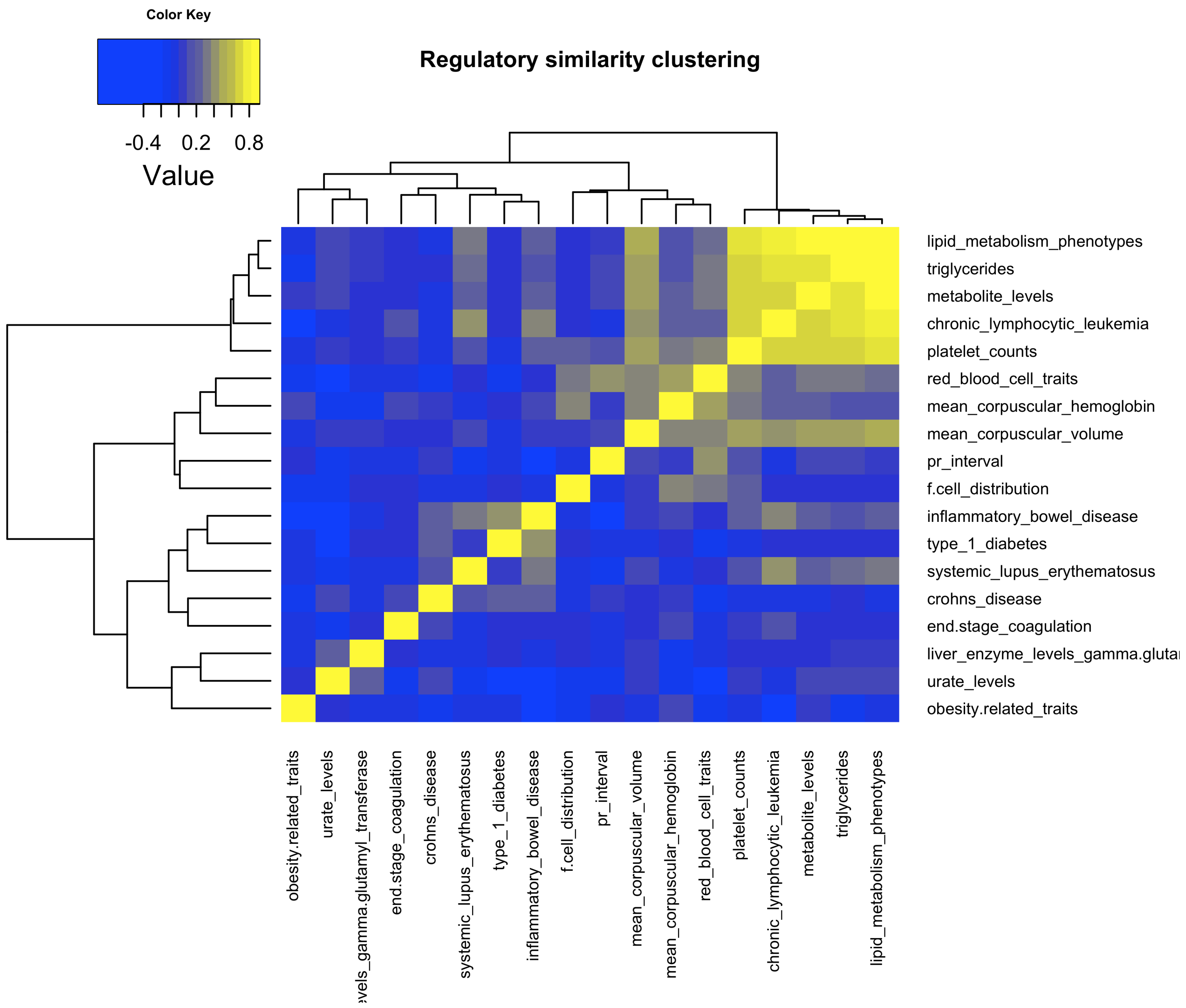
Visualizing regulatory similarity results

Regulatory similarity analysis groups SNP sets by correlation of their regulatory enrichemt profiles, or sets of the transformed p-values. The NxN square matrix of correlation coefficients is assembled, where N is the count of SNP sets.

```
# rcorr returns a list, [[1]] - correl coeffs, [[3]] - p-values. Type - pearson/sp
earman
mtx.cor <- rcorr(as.matrix(mtx), type = "pearson")
# Optionally, try kendall correlation mtx.cor[[1]]<-cor(as.matrix(mtx), method='ke
ndall')
```

The matrix of correlation coefficients is visualized as a clustered heatmap. The distance and clustering parameters can be tweaked, or a code snippet from 01_heatmap_corr.R (https://github.com/mdozmorov/R.genomerunner/blob/master/01_heatmap_corr.R) script can be used for automated testing of the combinations of them.

```
par(oma = c(5, 0, 0, 5), mar = c(10, 4.1, 4.1, 5), cex.main = 0.65)  # Adjust marg
ins
color <- colorRampPalette(c("blue", "yellow"))  # Define color gradient
# color <- greenred # Standard green-black-red palette Adjust clustering parameter
s.  Distance: 'euclidean', 'maximum','manhattan' or 'minkowski'. Do not use 'canbe
rra' or 'binary' Clustering: 'ward', 'single', 'complete', 'average', 'mcquitty',
'median' or 'centroid'
dist.method <- "euclidean"
hclust.method <- "ward.D2"
# Setting breaks to go from minimum to maximum correlation coefficients, excluding
 min/max outliers. This way we get rid of diagonale of 1's
granularity <- 10
my.breaks <- seq(min(mtx.cor[[1]][mtx.cor[[1]] != min(mtx.cor[[1]])]), max(mtx.cor
[[1]][mtx.cor[[1]] != max(mtx.cor[[1]])]), length.out = (2 * granularity + 1))
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 = 0.7, cexCol = 0.7, breaks = my.breaks, main = "Regulatory similarity c
lustering")
```



Most regulatory similar sets of SNPs

Out of all regulatory similarity results, the natural question may be asked: ”What pair of sets of SNPs show the strongest regulatory similarity?”

To ansver it, the correlation matrix is scanned to detect rows corresponding to minimum and maximum correlation coefficients. The perfect self-correlations should be ignored by setting the diagonal of the matrix containing such self-correlations to 0. The results are outputted in the results/maxmin_correlations_all.txt file.

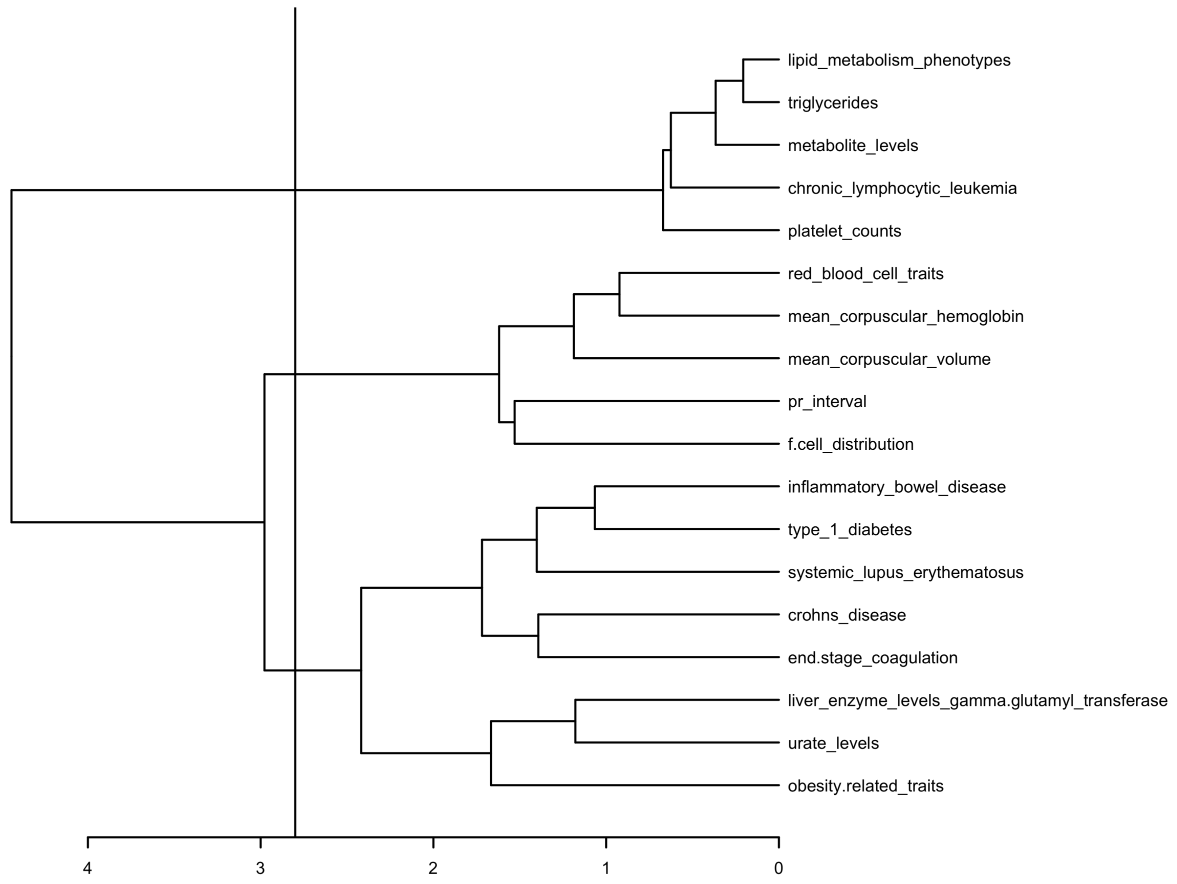
```
# Checking max/min correlations
mtx.cor1 <- mtx.cor[[1]]
diag(mtx.cor1) <- 0  # We don't need to consider self correlations, zero them out
mtx.cor1[lower.tri(mtx.cor1)] <- 0  # Also zero out one matrix triangle, to avoid
duplicate pairs
mtx.maxMin <- melt(mtx.cor1)  # Convert the matrix into tidy data
mtx.maxMin <- mtx.maxMin[order(mtx.maxMin$value, decreasing = T), ]  # Reorder the
 data by maxMin correlation
mtx.maxMin <- mtx.maxMin[mtx.maxMin$value != 0, ]  # Remove 0-correlated pairs
row.names(mtx.maxMin) <- NULL
colnames(mtx.maxMin) <- c("Disease 1", "Disease 2", "Corr. coefficient")
pander(head(mtx.maxMin, n = 10))
```

Disease 1	Disease 2	Corr. coefficient
triglycerides	lipid_metabolism_phenotypes	0.9151
lipid_metabolism_phenotypes	metabolite_levels	0.8686
lipid_metabolism_phenotypes	chronic_lymphocytic_leukemia	0.8501
triglycerides	metabolite_levels	0.8007
triglycerides	chronic_lymphocytic_leukemia	0.7778
lipid_metabolism_phenotypes	platelet_counts	0.7554
platelet_counts	chronic_lymphocytic_leukemia	0.7215
triglycerides	platelet_counts	0.7207
metabolite_levels	chronic_lymphocytic_leukemia	0.7157
metabolite_levels	platelet_counts	0.6966

```
write.table(mtx.maxMin, fn_maxmin, sep = "\t", quote = F, row.names = F)
```

Regulatory differences among the clusters (Differential enrichment analysis)

The heatmap object contains information about the clustering. This information can be visualized as a dendrogram, cut into separate clusters defined by the cut height (user defined). The cluster ordering is outputted into the results/clustering_all.txt file.


lipid_metabolism_phenotypes
triglycerides
metabolite_levels
chronic_lymphocytic_leukemia
platelet_counts
red_blood_cell_traits
mean_corpuscular_hemoglobin
mean_corpuscular_volume
pr_interval
f.cell_distribution
inflammatory_bowel_disease
type_1_diabetes
systemic_lupus_erythematosus
crohns_disease
end.stage_coagulation
liver_enzyme_levels_gamma.glutamyl_transferase
urate_levels
obesity.related_traits
4
3
2
1
0

```
Cluster01 has   8 members
obesity.related_traits
urate_levels
liver_enzyme_levels_gamma.glutamyl_transferase
end.stage_coagulation
crohns_disease
systemic_lupus_erythematosus
type_1_diabetes
inflammatory_bowel_disease

Cluster02 has   5 members
f.cell_distribution
pr_interval
mean_corpuscular_volume
mean_corpuscular_hemoglobin
red_blood_cell_traits

Cluster03 has   5 members
platelet_counts
chronic_lymphocytic_leukemia
metabolite_levels
triglycerides
lipid_metabolism_phenotypes
```

The p-values are -log10-transformed. The distributions of these transformed p-values can be tested for the statistically significant differences among the clusters using the Bioconductor *limma* package.

First, the cluster groups and labels are defined. If the number of SNP sets in each cluster is less than the minimum number (user defined), such cluster is not considered for the differential enrichment analysis.

For the *limma* analysis, the ExpressionSet, the design matrix, and the thresholds are defined. The limma analysis is performed for each pair of clusters. The results of limma analysis are outputted into the results/degs_all.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 an empty square matrix to hold 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, and adjustment for multiple testing
cutoff.pval <- 0.05
cutoff.lfc <- log2(1)
cutoff.adjust <- "none"  # Use 'fdr' or 'none'
unlink(fn_degs)
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 = "-"), l
evels = design)
            fit <- lmFit(eset, design)
            fit2 <- contrasts.fit(fit, contrast.matrix)
            fit2 <- eBayes(fit2)
            degs <- topTable(fit2, number = dim(exprs(eset))[[1]], adjust.method =
 cutoff.adjust, p.value = cutoff.pval, lfc = cutoff.lfc)
            if (nrow(degs) > 0) {
                # Average values in clusters i and j
                i.av <- rowMeans(matrix(exprs(eset)[rownames(degs), eset.groups ==
 as.numeric(sub("c", "", i))], nrow = nrow(degs)))
                j.av <- rowMeans(matrix(exprs(eset)[rownames(degs), eset.groups ==
 as.numeric(sub("c", "", j))], nrow = nrow(degs)))
                # Merge and convert the values
                degs.pvals.log <- cbind(i.av, j.av)
                degs.pvals <- matrix(0, nrow = nrow(degs.pvals.log), ncol = ncol(d
egs.pvals.log), dimnames = list(rownames(degs.pvals.log), c(i, j)))  # Empty matri
x to hold converted p
                for (ii in 1:nrow(degs.pvals.log)) {
                  for (jj in 1:ncol(degs.pvals.log)) {
                    if (degs.pvals.log[ii, jj] < 0) {
                      sign <- -1
                    } else {
                      sign <- 1
                    }
                    degs.pvals[ii, jj] <- sign/10^abs(degs.pvals.log[ii, jj])
                  }
                }
                degs <- cbind(degs, degs.pvals)  # Bind the differences p-values w
ith the converted averaged association p-values
                degs <- degs[degs$adj.P.Val < cutoff.pval & (abs(degs[, 7]) < 0.01
 | abs(degs[, 8]) < 0.01), ]  # Filter non-significant differences. Warning: Hardc
oded thresholds
                if (dim(degs)[[1]] > 0) {
                  ndegs <- nrow(degs)  # The number of differentially associated r
egulatory datasets
                  degs <- degs[order(degs$adj.P.Val, decreasing = F), ]  # Order t
hem by the ratio of the differences
                  print(paste(i, "vs.", j, ", number of degs significant at adj.p.
val <", cutoff.pval, ":", ndegs))
                  # Keep the number of DEGs in the matrix
                  degs.matrix[as.numeric(sub("c", "", i)), as.numeric(sub("c", "",
 j))] <- ndegs
                  degs.table <- merge(degs, gfAnnot, by.x = "row.names", by.y = "V
1", all.x = TRUE, sort = FALSE)  # Merge with the descriptions
                  if (ndegs > 10) {
                    ndegs <- 10
                  }
                  pandoc.table(degs.table[1:ndegs, c(1, 8, 9, 6, 10)])
                  write.table(degs.table[, c(1, 8, 9, 6, 10)], fn_degs, sep = "\t"
, quote = F, col.names = NA, append = T)
                }
            }
        }
    }
}
```

```
[1] "c1 vs. c3 , number of degs significant at adj.p.val < 0.05 : 1"

---------------------------------------------
 Row.names    c1      c3      adj.P.Val   V2
----------- ------ --------- ----------- ----
  POLR2A    0.1808 9.259e-05  0.003175    NA
---------------------------------------------

[1] "c2 vs. c3 , number of degs significant at adj.p.val < 0.05 : 1"

----------------------------------------------
 Row.names    c2       c3      adj.P.Val   V2
----------- ------- --------- ----------- ----
  POLR2A    0.06224 9.259e-05   0.01744    NA
----------------------------------------------
```

```
print("Counts of regulatory elements differentially associated with each group")
```

```
[1] "Counts of regulatory elements differentially associated with each group"
```

```
pander(degs.matrix)
```

	c1	c2	c3
c1	0	0	1
c2	0	0	1
c3	0	0	0

Enrichment results visualization

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

First, the minimum number of times a regulatory dataset should be statistically significant enriched should be defined - “1” in the following example. Then, the distributions of the transformed p-value and the standard deviations (SDs) are examined, and the corresponding cutoffs for matrix filtering are set - the 3rd quantiles in the example. The higher the p-value cutoff - the more regulatory datasets will be filtered out. The SD cutoff is used to filter out similarly enriched regulatory datasets and bring up *the most differentially enriched* regulatory datasets.

```
par(oma = c(1, 0, 0, 0), mar = c(5.1, 4.1, 4.1, 2.1), cex = 1)
# Define minimum number of times a row/col should have values above the cutoffs
numofsig <- 1
dim(mtx)  # Original dimensions
```

```
[1] 98 18
```

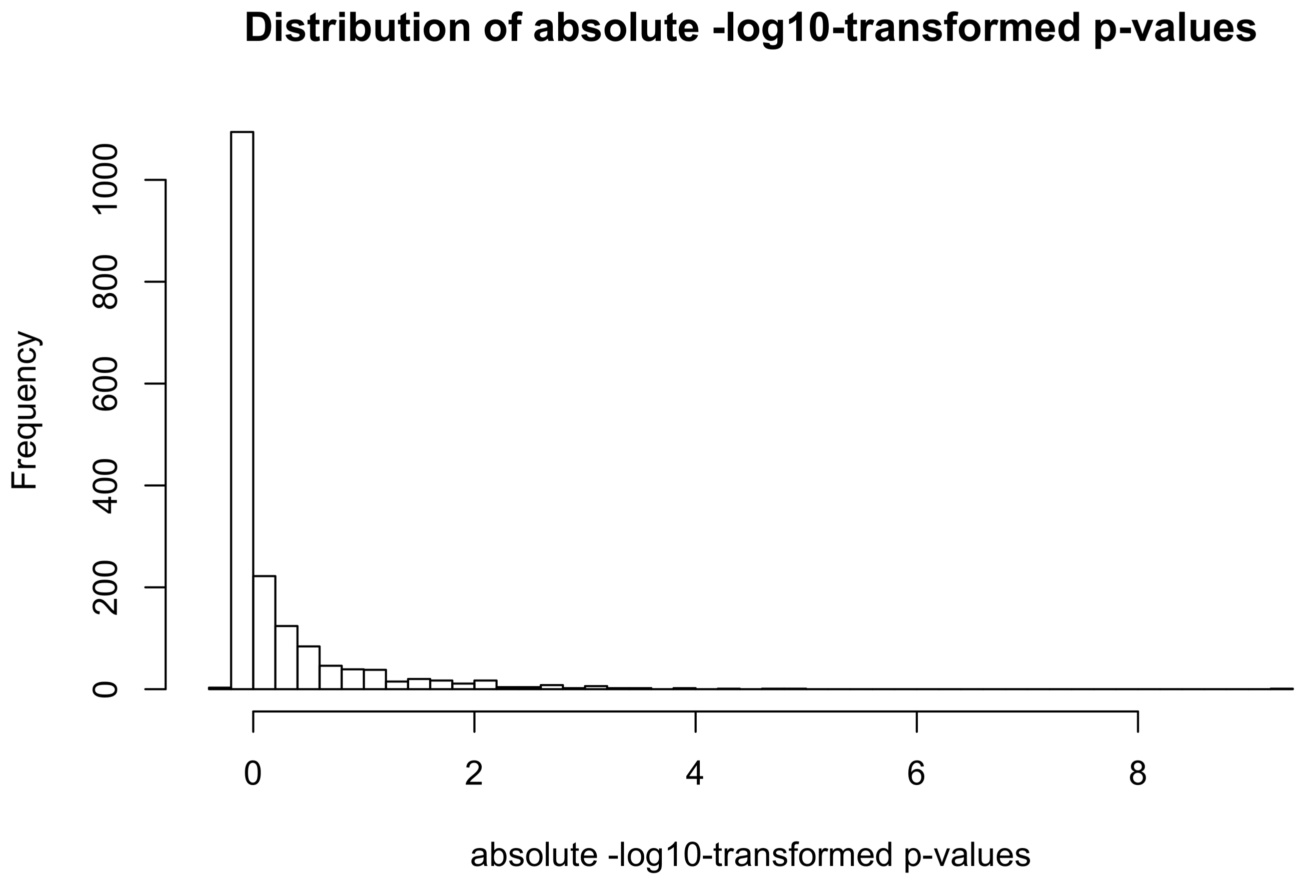
```
# Check summary and set p-value and variability cutoffs as means of their distribu
tions
summary(as.vector(abs(mtx)))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.2557  0.2056  9.3580 
```

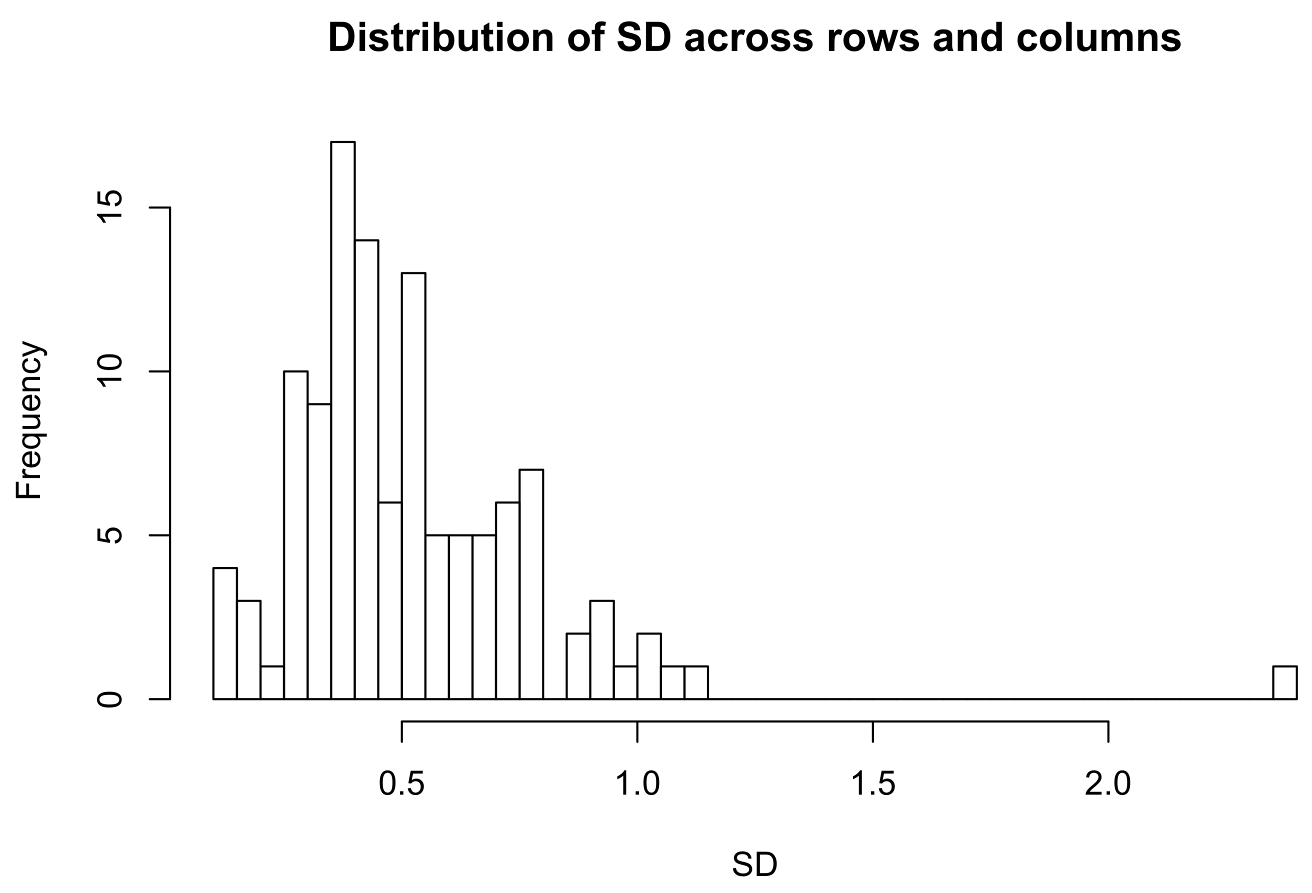
```
cutoff.p <- summary(as.vector(abs(mtx)))[[4]]
summary(as.vector(apply(abs(mtx), 1, sd)))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2530  0.3797  0.4887  0.5386  0.6427  2.3720 
```

```
cutoff.sd <- summary(as.vector(apply(abs(mtx), 1, sd)))[[4]]
# Check visual distributions and set p-value and variability cutoffs manually
hist(as.vector(mtx), breaks = 50, main = "Distribution of absolute -log10-transfor
med p-values", xlab = "absolute -log10-transformed p-values")
```



```
hist(c(as.vector(apply(mtx, 1, sd)), as.vector(apply(mtx, 2, sd))), breaks = 50, m
ain = "Distribution of SD across rows and columns", xlab = "SD")
```



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

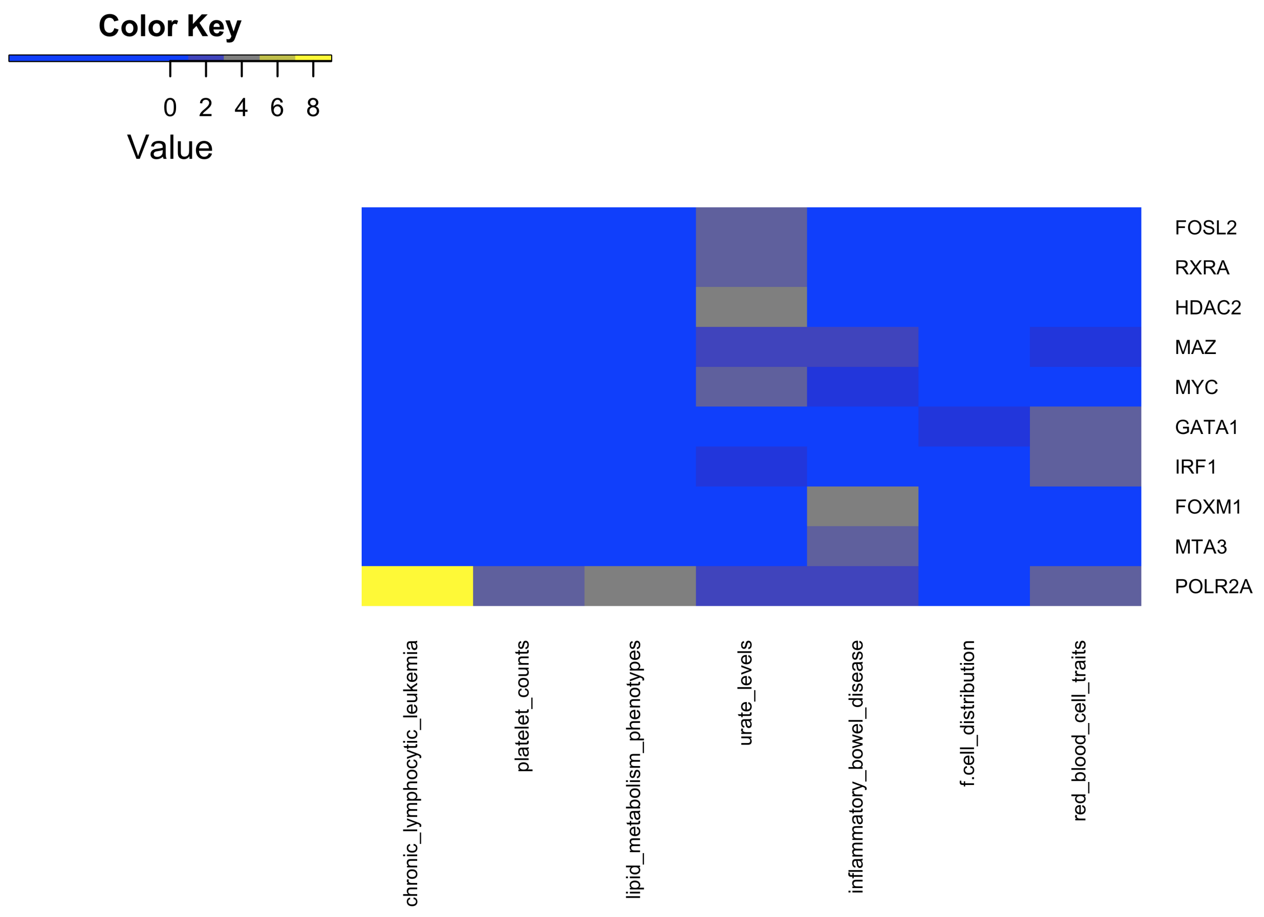
Rows and columns are both filtered. It is possible to either take the top 10 regulatory datasets most differentially enriched across the sets of SNPs, or to filter by the p-value cutoff.

```
# Take top 10 most variably enriched elements
mtx.gf <- mtx[order(apply(mtx, 1, sd), decreasing = T)[1:10], ]
# Or, remove rows/cols that do not show significant p-values less than numofsig ti
mes mtx.gf<-mtx[apply(mtx, 1, function(row){sum(abs(row)>cutoff.p)>=numofsig}), ap
ply(mtx, 2, function(col){sum(abs(col)>cutoff.p)>=numofsig})] Remove sets of SNPs
that do not show variability across the remaining rows
mtx.gf <- mtx.gf[apply(mtx.gf, 1, sd) > cutoff.sd, apply(mtx.gf, 2, sd) > cutoff.s
d]
dim(mtx.gf)  # Dimensions after trimming
```

```
[1] 10  7
```

The weak/strong relative enrichments are visualized using blue/yellow gradient. The absolute enrichment results visualization may be achieved by manually setting color breaks.

```
# Adjust clustering parameters.  Distance: 'euclidean', 'maximum','manhattan', 'ca
nberra', 'binary' or 'minkowski'.  Clustering: 'ward', 'single', 'complete', 'aver
age', 'mcquitty', 'median' or 'centroid'
dist.method <- "maximum"
hclust.method <- "ward.D2"
# granularity=7 my.breaks<-c(seq(min(mtx.gf), -cutoff.p, length.out=granularity),
seq(cutoff.p, max(mtx.gf), length.out=granularity)) my.breaks<-c(seq(min(mtx.gf),
-2, length.out=granularity), seq(2, max(mtx.gf), length.out=granularity))
my.breaks <- c(seq(min(mtx.gf), max(mtx.gf)))
par(oma = c(7, 0, 0, 0), mar = c(5.1, 4.1, 4.1, 2.1), cex = 0.5)
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 = "no
ne", cexCol = 0.7, cexRow = 0.7)
```


Color Key
Value
FOSL2
RXRA
HDAC2
MAZ
MYC
GATA1
IRF1
FOXM1
MTA3
POLR2A
chronic_lymphocytic_leukemia
platelet_counts
lipid_metabolism_phenotypes
urate_levels
inflammatory_bowel_disease
f.cell_distribution
red_blood_cell_traits