Engineering Evaluation by Gene Categorization of Microarray and RNA-seq Data with *eegc* package

***Xiaoyuan Zhou*[1], *Guofeng Meng*[2], *Christine Nardini*[1,3] *and Hongkang Mei*[2]**

[1em] [1] CAS-MPG Partner Institute for Computational Biology, Shanghai Institutes for Biological
Sciences, Chinese Academy of Sciences; University of Chinese Academy of Sciences;
[2] Computational and Modeling Science, PTS China, GSK R&D China, Shanghai;
[3] Personalgenomics, Strada Le Grazie 15 - 37134 Verona, Italy
*Correspondence to zhouxiaoyuan (at) picb.ac.cn

November 1, 2022

Abstract

This package has been developed to evaluate cellular engineering processes for direct differentiation of stem cells or conversion (transdifferentiation) of somatic cells to primary cells based on high throughput gene expression data screened either by DNA microarray or RNA sequencing. The package takes gene expression profiles as inputs from three types of samples: (i) somatic or stem cells to be (trans)differentiated (input of the engineering process), (ii) induced cells to be evaluated (output of the engineering process) and (iii) target primary cells (reference for the output). The package performs differential gene expression analysis for each pair-wise sample comparison to identify and evaluate the transcriptional differences among the 3 types of samples (input, output, reference). The ideal goal is to have induced and primary reference cell showing overlapping profiles, both very different from the original cells.

Using the gene expression profile of original cells versus primary cells, a gene in the induced cells can either be successfully induced to the expression level of primary cells, remain inactive as in the somatic cells, or be insufficiently induced. Based on such differences, we can categorizes differential genes into three intuitive categories: *Inactive*, *Insufficient*, *Successful* and two additional extreme states: *Over* and *Reversed* representing genes being over (way above/below the expected level of in/activation) or reversely regulated. By further functional and

gene regulatory network analyses for each of the five gene categories, the package evaluates the quality of engineered cells and highlights key molecules that represent transcription factors (TFs) whose (in)activation needs to be taken into account for improvement of the cellular engineering protocol, thus offering not only a quantification of the efficacy of the engineering process, but also workable information for its improvement.

***eegc* version:** 1.24.0 [1]

[1] This document used the vignette from *Bioconductor* package *DESeq2* as *knitr* template

Contents

1 Introduction

Cellular engineering is among the most promising and yet questioned cellular biology related approaches, and consists of the man-made differentiation of pluripotent undifferentiated stem cells into tissue-specific (primary) cells, mimicking the processes naturally occurring during human embryonic development, and of the direct conversion from somatic to primary cells, which is relatively efficient and rapid than differentiation but is limited by incomplete conversion. One of the earliest issues to be properly addressed in this area is the possibility to quantify and assess the quality of the induced cells, i.e. to measure if/how

the differentiation process has been successful and the obtained cells are sufficiently similar to the target primary cells, for further applications in regenerative therapy, disease modeling and drug discovery.

The natural approach consists of comparing the transcriptional similarity of engineered cells to the target primary cells, with a focus on the marker genes that are specifically expressed in somatic and primary cells [1]. Indeed, not all marker genes are successfully induced to the expression level typical of primary cells, implying imperfections of the reprogrammed cells to an extent that needs to be quantified and evaluated.

Indeed, the focus on a selected number of transcription fators (TFs) may be limiting, and unable to offer alternative solutions to improve an engineering process. As cells represent a complex and tightly interconnected system, the failure to activate one or more target genes impacts on a variable number of interconnected and downstream genes, affecting in turn a number of biological functions. For this reason the approach we propose is not target-specific but systemic and beyond offering a list of TF whose (in)activation needs attention, can also qualify and quantify the detrimental effects of an imperfect reprogramming on the topology of the gene network and the biological functions affected, thus offering information on the usability of the obtained cells and suggesting a potentially broader number of targets to be affected to improve the process.

Based on these observations, we propose -as briefly introduced above- to classify the genes into five different categories which describe the states of the genes upon (trans)differentiation. The *Successful* category is represented by the genes whose expression in the engineered cells is successfully induced to a level similar to the target primary cells; conversely the *Inactive* category is represented by the genes whose expression are unchanged from the level of initiating cells but should be induced to the expression level of primary cells. Between *Successful* and *Inactive*, genes can be defined *Insufficient* when their expression were modified from the input cells but not enough to reach the level of the target cells. Additionally, because of the induction of transcription factors, some genes can be over expressed in the engineered cells in comparison to the target primary cells, here defined as *Over*; finally, some genes appear to be differentially expressed in a direction opposite to the expected one, these are defined as *Reserved*. By these definitions, a successful engineering cell process is expected to offer a significant number of *Successful* genes, including the marker genes, and a minority of *Inactive* genes. All other three categories contribute to the understanding of the deviations that gave rise to the unexpected outcome of the engineering process. This is namely obtained with 3 outputs offered to the package users, in addition to the identification of inefficient TFs induction: (i) functional enrichment, (ii) tissue specific and (iii) gene regulatory network

analyses. Each of those contribute to a better understanding of the role that each gene category plays in the engineering process and clarify the deficiencies of the engineered cells for potential improvements.

2 Installing the *eegc* package

eegc requires the following CRAN-R packages: *R.utils*, *sna*, *wordcloud*, *igraph* and *pheatmap*, and the *Bioconductor* packages:*limma*, *edgeR*, *DESeq2*, *clusterProfiler*, *org.Hs.eg.db*, *org.Mm.eg.db*.

When *eegc* is installed from *Bioconductor*, all dependencies are installed.

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("eegc")
```

Then package is loaded by:

```
library(eegc)
```

3 Preparing Input Data

eegc takes an input of gene expression data (with genes in rows and samples in columns) screened by microarray or RNA-seq (in FPKM -Fragments Per Kilobase of transcript per Million mapped reads- or counts). Samples belong to three types of samples: original cells (input), induced cells (ouput), and primary cells (output target). In this vignette, we used human RNA-seq expression FPKMs data published by Sandler et Al. as example data. Here, human Dermal Microvascular Endothelial Cells (DMEC) were transduced with transcription factors and cultured in vascular niche to induce the growth of haematopoietic stem and multipotent progenitor cells (rEChMPP) compared with the target primary Cord Blood cells (CB).

```
# load Sandler's data set:
data(SandlerFPKM)

#the column names of the data, representing the samples CB, DMEC, and rEChMPP
colnames(SandlerFPKM)
```

```
## [1] "CB1"      "CB2"      "CB3"      "DMEC1"    "DMEC2"    "rEChMPP1" "rEChMPP2"
```

4 Gene Differential Analysis

The differential analysis is achieved by the `diffGene` function, and two packages *limma* [2] and *DESeq2* [3] are selectively applied to microarray/FPKM data and counts data, respectively. Before the gene differential analysis, the removal of low expressed genes is selectively performed (specifying `TRUE` or `FALSE` in the `filter` parameter) by removal of the genes absent above a given percentage of samples and a log transformation is done on the (filtered) data. The significantly differential genes are identified for each pair-wise sample comparison (DMEC vs rEChMPP, DEMC vs CB, rEChMPP vs CB) and with a given corrected p-value cutoff.

```
# differential expression analysis:
diffgene = diffGene(expr = SandlerFPKM, array=FALSE,  fpkm=TRUE,  counts=FALSE,
                    from.sample="DMEC", to.sample="rEChMPP", target.sample="CB",
                    filter=TRUE, filter.perc =0.4, pvalue = 0.05 )
```

The function gives a list of outputs further used in the following analyses, including the differential result detailed below in `diffgene.result`, the sole differential gene names in `diffgene.genes` and the filtered gene expression values as in `expr.filter`.

```
# differential analysis results
diffgene.result = diffgene[[1]]

# differential genes
diffgene.genes = diffgene[[2]]

#filtered expression data
expr.filter = diffgene[[3]]
dim(expr.filter)
```

```
## [1] 14391     7
```

```
dim(SandlerFPKM)
```

```
## [1] 16692     7
```

5 Gene Categorization

Gene (`g`) categorization is achieved through the pair-wise comparisons (Expression Difference, ED) as defined in eq. 1, a difference of `g` average expression in A and B samples among the three types of samples as shown in Table 1, and the ratio of such differences (`EDg` ratio, eq. 2) .

$$ED_g(A,B) = \overline{E_g} \text{ in A} - \overline{E_g} \text{ in B} \quad (1)$$

$$ED_g \text{ ratio} = \frac{ED_g(rEChMPP, DMEC)}{ED_g(CB, DMEC)} \quad (2)$$

The five gene categories are first defined by the ED patterns observed among three pair-wise comparisons as shown in the ED columns of Table 1. Based on these definitions, categories *Reserved* and *Over* are undistinguishable, but become clearly distinct by evaluation of the ED ratios (eq. 2) in the corresponding column of Table 1.

At this stage, *Inactive* and *Successful* ED ratios are, conveniently, around 0 and 1, however, they cover a relatively wide range of values, with queues overlapping with the *Over* and *Insufficient* categories for *Successful* genes, and with *Reverse* and *Insufficient* for *Inactive* genes. To gain an accurate and practical categorization (operational in term of indications as to which genes need attention in the engineering process), *Inactive* and *Successful* genes boundaries were set more stringently around the intuitive peaks of 0 and 1, by shrinking the ED ratio boundaries to what we name operational ranges in Table 1, which correspond to the 5th and 95th quantile of the ED-ranked *Successful* and *Inactive* genes.

Table 1: Gene categorization base on differential analysis and ED ratio

Category	ED Patterns			ED Ratio	Operational Ranges
	CB, DMEC	rEChMPP, DMEC	rEChMPP, CB		
Reversed	✓/✗	✗	✓	<0	
Over				>1	
Inactive	✓	✗	✓	~ 0	(Q^{5th} ED ratio, Q^{95th} ED ratio)
Insufficient	✓	✓	✓	$0 \sim 1$	
Successful	✓	✓	✗	~ 1	(Q^{5th} ED ratio, Q^{95th} ED ratio)

✓ differential, ✗ nondifferential; Q^{xth} ED ratio: xth quantile of the ranked ED ratios

Function `categorizeGene` performs the categorization and gives a list of outputs with five categories *Reversed*, *Inactive*, *Insufficient*, *Successful* and *Over* with: 1) the gene symbols in each category, 2) corresponding `ED` ratios.

```
# categorizate differential genes from differential analysis
category = categorizeGene(expr = expr.filter,diffGene = diffgene.genes,
                          from.sample="DMEC",
                          to.sample="rEChMPP",
                          target.sample="CB")
cate.gene = category[[1]]
cate.ratio = category[[2]]

# the information of cate.gene
class(cate.gene)

## [1] "list"

length(cate.gene)

## [1] 5

names(cate.gene)

## [1] "Reverse"      "Inactive"     "Insufficient" "Successful"   "Over"

head(cate.gene[[1]])

## [1] "ABCC4"   "ADAM12"  "ADCY6"   "AGPAT2"  "ANKRD37" "ANO9"

head(cate.ratio[[1]])

##         ED_ratio
## ABCC4    -282.44
## ADAM12    -11.57
## ADCY6      -4.79
```

```
## AGPAT2    -23.32
## ANKRD37    -9.24
## ANO9       -0.18
```

6 Gene Expression Pattern Visualization

Expression profile of genes in each category can be visualized as different expression patterns with the markerScatter function that generates a scatter plot of gene expression for the five categories in paired arms and highlight the marker genes in the output (Figure 1). We also apply a linear model to fit the expression profiles of samples on the x- and y-axes, with the possibility to selectively add the regression lines on the figure to show the sample correlation. To avoid confusion and allow a distinct visualization of all expression profiles, *Inactive*, *Insufficient* and *Successful* genes are plotted separately (Figure 1 above) from the *Reserved* and *Over* genes (Figure 1 below), with each of the 5 categories being plotted from left to right to highlight the expression trend change from DMEC to rEChMPP.

```
#load the marker genes of somatic and primary cells
data(markers)

#scatterplot
col = c("#abd9e9", "#2c7bb6", "#fee090", "#d7191c", "#fdae61")
markerScatter(expr = expr.filter, log = TRUE, samples = c("CB", "DMEC"),
              cate.gene = cate.gene[2:4], markers = markers, col = col[2:4],
              xlab = expression('log'[2]*' expression in CB (target)'),
              ylab = expression('log'[2]*' expression in DMEC (input)'),
              main = "")
```

```
markerScatter(expr = expr.filter, log = TRUE, samples = c("CB", "rEChMPP"),
              cate.gene = cate.gene[2:4], markers = markers, col = col[2:4],
              xlab = expression('log'[2]*' expression in CB (target)'),
              ylab = expression('log'[2]*' expression in rEC-hMPP (output)'),
              main = "")
```

```
markerScatter(expr = expr.filter, log = TRUE, samples = c("CB", "DMEC"),
              cate.gene = cate.gene[c(1,5)], markers = markers, col = col[c(1,5)],
              xlab = expression('log'[2]*' expression in CB (target)'),
              ylab = expression('log'[2]*' expression in DMEC (input)'),
              main = "")
```

```
markerScatter(expr = expr.filter, log = TRUE, samples = c("CB", "rEChMPP"),
              cate.gene = cate.gene[c(1,5)], markers = markers, col = col[c(1,5)],
              xlab = expression('log'[2]*' expression in CB (target)'),
              ylab = expression('log'[2]*' expression in rEC-hMPP (output)'),
              main = "")
```

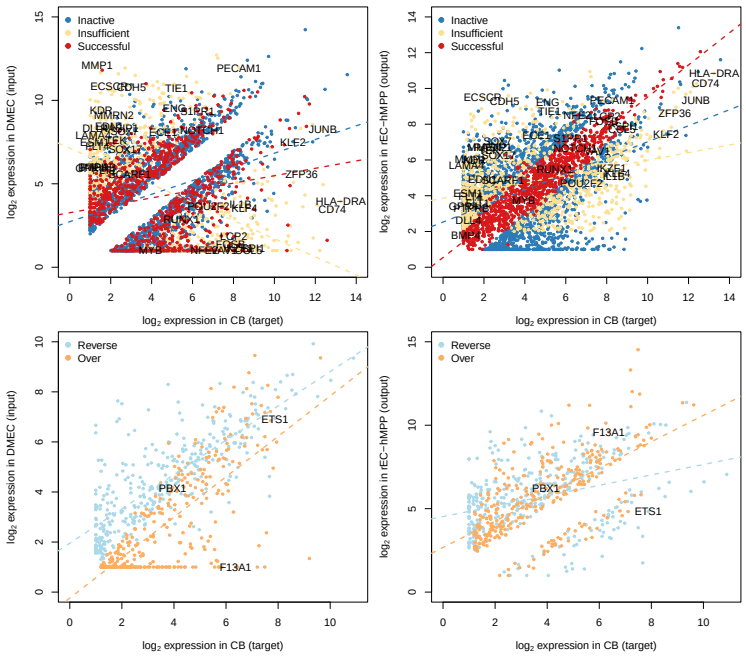

Figure 1: Expression profile (FPKM in log scale) of the five gene categories in CB primary cells against the endothelial cells (left) and rEChMPPs (right), respectively

7 Quantifying Gene Categories

One simple metric to quantify the success of the cellular engineering process is the proportion of genes in each category among all the categorized genes. A high proportion of *Successful* genes reflects a good (trans)differentiation. Thus we produce a density plot to quantify the ED ratios of each gene category. As proposed in Table 1, the ED ratios of *Successful* genes are around 1, *Inactive* genes around 0 and *Insufficient* genes are between 0 and 1. Extreme higher or lower ratios are given by the *Reserved* or *Over* genes, respectively, to make the ratios on x axis readable, we suggest to narrow the ratios of *Reserved* and *Over* genes to a maximum of their median values (Figure 2).

```
# make the extreme ED ratios in Reversed and Over categories to the median values
reverse = cate.ratio[[1]]
```

```
over = cate.ratio[[5]]
reverse[reverse[,1] <= median(reverse[,1]), 1]  = median(reverse[,1])
over[over[,1] >= median(over[,1]),1] = median(over[,1])
cate.ratio[[1]] = reverse
cate.ratio[[5]] = over

# density plot with quantified proportions
densityPlot(cate.ratio, xlab = "ED ratio", ylab = "Density", proportion = TRUE)
```

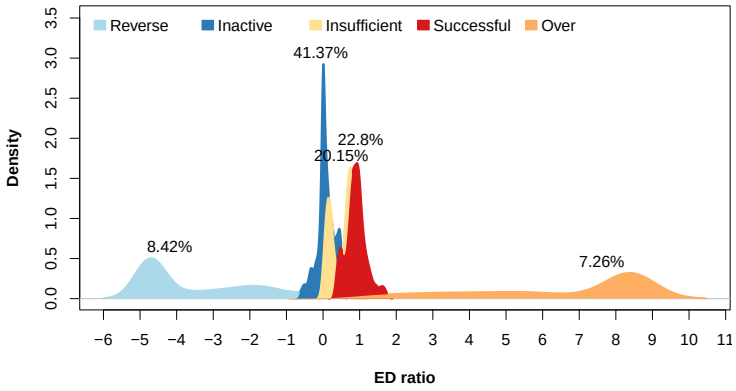

Figure 2: The proportion of genes in each category

8 Functional Enrichment Analysis and Visualization

The functional annotation for each gene category is performed by applying the R package *clusterProfiler* [4]. Given an input genelist with five gene categories and having set the `organism` parameter (optionally, `human` or `mouse`), `functionEnrich` performs the functional enrichment analysis for Gene Ontology (GO) [5] and Kyoto Encyclopedia of genes and Genomes (KEGG) pathway [6] with either hypergeometric test or Gene Set Enrichment Analysis (GSEA).

```
#  result in "enrichResult" class by specifying TRUE to enrichResult parameter
goenrichraw = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                             GO = TRUE, KEGG = FALSE, enrichResult = TRUE)
class(goenrichraw[[1]])
```

```
## [1] "enrichResult"
## attr(,"package")
## [1] "DOSE"
```

```
# result of the summary of "enrichResult" by specifying FALSE to enrichResult parameter
# GO enrichment
goenrich = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                          GO = TRUE, KEGG = FALSE, enrichResult = FALSE)
# KEGG enrichment
keggenrich = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                            GO = FALSE, KEGG = TRUE, enrichResult = FALSE)
```

To describe the significantly enriched functional terms in each category (Figure 3) and for further comparison within the five gene categories (Figure 4), a bar plot function `barplotEnrich` (modified from the `barplot.enrichResult` function in *DOSE* [7] package) and heatmap plot function `heatmapPlot` are added in the package, outputting the most enriched terms selected by parameter `top`.

```
# plot only the "enrichResult" of Inactive category
inactive = goenrichraw[[2]]
barplotEnrich(inactive, top =5, color ="#2c7bb6", title = "Inactive")
```

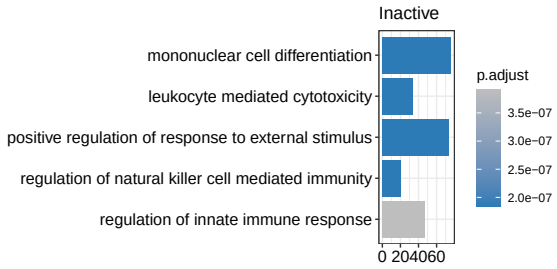

Figure 3: Example of top 5 significantly enriched Gene Ontology terms in the *Inactive* category
The shade of colors represents enrichment p-values and bar length represents the count of genes involved in each GO term.

```
# plot the enrichment results by the five gene categories
data(goenrich)
heatmaptable = heatmapPlot(goenrich, GO = TRUE, top = 5, filter = FALSE,
                           main = "Gene ontology enrichment",
                           display_numbers =  FALSE)
```

This analysis helps to evaluate the engineered cells at the functional level by identifying the non-activated functions that play important roles in primary cells but, being enriched in the *Inactive* genes categories, lack in the engineered cells.

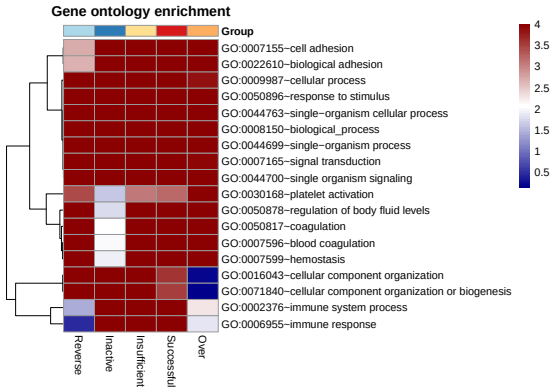

Figure 4: Top 5 significantly enriched Gene Ontology terms in each gene category based on the log transformed corrected p-values
The counts of genes involved in each functional term are displayed on the heatmap by specifying TRUE to the `display_numbers` parameter in heatmapPlot function.

9 Cell/Tissue-specific Enrichment Analysis

The five gene categories represent different reprogramming progresses or directions of the original cells towards primary cells. These progresses or directions can also be explored, in addition to the functional role investigated above, by cell/tissue (C/T)-specific analysis. Downstream of a successful cellular engineering process, the expression of somatic cell-specific genes will be down-regulated while the expression of primary cell-specific genes will be up-regulated. So ideally each of the five gene categories is mainly composed by two gene types: somatic or primary genes. By the C/T-specific analysis, assuming the most extreme values of expression represent effective up or down regulation and compute based on this, we can explore which C/T-specific genes results the success or failure of the engineering process.

The database Gene Enrichment Profiler, containing the expression profiles of 12,000 genes with NCBI GeneID entries across 126 primary human cells/tissues in 30 C/T groups, is used for this C/T-specific analysis [8].

In Gene Enrichment Profiler, genes specificity to a given cell/tissue is ranked using a custom defined enrichment score. For our usage in this package ranking is not sufficient as cell/tissue-genes sets are needed and therefore we applied *SpeCond* [9], a method to detect condition-specific gene, to identify the C/T-specific gene sets . Then we apply the hypergeometric test to assess the specificity significance of gene categories in each tissue with the enrichment function and visualize the enrichment results by the heatmapPlot function (Figure 5).

```
#load the cell/tissue-specific genes
data(tissueGenes)
```

```
length(tissueGenes)

## [1] 126

head(names(tissueGenes))

## [1] "ESCells"                    "HSCFetalBloodCD34CD38"
## [3] "HSCCordBloodCD34CD38"       "HSCCordBloodCD34CD38CD33"
## [5] "HSCBoneMarrowCD34CD38CD33" "HSCFetalBloodCD34CD381"

#load the mapping file of cells/tissues to grouped cells/tissues
data(tissueGroup)
head(tissueGroup)

##                              Tissue              Tissue_abbr      Group
## 1                          ES cells                  ESCells   ES cells
## 2       HSC fetal blood CD34+ CD38-     HSCFetalBloodCD34CD38 stem cells
## 3        HSC cord blood CD34+ CD38-      HSCCordBloodCD34CD38 stem cells
## 4  HSC cord blood CD34+ CD38- CD33-  HSCCordBloodCD34CD38CD33 stem cells
## 5 HSC bone marrow CD34+ CD38- CD33- HSCBoneMarrowCD34CD38CD33 stem cells
## 6       HSC fetal blood CD34+ CD38+    HSCFetalBloodCD34CD381 stem cells

#get the background genes
genes = rownames(expr.filter)
#enrichment analysis for the five gene categories
tissueenrich = enrichment(cate.gene = cate.gene, annotated.gene = tissueGenes,
                          background.gene = genes, padjust.method = "fdr")


#select a group of cells/tissues
tissueGroup.selec = c("stem cells","B cells","T cells","Myeloid","Endothelial CD105+")
tissues.selec = tissueGroup[tissueGroup[,"Group"] %in% tissueGroup.selec,c(2,3)]
tissuetable = heatmapPlot(tissueenrich, terms = tissues.selec, GO=FALSE,
                          annotated_row = TRUE,annotation_legend = TRUE,
                          main = "Tissue-specific enrichment")
```

It is expected that the *Successful* genes are enriched in both somatic and primary cells/tissues but not the *Inactive* or *Insufficient* genes.

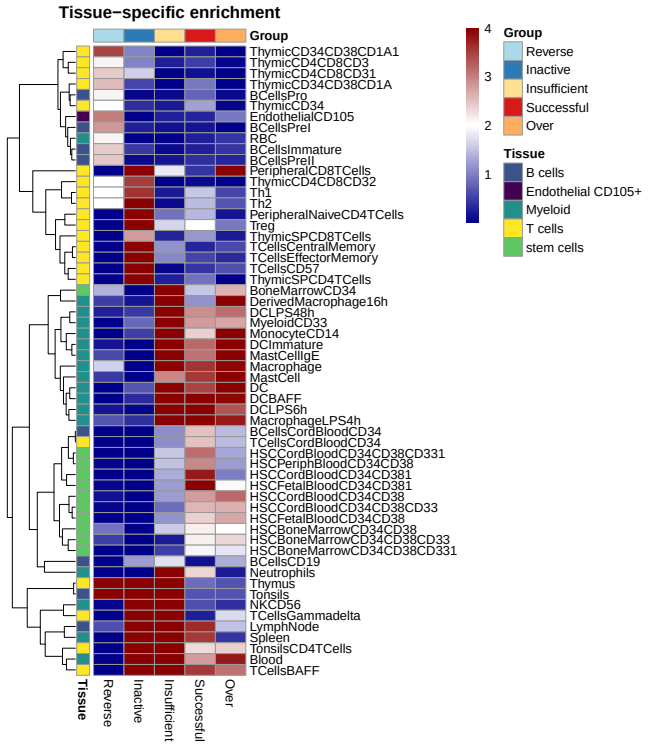

Figure 5: **Cells/tissues-specific enrichment among five gene categories in selected cells/tissues**

10 Evaluation Based on Gene Regulatory Network (GRN) Analysis

The package also build the gene-gene regulation network for each category based on cell/tissue-specific gene regulatory networks (GRNs) constructed by the CellNet [10] team through the analysis of 3419 published gene expression profiles in 16 human and mouse cells/tissues. The complete table of regulatory relationships for all genes (not limited to C/T-specific ones) and for C/T-specific ones are downloaded from the CellNet website.

10.1 CellNet-based Cell/Tissue-specific analysis

Construction is done by checking the percentage of overlapping genes in each category with the genes involved in each C/T-specific GRN by the dotPercentage function (Figure 6) and then performing a C/T-specific enrichment analysis, as in the Cell/Tissue-specific Enrichment Analysis section, based on these gene sets.

```
#load the C/T-specific genes in 16 cells/tissues
data(human.gene)

# the 16 cells/tissues
head(names(human.gene))

## [1] "Esc"        "Ovary"      "Neuron"     "Skin"       "Hspc"
## [6] "Macrophage"

perc = dotPercentage(cate.gene = cate.gene, annotated.gene = human.gene,
              order.by = "Successful")
```

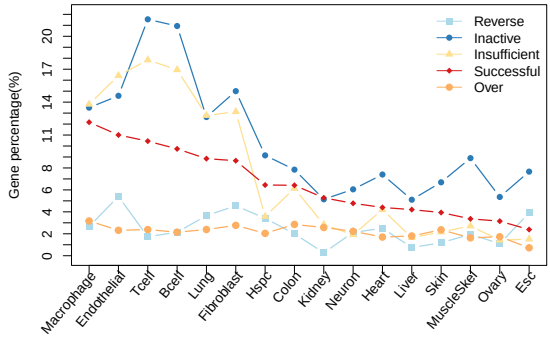

Figure 6: Percentage of genes in each category overlapping in each cell- and tissue-specific gene set

```
# CellNet C/T-specific enrichment analysis
cellnetenrich = enrichment(cate.gene = cate.gene, annotated.gene = human.gene,
                           background.gene = genes, padjust.method ="fdr")
cellnetheatmap = heatmapPlot(cellnetenrich,
                             main = "CellNet tissue specific enrichment")
```

10.2 Cell/Tissue-specific Transcription Factor Analysis

In our specific example, the observation that some *Inactive* genes are enriched in the primary cells, represents an important information regarding the transcription factors regulating these genes, TFs that are potentially necessary for a successful cellular engineering. Therefore in a second step, we extract the cell- and tissue-specific transcription factors and their down-stream regulated genes into gene sets, and apply the gene set enrichment analysis on the five gene categories. A heatmap can be plotted to compare the C/T-specific transcription factors enriched by different categories just as shown in Figure 7.

```
# load transcription factor regulated gene sets from on CellNet data
data(human.tf)
tfenrich = enrichment(cate.gene = cate.gene,  annotated.gene = human.tf,
                      background.gene = genes, padjust.method ="fdr")
tfheatmap = heatmapPlot(tfenrich, top = 5,
                        main = "CellNet transcription factor enrichment")
```

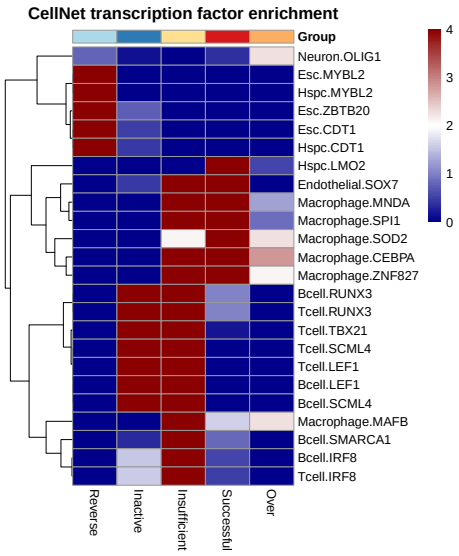

Figure 7: Top 5 significantly enriched cell/tissue-specific transcription factors by each gene category based on the log transformed corrected p-values

10.3 Network Topological Analysis and Visualization

Finally, we introduce the topological analysis for the C/T-specific gene regulatory networks including genes in each of five categories, by calculating the degree, closeness, betweenness and stress centrality with the *igraph* package [11] and *sna* [12]. Given an input of genes and their centrality, `grnPlot` function plots the regulatory network with these genes as nodes and centrality as node size to represent their importance in the network (Figure 8).

```
# load the CellNet GRN
data(human.grn)

# specify a tissue-specifc network
tissue = "Hspc"
degree = networkAnalyze(human.grn[[tissue]], cate.gene = cate.gene,
                        centrality = "degree", mode ="all")
head(degree)
```

```
##       Gene centrality.score  Type Category
## 977   ORC1            0.215 TF;TG     <NA>
## 1042 POLA1            0.214 TF;TG     <NA>
## 103  BRCA1            0.210 TF;TG     <NA>
## 235   CDT1            0.203 TF;TG     <NA>
## 1456 UHRF1            0.201 TF;TG     <NA>
## 595  HELLS            0.199 TF;TG     <NA>
```

```
# select genes to shown their regulation with others
node.genes = c("ZNF641", "BCL6")

# enlarge the centrality
centrality.score = degree$centrality*100
names(centrality.score) = degree$Gene
par(mar = c(2,2,3,2))
grnPlot(grn.data = human.grn[[tissue]], cate.gene = cate.gene, filter = TRUE,
        nodes = node.genes, centrality.score = centrality.score,
        main = "Gene regulatory network")
```

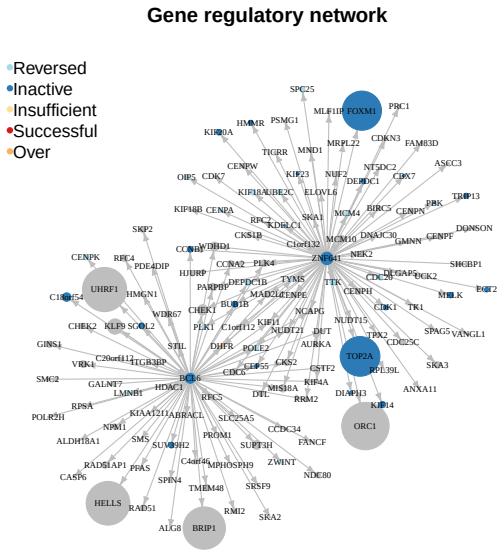

Figure 8: The hematopoietic stem/progenitor cell (Hspc)-specifc gene regulatory network regulated by "ZNF641" and "BCL6" genes
Node size is represented by the degree centrality of node in a general Hspc-specific network.

The analysis clarifies the importance of genes, especially the transcription factors, in terms of their ability to connect other genes in a network. Hence, this provides a way to predict the relevance of molecules in terms of their topological centrality, and offer information regarding potential transcription factors to be used for an improvement of cellular engineering.

References

[1] Sandler, V.M., et al., Reprogramming human endothelial cells to haematopoietic cells requires vascular induction. *Nature*, 2014. 511(7509): p. 312-8.

[2] Smyth, G.K., Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Appl Genet Mol Biol*, 2004. 3: p. Article3.

[3] Love, M.I., W. Huber, and S. Anders, Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol*, 2014. 15(12): p. 550.

[4] Yu, G., et al., clusterProfiler: an R package for comparing biological themes among gene clusters. *OMICS*, 2012. 16(5): p. 284-7.

[5] Ashburner, M., et al., Gene ontology: tool for the unification of biology. *The Gene Ontology Consortium. Nat Genet*, 2000. 25(1): p. 25-9.

[6] Kanehisa, M., et al., KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, 2012. 40(Database issue): p. D109-14.

[7] He, G.Y.a.L.-G.W.a.G.-R.Y.a.Q.-Y., DOSE: an R/Bioconductor package for Disease Ontology Semantic and Enrichment analysis. *Bioinformatics*, 2015. 31(4): p. 608-609.

[8] Benita, Y., et al., Gene enrichment profiles reveal T-cell development, differentiation, and lineage-specific transcription factors including ZBTB25 as a novel NF-AT repressor. *Blood*, 2010. 115(26): p. 5376-84.

[9] Cavalli, F., 2009. SpeCond: Condition specific detection from expression data. 2009.

[10] Morris, S.A., et al., Dissecting engineered cell types and enhancing cell fate conversion via CellNet. *Cell*, 2014. 158(4): p. 889-902.

[11] Nepusz, G.C.a.T., The igraph software package for complex network research. *InterJournal*, 2006. Complex Systems: p. 1695.

[12] Carter T. Butts, sna: Tools for Social Network Analysis. *R package version 2.3-2*, 2014.

11 Session Info

```
sessionInfo()

## R version 4.2.1 (2022-06-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.5 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.16-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.16-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB              LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] org.Hs.eg.db_3.16.0  AnnotationDbi_1.60.0 IRanges_2.32.0
## [4] S4Vectors_0.36.0     Biobase_2.58.0       BiocGenerics_0.44.0
## [7] eegc_1.24.0          knitr_1.40
##
## loaded via a namespace (and not attached):
##   [1] shadowtext_0.1.2           fastmatch_1.1-3
##   [3] plyr_1.8.7                 igraph_1.3.5
##   [5] lazyeval_0.2.2             splines_4.2.1
##   [7] BiocParallel_1.32.0        GenomeInfoDb_1.34.0
##   [9] ggplot2_3.3.6              digest_0.6.30
##  [11] yulab.utils_0.0.5          htmltools_0.5.3
##  [13] GOSemSim_2.24.0            viridis_0.6.2
##  [15] GO.db_3.16.0               fansi_1.0.3
##  [17] magrittr_2.0.3             memoise_2.0.1
##  [19] limma_3.54.0               sna_2.7
##  [21] Biostrings_2.66.0          annotate_1.76.0
##  [23] graphlayouts_0.8.3         wordcloud_2.6
##  [25] matrixStats_0.62.0         R.utils_2.12.1
##  [27] enrichplot_1.18.0          colorspace_2.0-3
```

```
##  [29] blob_1.2.3                 ggrepel_0.9.1
##  [31] xfun_0.34                  dplyr_1.0.10
##  [33] crayon_1.5.2               RCurl_1.98-1.9
##  [35] jsonlite_1.8.3             org.Mm.eg.db_3.16.0
##  [37] scatterpie_0.1.8           genefilter_1.80.0
##  [39] survival_3.4-0             ape_5.6-2
##  [41] glue_1.6.2                 polyclip_1.10-4
##  [43] gtable_0.3.1               zlibbioc_1.44.0
##  [45] XVector_0.38.0             DelayedArray_0.24.0
##  [47] scales_1.2.1               DOSE_3.24.0
##  [49] pheatmap_1.0.12            DBI_1.1.3
##  [51] edgeR_3.40.0               Rcpp_1.0.9
##  [53] viridisLite_0.4.1          xtable_1.8-4
##  [55] gridGraphics_0.5-1         tidytree_0.4.1
##  [57] bit_4.0.4                  httr_1.4.4
##  [59] fgsea_1.24.0               gplots_3.1.3
##  [61] RColorBrewer_1.1-3         pkgconfig_2.0.3
##  [63] XML_3.99-0.12              R.methodsS3_1.8.2
##  [65] farver_2.1.1               locfit_1.5-9.6
##  [67] utf8_1.2.2                 labeling_0.4.2
##  [69] ggplotify_0.1.0            tidyselect_1.2.0
##  [71] rlang_1.0.6                reshape2_1.4.4
##  [73] munsell_0.5.0              tools_4.2.1
##  [75] cachem_1.0.6               downloader_0.4
##  [77] cli_3.4.1                  generics_0.1.3
##  [79] RSQLite_2.2.18             statnet.common_4.7.0
##  [81] gson_0.0.9                 evaluate_0.17
##  [83] stringr_1.4.1              fastmap_1.1.0
##  [85] yaml_2.3.6                 ggtree_3.6.0
##  [87] bit64_4.0.5                tidygraph_1.2.2
##  [89] caTools_1.18.2             purrr_0.3.5
##  [91] KEGGREST_1.38.0            ggraph_2.1.0
##  [93] nlme_3.1-160               R.oo_1.25.0
##  [95] aplot_0.1.8                BiocStyle_2.26.0
##  [97] compiler_4.2.1             png_0.1-7
##  [99] treeio_1.22.0              tibble_3.1.8
## [101] tweenr_2.0.2               geneplotter_1.76.0
## [103] stringi_1.7.8              highr_0.9
## [105] lattice_0.20-45            Matrix_1.5-1
## [107] vctrs_0.5.0                pillar_1.8.1
## [109] lifecycle_1.0.3            BiocManager_1.30.19
## [111] data.table_1.14.4          cowplot_1.1.1
```

```
## [113] bitops_1.0-7                patchwork_1.1.2
## [115] GenomicRanges_1.50.0        qvalue_2.30.0
## [117] R6_2.5.1                    network_1.18.0
## [119] KernSmooth_2.23-20          gridExtra_2.3
## [121] codetools_0.2-18            MASS_7.3-58.1
## [123] gtools_3.9.3                assertthat_0.2.1
## [125] SummarizedExperiment_1.28.0 DESeq2_1.38.0
## [127] withr_2.5.0                 GenomeInfoDbData_1.2.9
## [129] parallel_4.2.1              clusterProfiler_4.6.0
## [131] grid_4.2.1                  ggfun_0.0.7
## [133] coda_0.19-4                 tidyr_1.2.1
## [135] HDO.db_0.99.1               rmarkdown_2.17
## [137] MatrixGenerics_1.10.0       ggforce_0.4.1
```