Final Report

Alexander Karr

2024-04-23

Contents

Introduction:

Despite the OSKM reprogramming factors having been initially reported as inducing genome-wide transcriptional responses, actually, less than 1% of cells undergo *authentic* pluripotency reprogramming (become iPSCs). In reality, their pluripotency induction is extremely inefficient, slow, and stochastic. To develop a sense of what kinds of transcriptomic changes the Yamanaka factors struggle to induce, and consistently succeed at, the authors of the paper that most inspired me - called Human Transcription factor responsive to initial reprogrmaming predominantly undergo legitimate reprogramming during fibroblast conversion to iPSCs - developed the concept of the "reprogramome" – or the full complement of genes requiring reprogramming for a differentiated cell to become an iPSC. Reprogamming letigimacy, then, describes scenarios where genes fully reach iPSC expression patterns. One of the most essential steps towards pluripotency induction is erasing the differentiated cell's transcription factor system, and inducing the pluripotent TF system. These transcription factors can be thought of in four different categories in terms of their response to OSKM induction - in their progression from a differentiated state to that of an iPSC. The authors of the paper that most inspired me - *Human transcription factors responsive to initial reprogramming predominantly undergo legitimate reprogramming during fibroblast conversion to iPSCs* refer to this suite of TFs as the reprogrammome. It includes:

1. Transcription factors that were legitimately reprogrammed (their expression levels returned to or exceeded levels observed in the human embryonic stem cells).
2. Transcription factors that were partially reprogrammed.
3. Transcription factors that were aberrantly reprogrammed.
4. Transcription factors that were irresponsive to reprogramming.

My initial goal was to determine whether or not specific types of gene ontologies distributed differently amongst the reprogramming categories, in the hope of then developing a better nascent understanding of what pathways the Yamanaka factors are enacting to epigenetically remodel cells. Unfortunately, however, due to some limitations in my data that I will go into further depth on later, I had limited ability to compare the TF expression patterning of my control and OSKM-treated samples to ground-truth human embryonic stem cell data. With my two conditions, then, of RNA sequenced from the same human fibroblast cell line untreated and treated with the OSKM factors, I had to ask different questions. I couldn't define specific transcriptomic thresholds after which I could consider any TF expression counts to be officially iPSC-like (successfully reprogrammed), so I tried to boostrap a method of quantifying how much more enriched the set of TFs enriched in iPSCs as compared to fibroblasts (the upreprogramome) were in my data's significantly upregulated TF counts, and how much more enriched the set of TFs enriched in fibroblasts as compared to iPSCs (the downreprogramome) were in my data's significantly downregulated TF counts. My hypothesis was that we would see differential gene expression and GO enrichment patterns between the two conditions. But given that the process of reprogramming fibroblasts to iPSCs is known to take 2-3 weeks, I assumed that - if the upreprogramome and downreprogramome were enriched in upregulated and downregulated genes, respectively - that the degree of enrichment would be mild.

Results:

Dataset limitations

To preface, I will first lay out the limitations of my dataset, both potential and definite. Firstly - although I did my very best to address this by testing for custom BGI adapter sequences when using fastQC, I know that the BGI sequencing platform can sometimes produce peculiar data. Secondly - I have only three replicates per condition. Thirdly, the human ESC samples from the original paper that I intended to use as a reference point for the iPSC-like-ness of my OSKM-treated sample had even fewer replicates, and I was thus not comfortable using it. And finally, differential gene expression among solely transcription factors, and trying to make strong conclusions from TF gene counts at particular time points, can be noisy and risky. TF have short half-lives, can be expressed in short bursts, and are mostly not ubiquitously expressed. I detail my response to this in the methods section.

Processing the featureCounts count table.

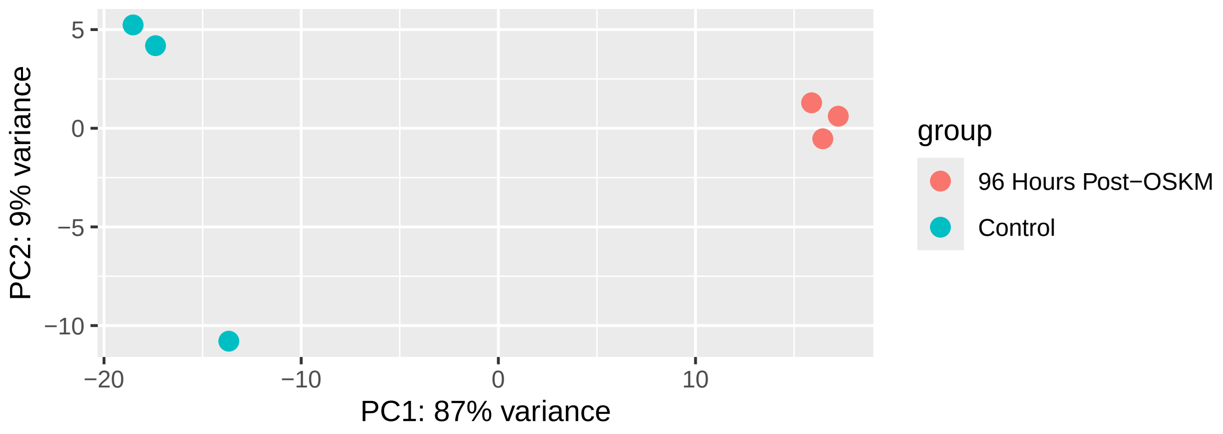
```
## Warning: package 'DESeq2' was built under R version 4.3.3

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows
## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Warning: package 'GenomeInfoDb' was built under R version 4.3.3

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

## Warning: package 'matrixStats' was built under R version 4.3.3

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.


##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians
## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## Warning: package 'ggplot2' was built under R version 4.3.3

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:GenomicRanges':
##
##     subtract

## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## gene-wise dispersion estimates

## mean-dispersion relationship

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## final dispersion estimates

## Warning: package 'tidyr' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%within%() masks IRanges::%within%()
## x dplyr::collapse()     masks IRanges::collapse()
## x dplyr::combine()      masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count()        masks matrixStats::count()
## x dplyr::desc()         masks IRanges::desc()
## x tidyr::expand()       masks S4Vectors::expand()
## x tidyr::extract()      masks magrittr::extract()
## x dplyr::filter()       masks stats::filter()
## x dplyr::first()        masks S4Vectors::first()
## x dplyr::lag()          masks stats::lag()
## x ggplot2::Position()   masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()       masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()       masks S4Vectors::rename()
## x lubridate::second()   masks S4Vectors::second()
## x lubridate::second<-() masks S4Vectors::second<-()
## x purrr::set_names()    masks magrittr::set_names()
## x dplyr::slice()        masks IRanges::slice()
## x magrittr::subtract()  masks GenomicRanges::subtract()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

TF PCA Plotting by Condition

The first result that I will demonstrate is just an exploratory PCA on variance-stabilization-transformed TF data. We can see an extremely clear separating between thw two conditions, especially given that the x-axis represents 87% of the variacne, and thus the x-plane-dominated distance between the conditions bespeaks truly large transcriptomic differences. One control sample may appear to vary significantly from the other two, but most of that difference is y-distance, with PC2 only explaining 9% of the variance in the dataset.

```
plotPCA(vst(tf_dds))
```

```
## using ntop=500 top features by variance
```



Differential Gene Expression

I originally used Benjamini-Hochberg, and while the Yamanaka factors are renownedly transcriptionally transformative, and I thus expect highly differential expression, My general prior is not quite so high as 50% of TFs showing significantly differential expression, and I am still very wary of the erraticness of TF expression patterns.

I will illustrate some overall differential gene expression analysis of the two conditions, before delving into specific GO term enrichment analysis

```
## Loading required package: AnnotationDbi
```

```
##
## Attaching package: 'AnnotationDbi'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

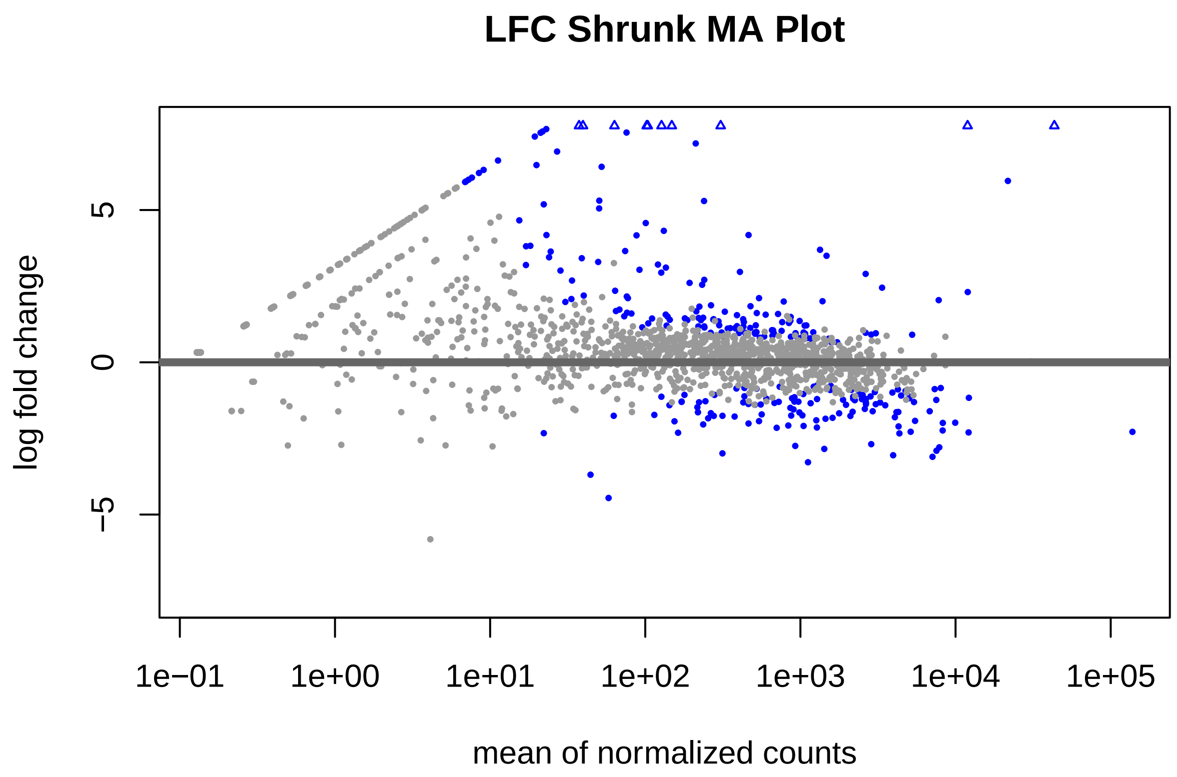
```
##
```

```
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

Our MA plot is showing more signifiant results in the direction of upregulation. And, although the difference is not exceptionally significant, we are seeing the expected pattern manifesting of finding more differentially expressed genes as one moves towards the right hand side of the plot, where we find more strongly expressed genes.

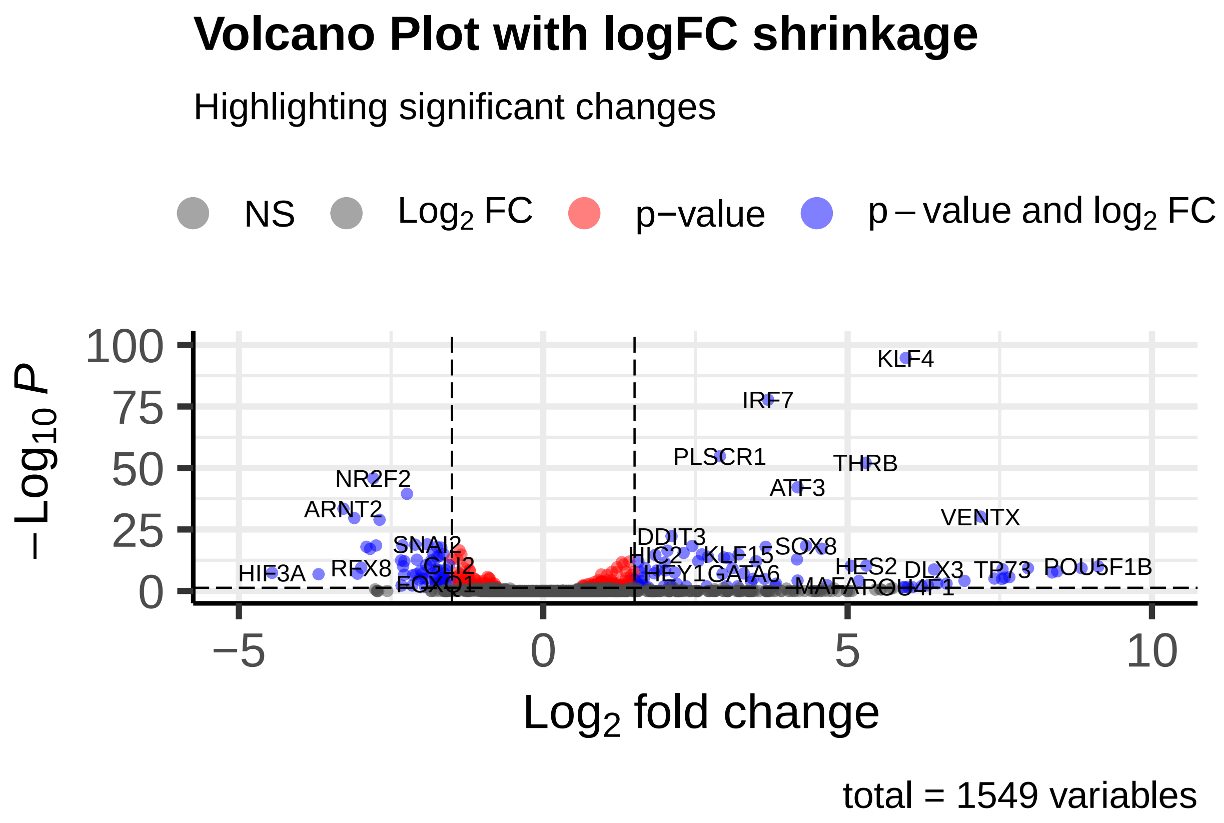
```r
plotMA(shrunk_tf_results_sorted, alpha=0.05,
main="LFC Shrunk MA Plot")
```



```r
library(EnhancedVolcano)
```

```
## Loading required package: ggrepel

## Warning: package 'ggrepel' was built under R version 4.3.3
```

```r
# Handling p-values of 0 by adjusting to a small number if necessary
shrunk_tf_results_sorted$padj[shrunk_tf_results_sorted$padj == 0] <- min(shrunk_tf_results_sorted$padj[s

# Updated EnhancedVolcano call
vp2 <- EnhancedVolcano(shrunk_tf_results_sorted,
                       lab = shrunk_tf_results_sorted$gene_symbol_or_ORF,
                       x = 'log2FoldChange',
                       y = 'padj',
                       pCutoff = 0.05,
                       FCcutoff = 1.5,  # Log2 fold change cutoff
                       title = "Volcano Plot with logFC shrinkage",
                       subtitle = "Highlighting significant changes",
                       col=c('grey30', 'grey30', 'red', 'blue'),  # Colors: non-significant, significan
                       labSize = 3.0,
                       pointSize = c(1.5),  # Ensure this matches the data size; might need adjustment
                       xlim = c(-5, 10))

# Print the plot
print(vp2)
```



This volcano plot further communicates that we are finding more significance among upregulated genes (OSKM-treatment-enriched). Some standouts in terms of significance are KLF4, IRF7, and NR2F2 on the negative L2FC side.

Individual gene expression

I look for the top five most positively (higher expression in the 96 hour sample) and negatively expressed genes (lower expression in the 96 hour sample), and report their summaries.

```r
# Selecting the top ten most significant genes
top_ten_genes <- head(shrunk_tf_results_sorted, 10)

# Displaying the gene symbols, log fold changes, and adjusted p-values of the top ten genes
top_ten_summary <- top_ten_genes[, c("gene_symbol_or_ORF", "log2FoldChange", "padj")]
print(top_ten_summary)
```

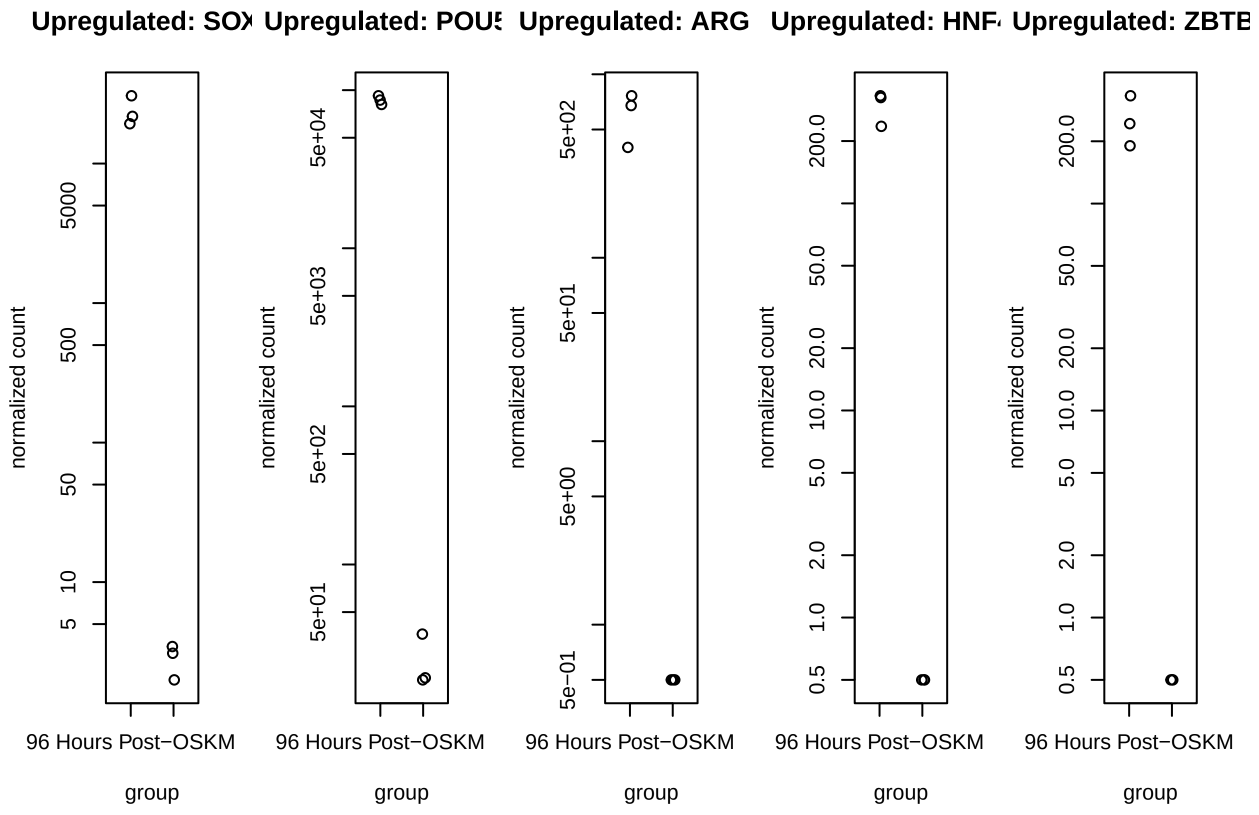
```
## log2 fold change (MLE): condition 96 Hours Post-OSKM vs Control
##
## DataFrame with 10 rows and 3 columns
##                 gene_symbol_or_ORF log2FoldChange        padj
##                        <character>      <numeric>   <numeric>
## ENSG00000204531             POU5F1       11.84677 1.95054e-96
## ENSG00000136826               KLF4        5.95566 1.95054e-95
## ENSG00000181449               SOX2       13.26263 1.80428e-92
## ENSG00000185507               IRF7        3.69423 1.73650e-78
## ENSG00000188313             PLSCR1        2.90243 1.66911e-55
## ENSG00000151090              THRB        5.29659 7.43913e-53
## ENSG00000185551             NR2F2       -2.79261 1.56144e-46
## ENSG00000162772              ATF3        4.17905 7.16016e-43
## ENSG00000149948             HMGA2       -2.23856 3.17670e-40
## ENSG00000172379             ARNT2       -3.28282 3.69143e-34
```

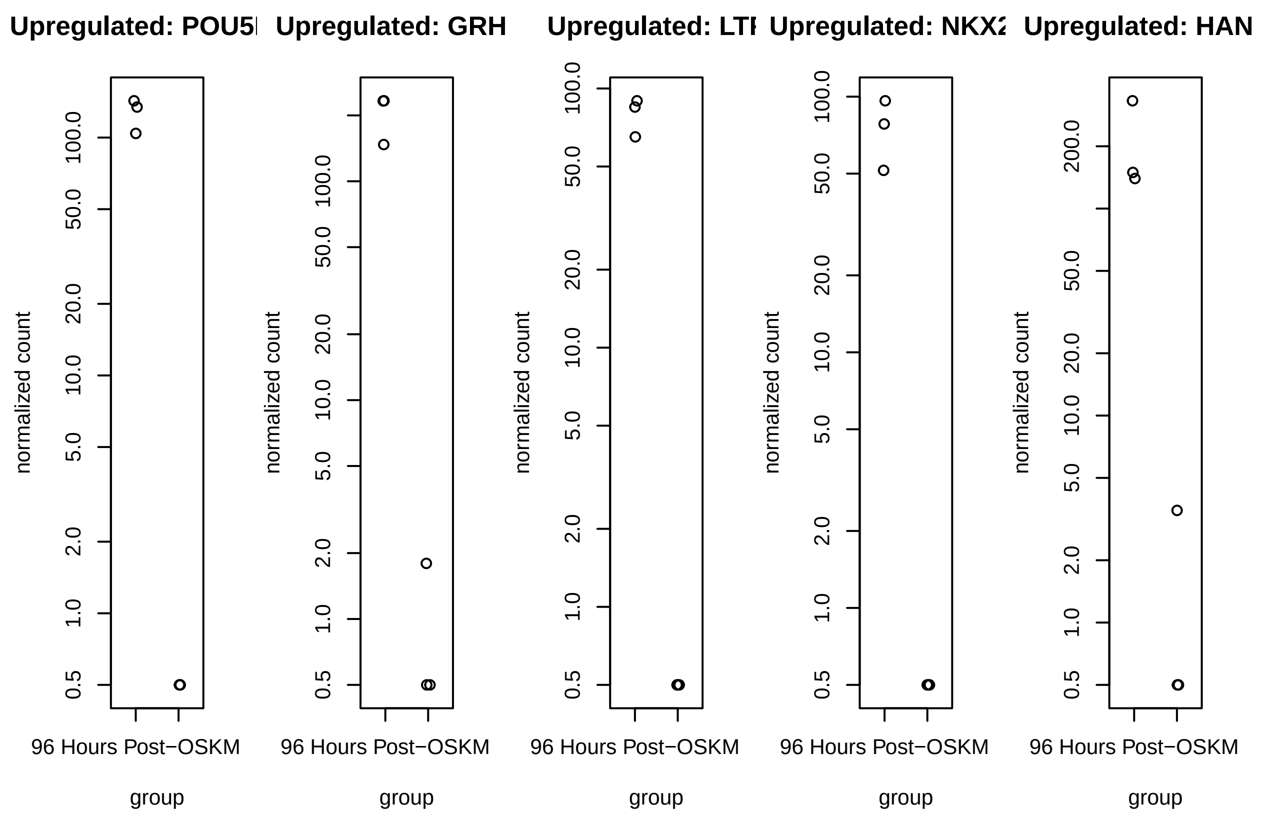
I will then include some example plots of the control-condition comparison of actual normalized expression counts for the top ten most significant down-regulated and up-regulated TF genes.

```r
# Setup the plotting parameters for five plots per display
par(mfrow=c(1, 5))  # 1 row, 5 columns

# Iterate over the gene list and plot normalized counts
plot_gene_counts <- function(genes_df) {
  for (i in 1:nrow(genes_df)) {
    gene_id <- rownames(genes_df)[i]
    gene_symbol <- genes_df$gene_symbol_or_ORF[i]
    direction <- ifelse(genes_df$log2FoldChange[i] > 0, "Upregulated", "Downregulated")

    plot_title <- paste(direction, gene_symbol, sep=": ")

    # Plot counts for this gene
    plotCounts(dds_yamanaka, gene=gene_id, intgroup="condition", normalized=TRUE, main=plot_title)

    # Refresh the plotting device after every five plots or at the end of the list
    if (i %% 5 == 0 || i == nrow(genes_df)) {
      # This ensures a new image starts if there are more genes to plot
      if (i != nrow(genes_df)) {

      }
    }
  }
}

# Run the plotting function for the top ten positive log fold changes
cat("Displaying Upregulated Transcription Factors:\n")
```

```
## Displaying Upregulated Transcription Factors:
```

```r
plot_gene_counts(top_ten_positive)
```

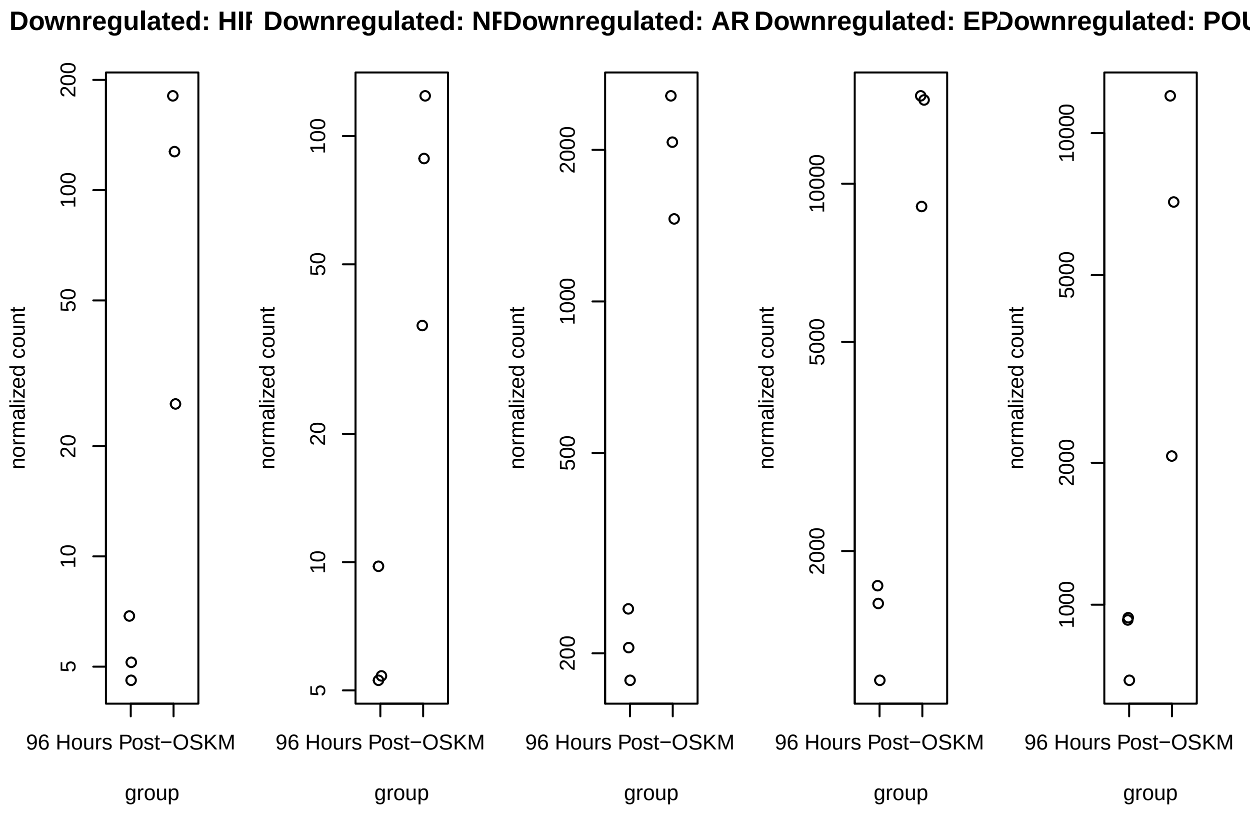

Upregulated: SOX
Upregulated: POU5
Upregulated: ARG
Upregulated: HNF4
Upregulated: ZBTB
normalized count
5
10
50
500
5000
5e+01
5e+02
5e+03
5e+04
5e−01
5e+00
5e+01
5e+02
0.5
1.0
2.0
5.0
10.0
20.0
50.0
200.0
96 Hours Post−OSKM
group

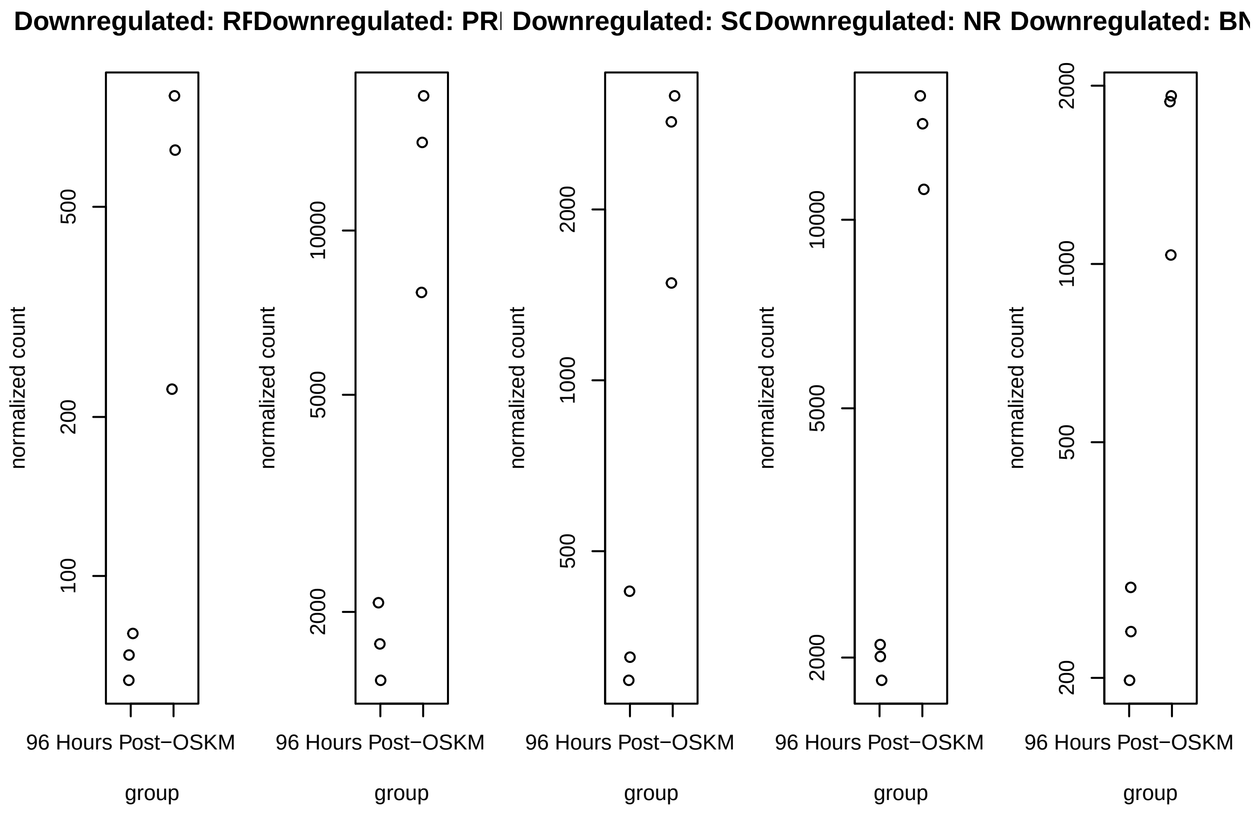


```r
par(mfrow=c(1, 5))  # Setup for new series of plots
cat("Displaying Downregulated Transcription Factors:\n")
```

```
## Displaying Downregulated Transcription Factors:
```

```r
plot_gene_counts(top_ten_negative)
```


Downregulated: HII
Downregulated: NI
Downregulated: AR
Downregulated: EP
Downregulated: POU
normalized count
96 Hours Post−OSKM
group
200
100
50
20
10
5
2000
1000
500
10000
5000



## Gene ontology

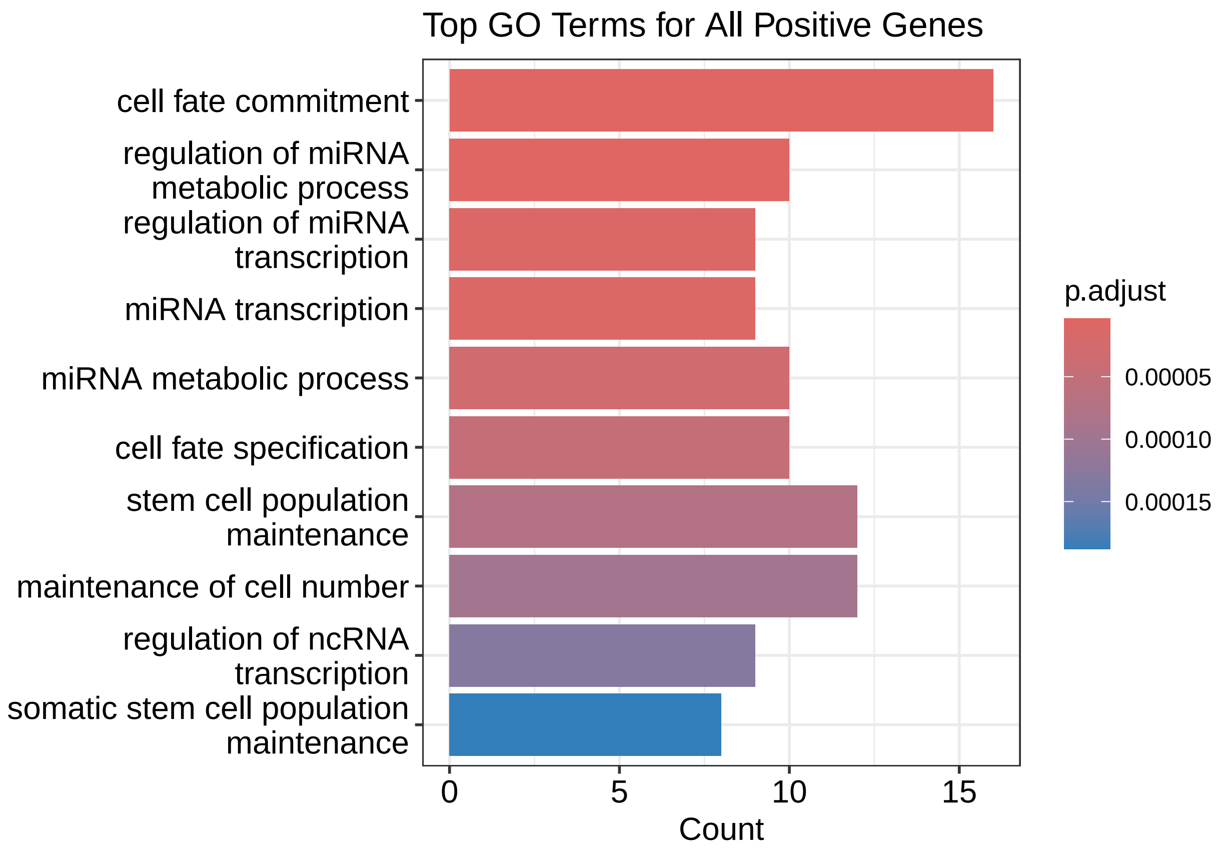
For purposes of interpretability, however, I am more interested in gene ontology information and how it separates the conditions. The goal is to get a general impression of - assuming that we are seeing signal and not merely the noise of TF's erratic transcriptomic patterning - what biological processes the Yamanaka factors are promoting/inhibiting.

I made some GO visualizations for the entire set of positive L2FC, statistically significant, 96-hour-sample-enriched genes, and the entire set of negative L2FC, statistically signficant, untreated-fibroblast-enriched genes.

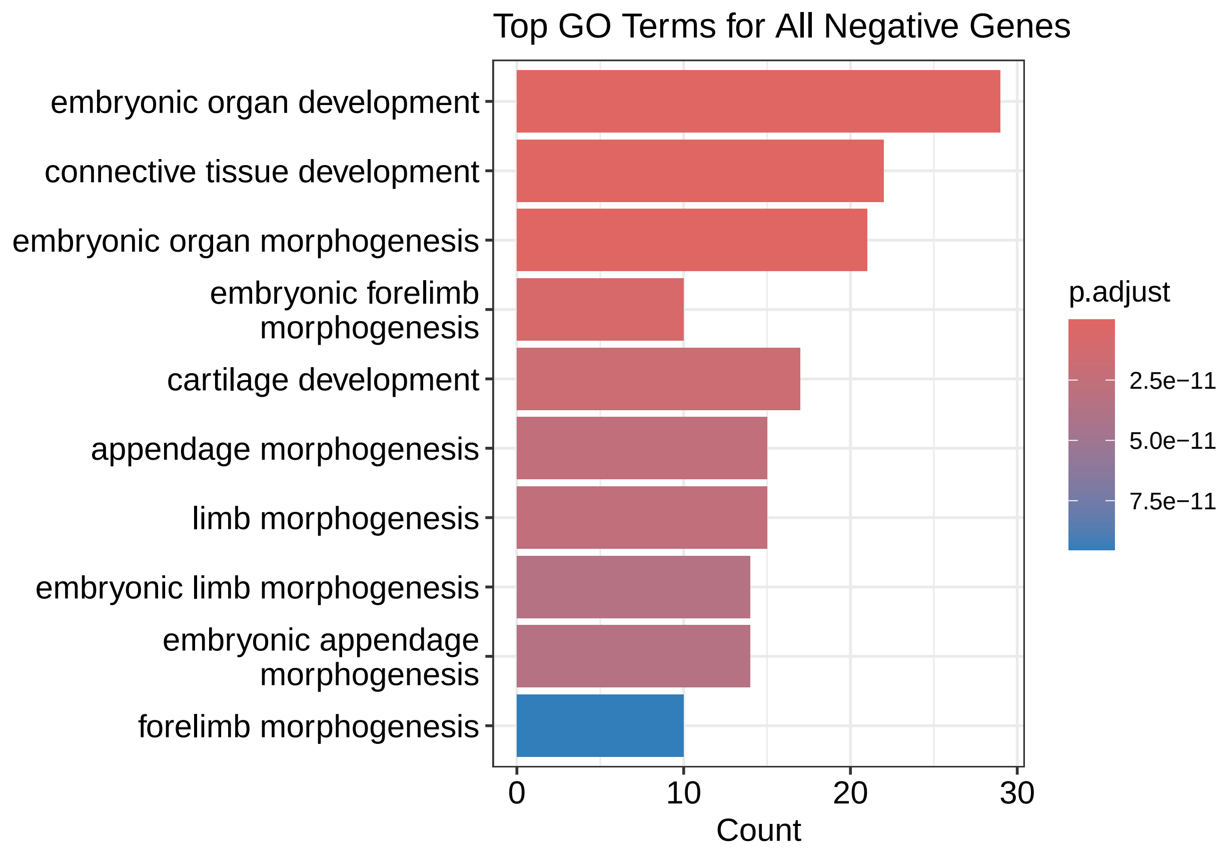
I begin with the larger sets.

```
## Warning: package 'clusterProfiler' was built under R version 4.3.3

##

## clusterProfiler v4.10.1  For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use clusterProfiler in published research, please cite:
## T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu.

##
## Attaching package: 'clusterProfiler'

## The following object is masked from 'package:AnnotationDbi':
##
##     select
## The following object is masked from 'package:purrr':
##
##     simplify

## The following object is masked from 'package:IRanges':
##
##     slice

## The following object is masked from 'package:S4Vectors':
##
##     rename

## The following object is masked from 'package:stats':
##
##     filter

## 'select()' returned 1:1 mapping between keys and columns
## 'select()' returned 1:1 mapping between keys and columns
```

```
# Bar plot of GO terms
barplot(ego_pos, showCategory=10) + ggtitle("Top GO Terms for All Positive Genes")
```



```
barplot(ego_neg, showCategory=10) + ggtitle("Top GO Terms for All Negative Genes")
```



```
# Convert gene symbols to ENTREZ IDs if necessary
positive_entrez <- bitr(positive_genes, fromType="SYMBOL", toType="ENTREZID", OrgDb="org.Hs.eg.db")
```

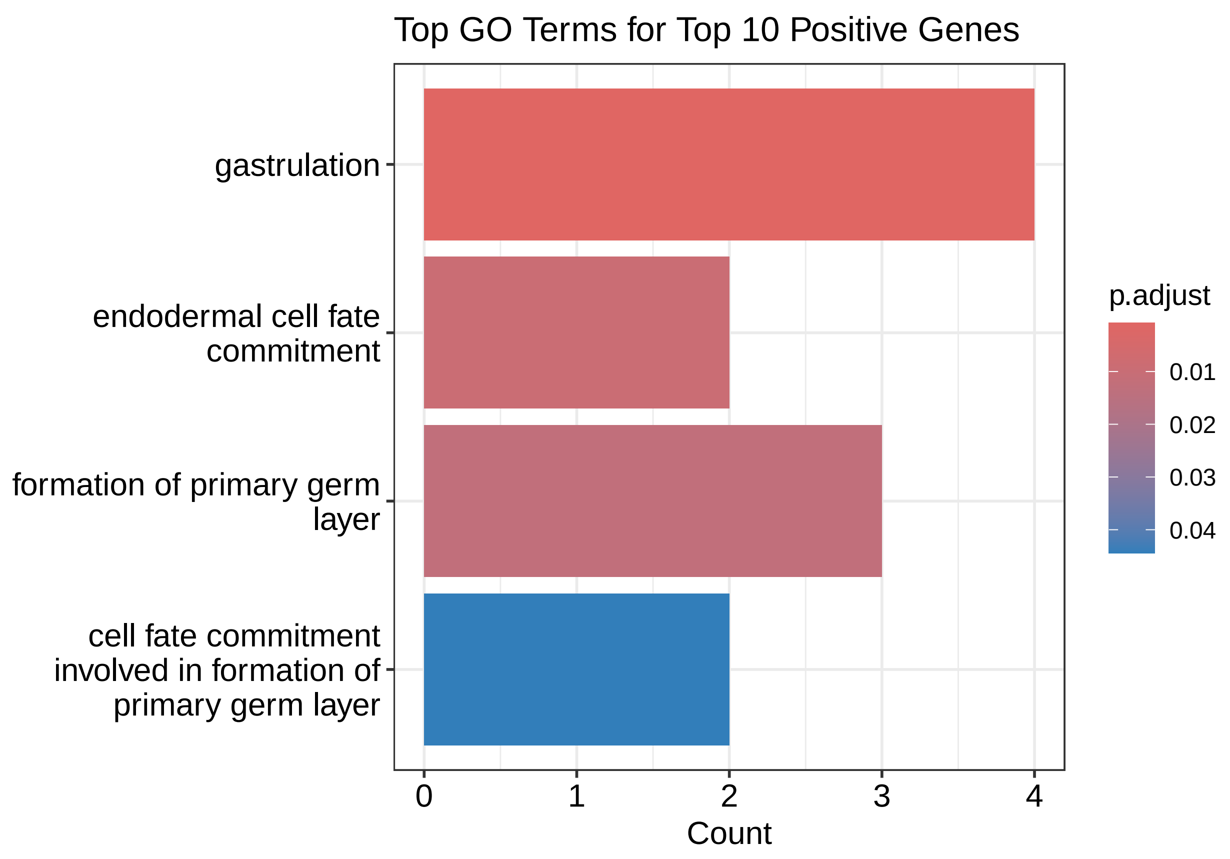
```
## 'select()' returned 1:1 mapping between keys and columns
```

```
negative_entrez <- bitr(negative_genes, fromType="SYMBOL", toType="ENTREZID", OrgDb="org.Hs.eg.db")
```

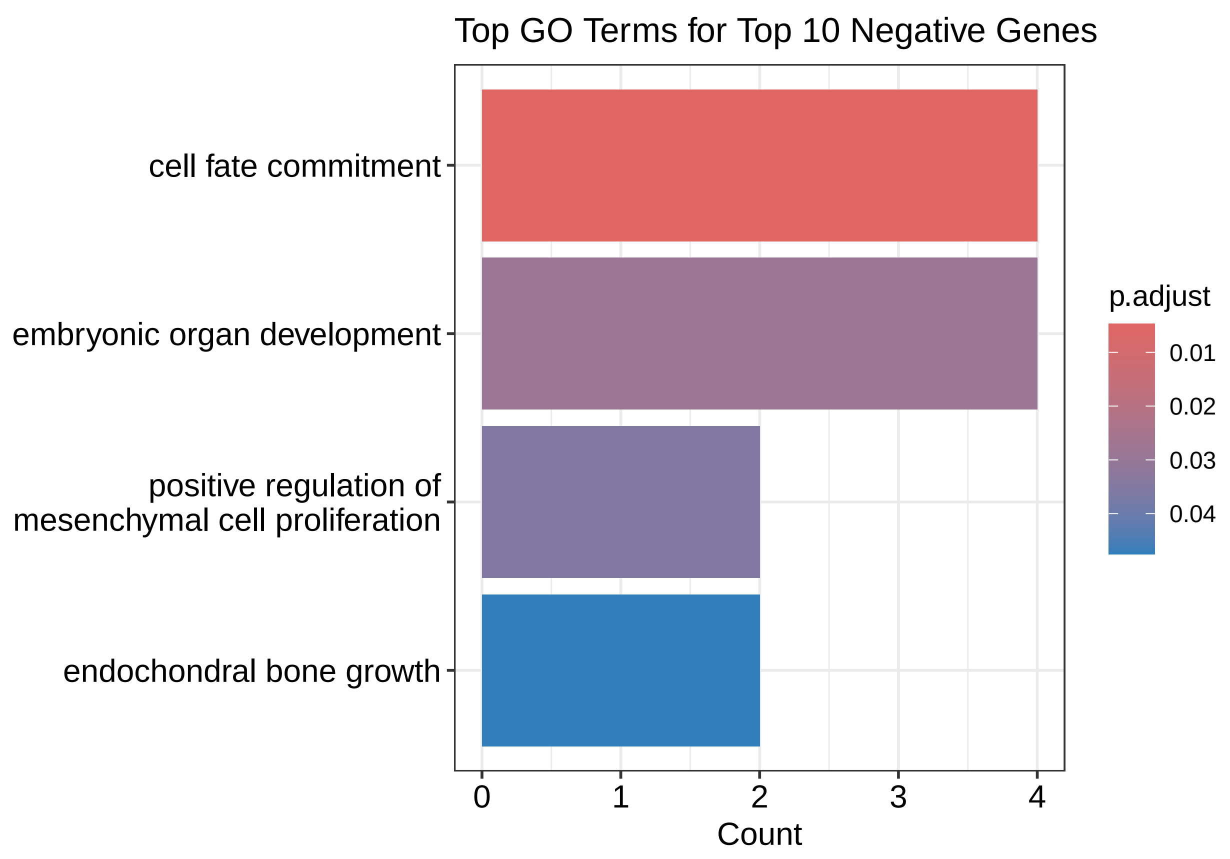
```
## 'select()' returned 1:1 mapping between keys and columns
```

```
# Perform GO enrichment analysis
ego_pos_ten <- enrichGO(gene          = positive_entrez$ENTREZID,
                    OrgDb         = org.Hs.eg.db,
                    keyType       = "ENTREZID",
                    ont           = "BP",  # Biological Process
                    pAdjustMethod = "holm",
                    qvalueCutoff  = 0.05)

ego_neg_ten <- enrichGO(gene          = negative_entrez$ENTREZID,
                    OrgDb         = org.Hs.eg.db,
                    keyType       = "ENTREZID",
                    ont           = "BP",
                    pAdjustMethod = "holm",
                    qvalueCutoff  = 0.05)
```

```r
# Bar plot of GO terms
barplot(ego_pos_ten, showCategory=10) + ggtitle("Top GO Terms for Top 10 Positive Genes")
```



```r
barplot(ego_neg_ten, showCategory=10) + ggtitle("Top GO Terms for Top 10 Negative Genes")
```



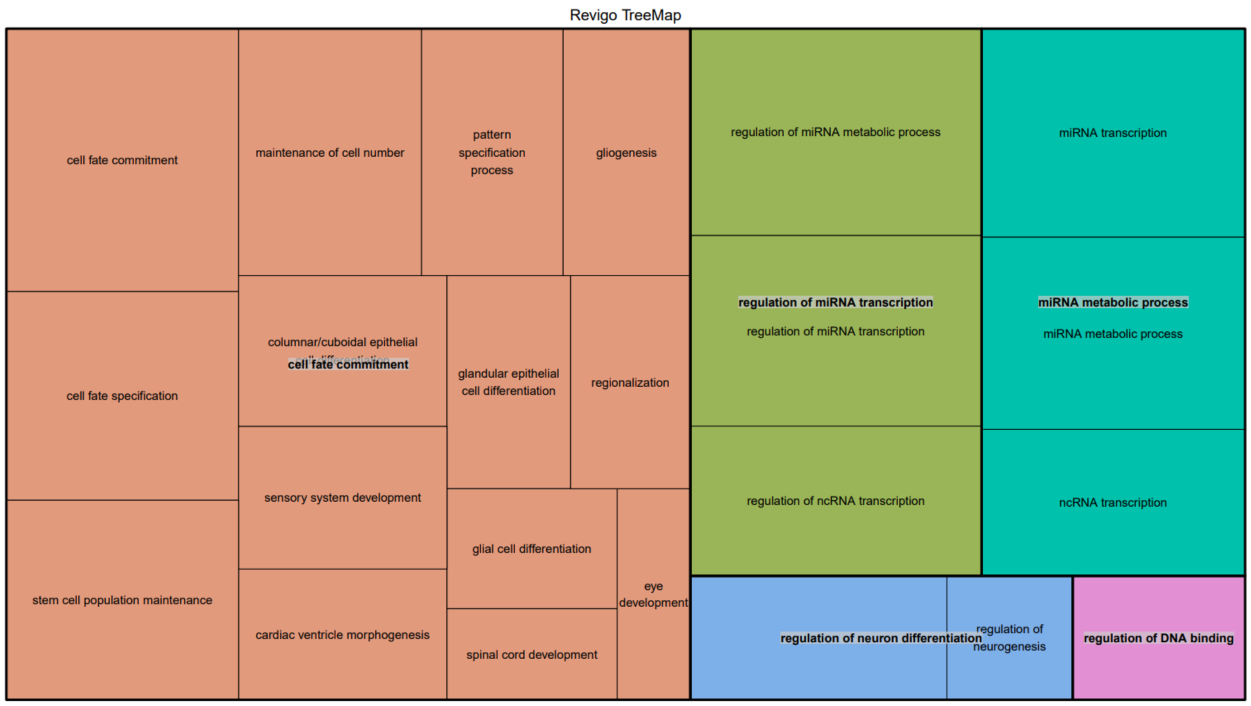
I then made some Revigo treemaps to simplify the GO compositions of these four categories even further, and level them for comparison as much as possible.

```r
# Assuming 'ego_pos' is your enrichGO result for all positive genes
ego_pos_data <- as.data.frame(ego_pos)
revigo_input_pos <- ego_pos_data[, c("ID", "p.adjust")]

# Write this data to a CSV file for uploading to REVIGO
write.csv(revigo_input_pos, "revigo_input_pos.csv", row.names = FALSE)

# Assuming 'ego_pos_ten' is your enrichGO result for the top ten positive genes
ego_pos_ten_data <- as.data.frame(ego_pos_ten)
revigo_input_pos_ten <- ego_pos_ten_data[, c("ID", "p.adjust")]

# Write this data to a CSV file for uploading to REVIGO
write.csv(revigo_input_pos_ten, "revigo_input_pos_ten.csv", row.names = FALSE)

# Assuming 'ego_neg' is your enrichGO result for all negative genes
ego_neg_data <- as.data.frame(ego_neg)
revigo_input_neg <- ego_neg_data[, c("ID", "p.adjust")]

# Write this data to a CSV file for uploading to REVIGO
write.csv(revigo_input_neg, "revigo_input_neg.csv", row.names = FALSE)

# Assuming 'ego_neg_ten' is your enrichGO result for the top ten negative genes
ego_neg_ten_data <- as.data.frame(ego_neg_ten)
```

```r
revigo_input_neg_ten <- ego_neg_ten_data[, c("ID", "p.adjust")]

# Write this data to a CSV file for uploading to REVIGO
write.csv(revigo_input_neg_ten, "revigo_input_neg_ten.csv", row.names = FALSE)
```

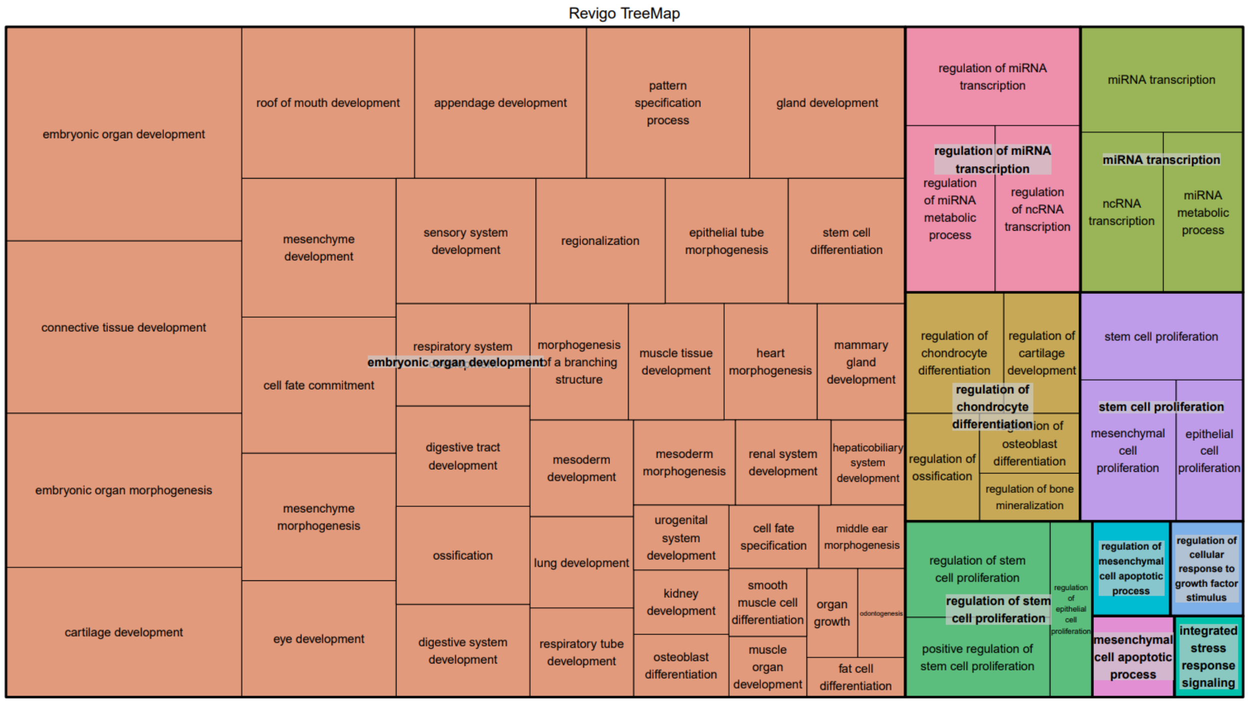
I screenshoted the revigo reduced ontology results, and include them below. Running the code on my .rmd document will download them as a csv to your local computer, however, and you can run the contents through Revigo's querying box if you so desire. I set the genome reference to humans, from amongst their options.

Revigo Treemap of all significant positive LFC genes, enriched in the 96 hour condition:



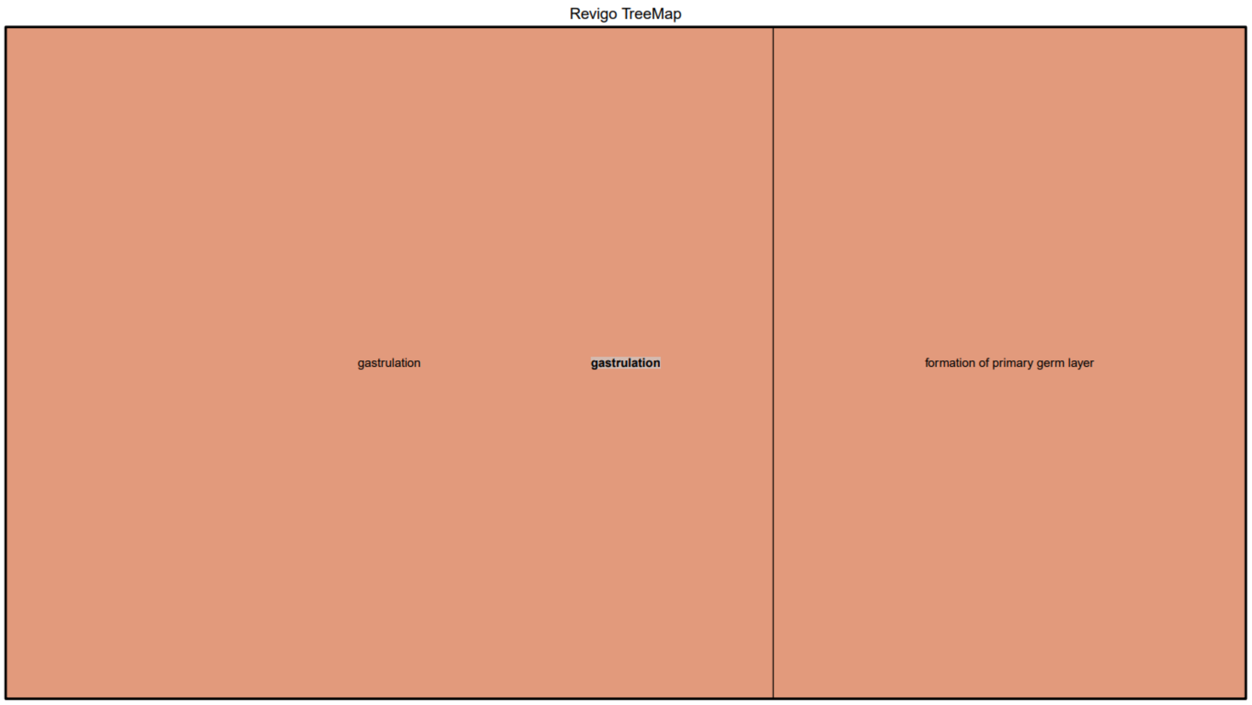
We're seeing some intuitive GO categories, like stem cell population maintenance, but some unexpected ones, like cell fate commitment. Many GO terms indicate TFs that help "regulate" other processes. We don't know whether this is positive or negative regulation, so Revigo in general may yield slightly more inscrutable results with Transcription Factors, given that their annotations in general are not as clear as, say, a protein-encoding gene.

Revigo Treemap of all significant negative LFC genes, enriched in the untreated fibroblast condition:



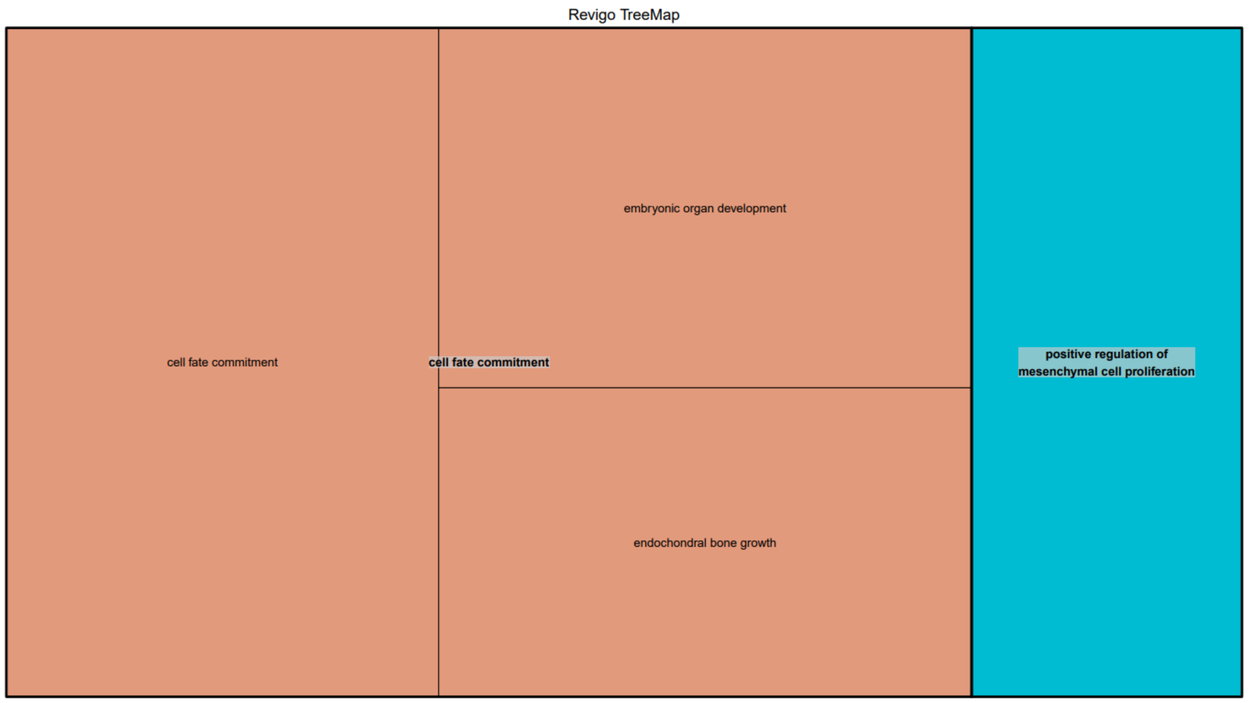
The embryonic organ development" semantic group, which very much dominates the GO composition, is aggregating developmental/cell differentiation processes, for the most part. This is very much as to be expected if the trajectory of Yamanaka factor induction is meant to take a fibroblast from differentiated to stem-like.

Revigo Treemap of top ten most significant positive LFC OSKM-treatment-enriched genes:



Revigo Treemap of top ten most significant negative LFC untreated-fibroblast-enriched genes:

Gastrulation is a crucial phase early in the embryonic development of animals, during which the single-layered blastula is reorganized into a multilayered structure known as the gastrula. It's essentially the process by which the embryo transforms from a simple spherical ball of cells into a structure with multiple layers and a defined axis, which will give rise to distinct tissues and organs. Figuring this out, and seeing this so enriched in the most significantly upregulated genes in the whole TF dataset, confused me at first. This is a process by which embryos transform out of their initial shape, which made me immediately assume that it was more in the category of cell fate determination and differentiation. However, given that the Yamanaka factors revert differentiated cells back to iPSCs in a continuous manner, it makes sense that we'd see an enrichment for embryonic traits - even if the phenotypes they describe transition embryos away from their embryonic-ness.



This group is small enough that Revigo is not spectacularly enlightening, but the single largest GO term being downregulated is cell fate commitment, further in-keeping with our intuitions. Interestingly, however, "cell fate commitment" ontologies are listed very prominently amongst the top ten most significantly untreated-fibroblast-enriched genes. Yet it's also a sizable ontology among the greater set of 96-hour-enriched genes. I can imagine, however, how cells becoming more stem-like could involve downregulating TFs that help preserve a current cell fate trajectory *and* upregulating TFs that have to be more active in steering a more embryonic cell towards an ultimate cell fate (and are thus markers of a more stem-like cell).

And finally, I proceed to the GSEA portion of my results.

I now load a table from the supplementary information of the paper I have most referenced, that includes information that will be the basis for my constructing the gene sets of the "upreprogramome" and "down-reprogramome" (see introduction). They compared human fibroblasts and human embryonic stem cells TF counts to establish each's relative patterning. And while I wasn't comfortable using their hESC data in direct comparison with mine (to establish whether or not any of my TF gene counts appear iPSC-like) given that it was generated with subtle differences in QC and processing from mine, I was comfortable extracting the list of TFs that they assigned to the upreprogamome and downreprogamome to use as gene sets in GSEA for my own case-control study.

And again, for clarity's sake:

1. The TF upreprogramome refers to transcription factors that have to be upregulated in the fibroblast for pluripotency induction.

2. The TF downreprogramome refers to fibroblast-enriched and specific transcriptional factors that have to be downregulated for pluripotency.

```
## Warning: package 'readxl' was built under R version 4.3.3
```

Processing the upreprogramome and downreprogramome in

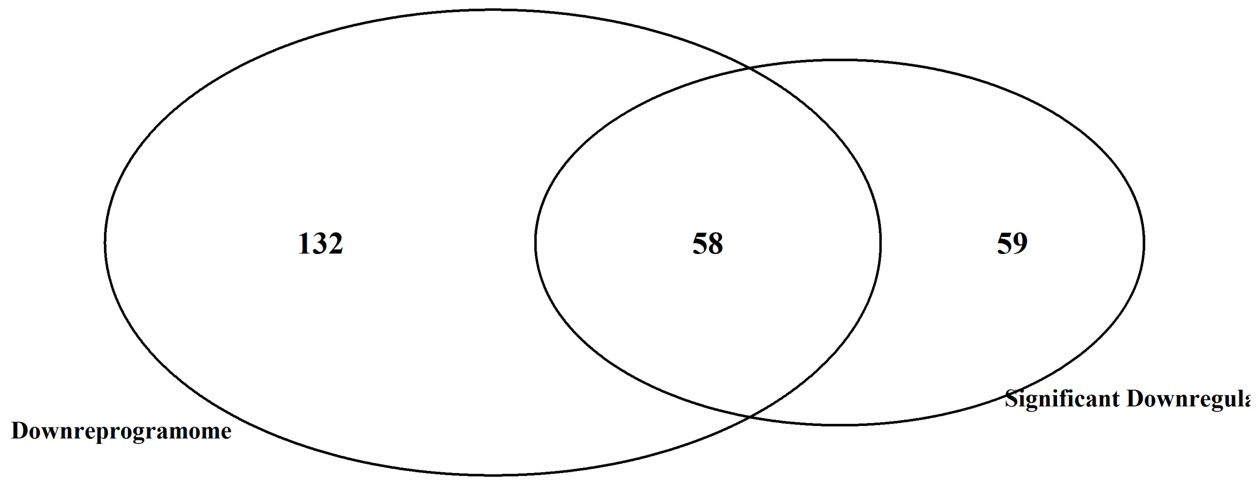
```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `log2FoldChange = as.numeric(as.character(log2FoldChange))`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

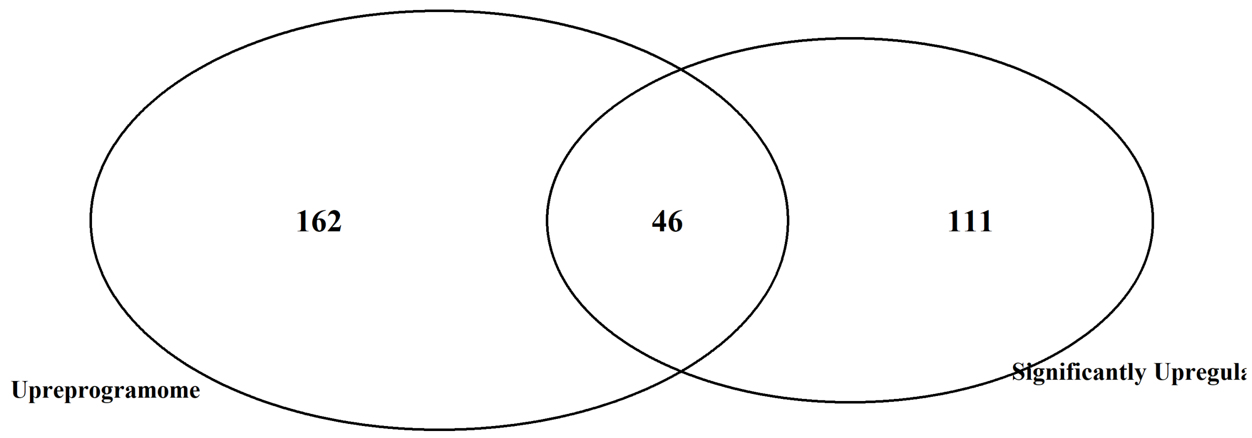
```r
library(VennDiagram)
```

```
## Warning: package 'VennDiagram' was built under R version 4.3.3

## Loading required package: grid

## Loading required package: futile.logger

## Warning: package 'futile.logger' was built under R version 4.3.3
```

```r
library(grid)

# Define dimensions in inches
width_in_inches <- 10
height_in_inches <- 6

# Create Venn Diagrams and save as PNG files
venn.plot.up <- venn.diagram(
  x = list(
    Significant_Up = significant_up_gene_ensembl,
    Upreprogramome = upreprogramome_gene_ensembl
  ),
  category.names = c("Significantly Upregulated", "Upreprogramome"),
  filename = "venn_diagram_up.png",  # Saving as PNG file
  output = TRUE,
  imagetype = "png",
  height = height_in_inches,  # height in inches
  width = width_in_inches,    # width in inches
  resolution = 300,
  compression = "lzw",
  units = "in",  # specify units
  lwd = 2,       # line width
  cex = 1.5,     # font size scaling factor
  fontface = "bold",  # font style
  cat.cex = 1.2,      # category label scaling factor
  cat.fontface = "bold",  # category label font style
  cat.default.pos = "outer",  # category label position
  cat.dist = 0.05,   # distance for category labels from the center of the diagram
  margin = 0.05      # margin around the plot
)
```

```
venn.plot.down <- venn.diagram(
  x = list(
    Significant_Down = significant_down_gene_ensembl,
    Downreprogramome = downreprogramome_gene_ensembl
  ),
  category.names = c("Significant Downregulated", "Downreprogramome"),
  filename = "venn_diagram_down.png",  # Saving as PNG file
  output = TRUE,
  imagetype = "png",
  height = height_in_inches,
  width = width_in_inches,
  resolution = 300,
  compression = "lzw",
  units = "in",
  lwd = 2,
  cex = 1.5,
  fontface = "bold",
  cat.cex = 1.2,
  cat.fontface = "bold",
  cat.default.pos = "outer",
  cat.dist = 0.05,
  margin = 0.05
)
```





I generate simple venn diagrams for a clear first introduction to the overlap of the up/down-reprogamome and our significantly upregulated/downregulated control/OSKM data. The down venn diagram has 58 overlapping TFs, and the up venn diagram has 46. Obviously, this visualization is agnostic to a TF's degree of L2FC and p value.

GSEA

```
# Prepare gene sets from the filtered dataframes
gene_sets <- list(
  Upreprogramome = upreprogramome_genes$gene_symbol_or_ORF,
  Downreprogramome = downreprogramome_genes$gene_symbol_or_ORF
)

# Run GSEA for upregulated genes
# 'scoreType = "pos"' focuses the analysis on the positive side of the ranked list
gsea_results_up <- fgsea(pathways = gene_sets["Upreprogramome"],
                         stats = ranked_list,
                         scoreType = "pos",
                         minSize = 15,    # Minimum size of gene set to consider
                         maxSize = 500,   # Maximum size of gene set to consider
                         nperm = 10000)   # Number of permutations for estimation
```

```
## Warning in fgsea(pathways = gene_sets["Upreprogramome"], stats = ranked_list, :
## You are trying to run fgseaSimple. It is recommended to use fgseaMultilevel. To
## run fgseaMultilevel, you need to remove the nperm argument in the fgsea
## function call.

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize,
## gseaParam, : There are duplicate gene names, fgsea may produce unexpected
## results.
```

```
# Run GSEA for downregulated genes
# 'scoreType = "neg"' focuses the analysis on the negative side of the ranked list
gsea_results_down <- fgsea(pathways = gene_sets["Downreprogramome"],
                           stats = ranked_list,
                           scoreType = "neg",
                           minSize = 15,
                           maxSize = 500,
                           nperm = 10000)
```

```
## Warning in fgsea(pathways = gene_sets["Downreprogramome"], stats = ranked_list,
## : You are trying to run fgseaSimple. It is recommended to use fgseaMultilevel.
## To run fgseaMultilevel, you need to remove the nperm argument in the fgsea
## function call.

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize,
## gseaParam, : There are duplicate gene names, fgsea may produce unexpected
## results.
```

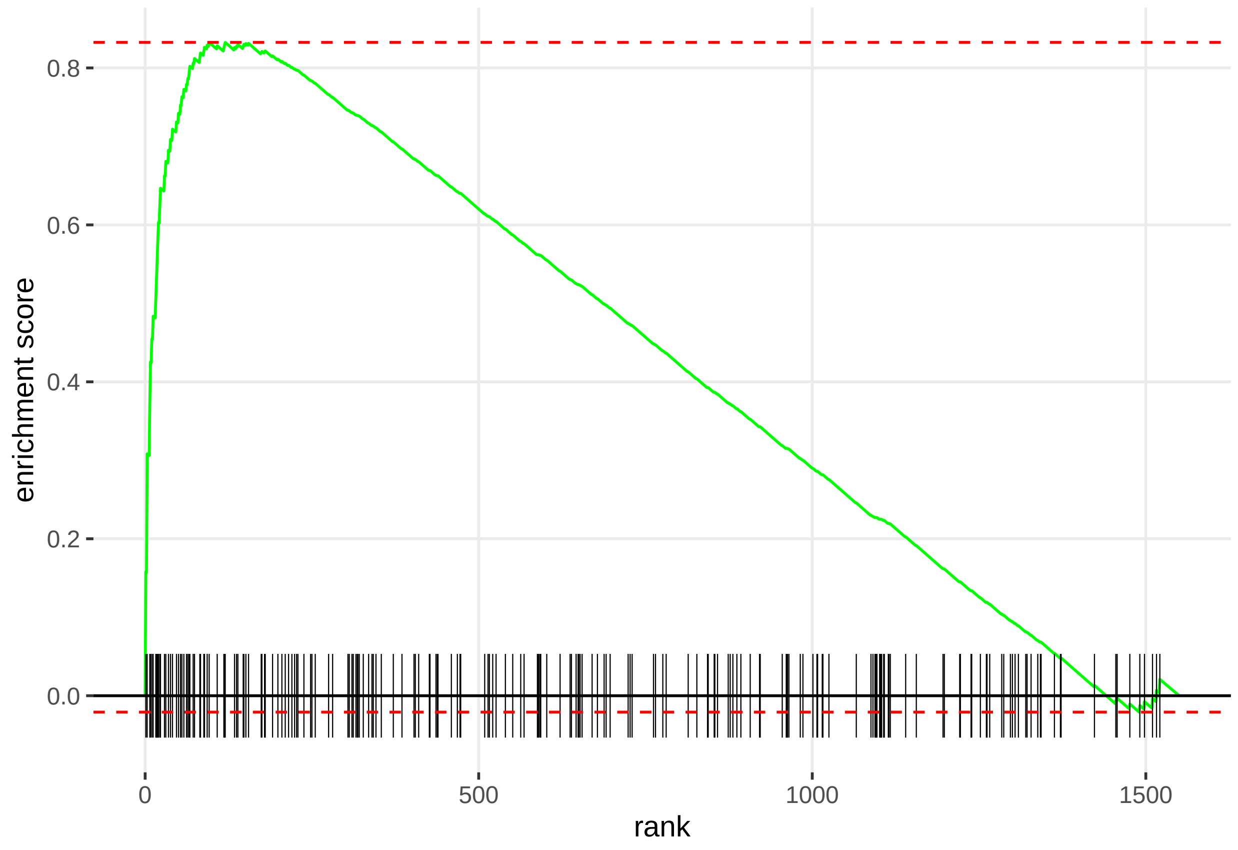
```
# Print the GSEA results to examine the outputs
print(gsea_results_up)
```

```
##           pathway       pval       padj        ES      NES nMoreExtreme  size
##            <char>      <num>      <num>     <num>    <num>        <num> <int>
## 1: Upreprogramome 0.00039996 0.00039996 0.8325559 1.728331            3   208
##     leadingEdge
##          <list>
## 1: POU5F1, ....
```

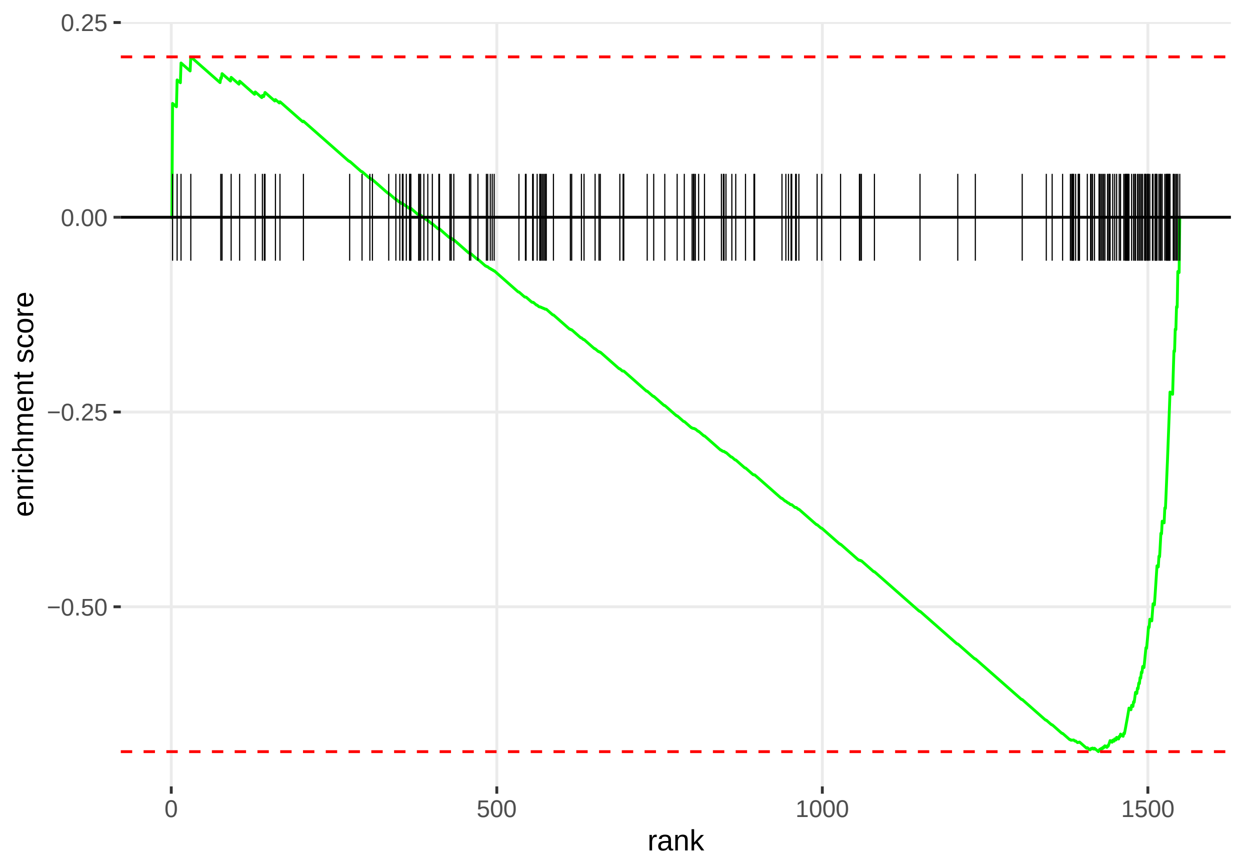
```
print(gsea_results_down)
```

```
##             pathway      pval      padj         ES       NES nMoreExtreme  size
##              <char>     <num>     <num>      <num>     <num>        <num> <int>
## 1: Downreprogramome 0.0009999 0.0009999 -0.6859845 -2.075416            9   190
##     leadingEdge
##          <list>
## 1: NR2F2, E....
```

```
# Visualization of the most significant GSEA results
# Ensure the pathway is selected correctly

# Since 'most_significant_up$pathway' and 'most_significant_down$pathway' return the names directly:
plotEnrichment(pathway = gene_sets[["Upreprogramome"]], ranked_list)
```



```r
# Add a title manually using the title() function in base R


plotEnrichment(pathway = gene_sets[["Downreprogramome"]], ranked_list)
```



```
# Add a title manually using the title() function in base R
```

These plots extend over my transcription factors ranked from left to right by most upregulated to most downregulated.

The first of these two enrichment plots is for the upreprogramome across my TFs, and the second is for the downreprogramome.

From viewing the summary of the GSEA objects, the leadingEdge gene that contributes most to the enrichment score of the upreprogramome before it peaks is POU5F1, which drove the gastrulation GO term in the original DGE analysis. It plays a key and well characterized role in embryonic development and stem cell pluripotency. The leadingEdge gene that contributed most to the enrichment score of the downreprogramome before it peaked was NR2F2. NR2F2 has widely-varying purposes, but importantly has been implicated in cell differentiation regulation. Its downregulation driving a downregulome enrichment stands to reason, then.

And overall, these significance values for the enrichment of the upreprogramome in upregulated genes and the downreprogramome in downregulated genes are very low, well below, the significance cutoff, and even lower for the upreprogramome.

Methods:

Conditions

The RNA sequencing data used in this analysis was derived from human fiborblast cells (the BJ line) at 48 hours post-seeding with no OSKM treatment (control), and the same type of fibroblast 96 hours after seeding and OSKM transduction. I found the samples from the Kejin Hu Nature paper that I have referenced so frequently.

Whole Process Overview

Fastq files were downloaded using prefetch from the SRA database and processed with fasterq-dump. Quality control was performed using FastQC and MultiQC, followed by alignment using STAR with default parameters and subsequent quality checks with SAMtools and QoRTs. The library was stranded and the reads were paired-end. The data being BGI, I provided FastQC with custom BGI adapter sequences I found online in .txt file, which was necessary given that FastQC by default seeks out Illumina sequencers. I chose QoRTs after repeatedly sustaining intractable error message with RSeQC. A high duplicate rate was initially concerning, but ~26M aligned read pairs per sample greatly ameliorated this issue. The aligned reads were quantified using featureCounts, and differential expression analysis was conducted with DESeq2. I used clusterProfiler and Revigo for many of my GO visualization, and the fgsea library for the gene set enrichment analysis that I concluded with. Throughout the analysis, reproducibility and data integrity were prioritized, as all bash scripts were version-controlled and shared on GitHub.

Discussion:

Couldn't evaluate reprogramming legitimacy

As aforementioned, my greatest and most higher level difficulty was not being able to evaluate reprogramming legitimacy for TFs on account of the hESC cell line from the same experiment having two few technical replicates. And again, my solution was load a table from the reprogramming legitimacy paper that includes information that served as the basis for my constructing the gene sets of the "upreprogramome" and "downreprogramome" (see introduction). They compared human fibroblasts and human embryonic stem cells TF counts to establish each ones' relative patterning. And while I wasn't comfortable using their hESC data in direct comparison with mine (to establish whether or not any of my TF gene counts appear iPSC-like and had been "legitimately reprogrammed") given that it was generated with subtle differences in QC and processing from mine, I was comfortable extracting the list of TFs that they assigned to the upreprogamome and downreprogamome to use as gene sets in GSEA for my control versus OSKM comparison.

Normalization questions for TF subsetting

As far as real technical quandaries are concerned, I wasn't certain initially if I should recompute size factors for my gene set after subsetting for just transcription factors, or keep the TF counts normalization relative to the whole transcriptome. Following Merv's advice, I wanted to evaluate how dramatically the size factors would change if I recomputed post-subsetting, as well as whether or not any outlier high expression value TFs among a fairly small total TF count sample size could mess up the normalization.

I proceeded testing this below:

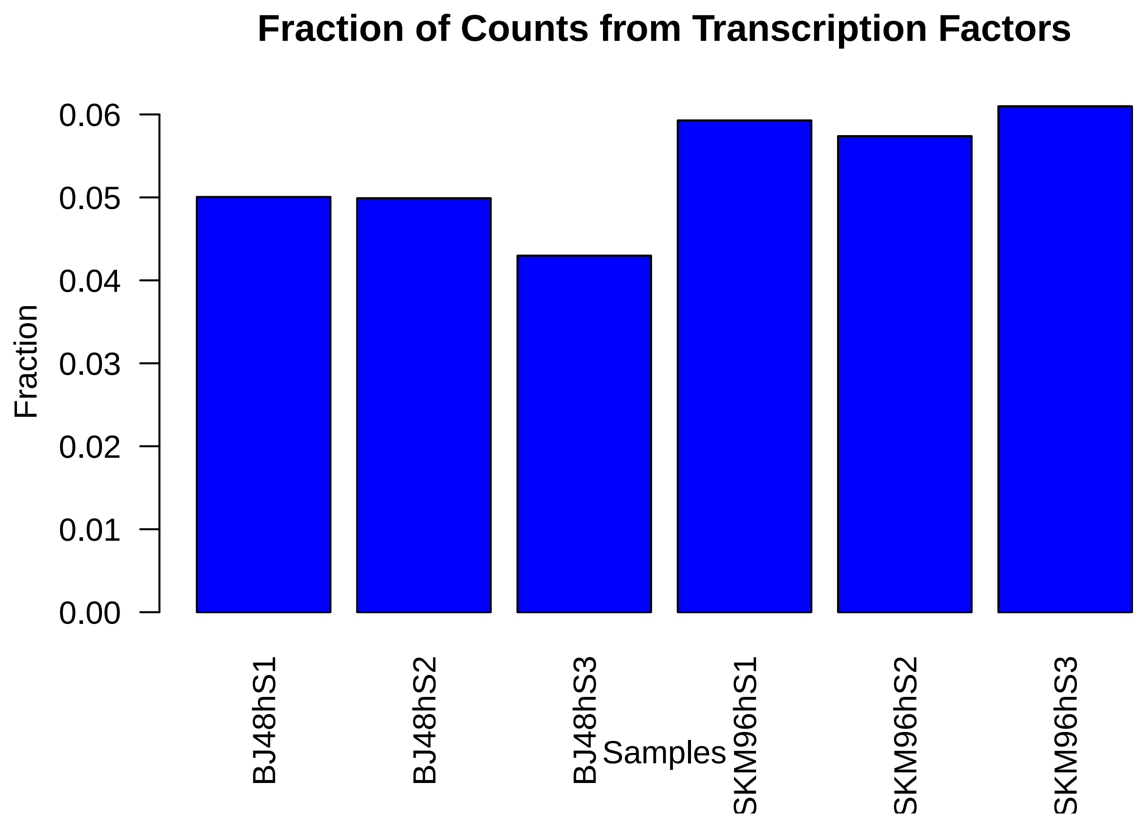
This plot, besides the ultimate hubris of not subsetting directly from a results object of dds_yamanaka, is useful. Knowing how much TFs contribute to overall expression can give us a sense of whether or not focusing exclusively on them would require specialized processing, and hwo much expression stability can be expected from them.

```r
# Calculate total counts per sample in the original dataset
total_counts_all_genes <- colSums(counts(dds_yamanaka))

# Calculate total counts per sample for just transcription factors
```

```r
total_counts_tfs <- colSums(counts(tf_dds))

# Calculate the fraction of counts from TFs
fraction_tfs <- total_counts_tfs / total_counts_all_genes

# Print the fraction of counts from TFs
print(fraction_tfs)
```

```
##     BJ48hS1     BJ48hS2     BJ48hS3 BJOSKM96hS1 BJOSKM96hS2 BJOSKM96hS3
##  0.05005134  0.04990567  0.04297827  0.05927973  0.05738511  0.06098927
```

```r
# Create a barplot of the fraction of counts from TFs
barplot(fraction_tfs,
        main = "Fraction of Counts from Transcription Factors",
        ylab = "Fraction",
        xlab = "Samples",
        col = "blue",
        las = 2)  # Adjust label orientation if necessary
```



The fraction of all counts per sample that are coming from human transcriptions factors ranges from 4.99% to 6.09%. This is, self-evidently, quite a small fraction of the whole gene set.

Following Merv's advice from Piazza, I want to evaluate how my size factors change from the original ones for the entire gene set, versus recomputed ones for just my 1,549 human transcription factors.

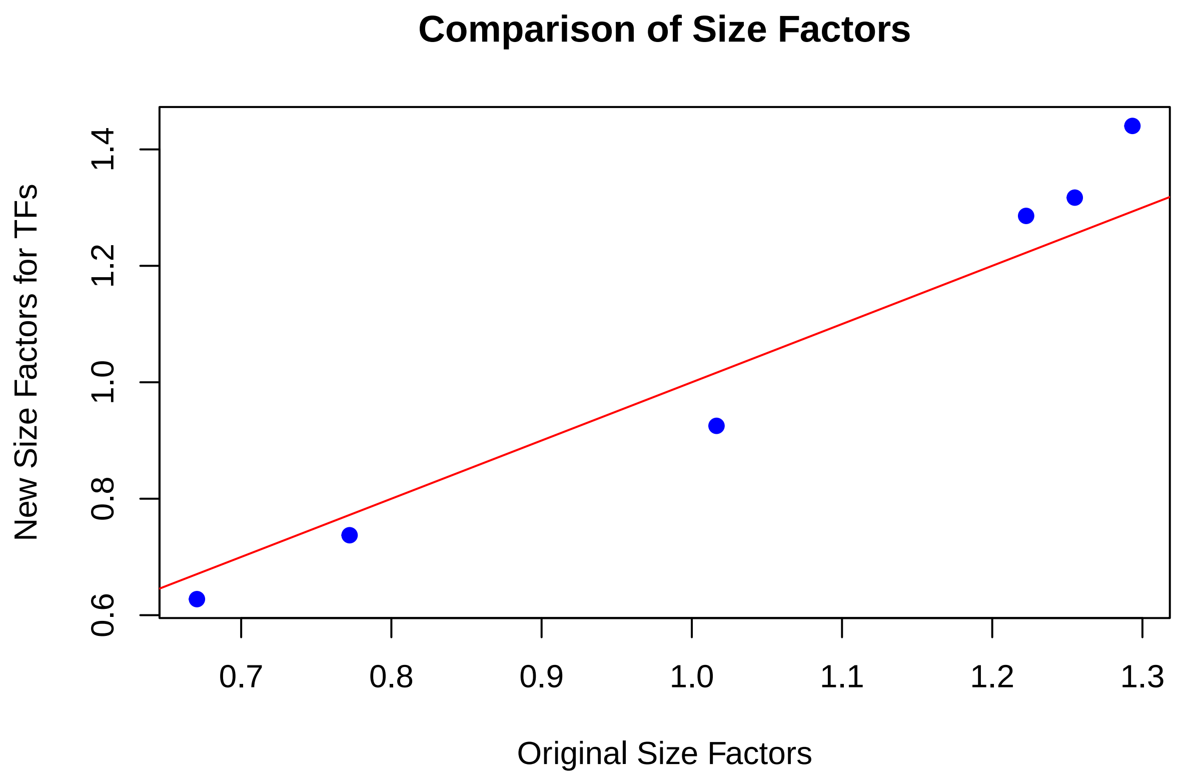
```r
# Create a copy of the original DESeq2 object
tf_dds_copy <- tf_dds

# Recalculate size factors for just the transcription factors
tf_dds_copy <- estimateSizeFactors(tf_dds_copy)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```r
sizeFactors(tf_dds_copy)
```

```
##     BJ48hS1     BJ48hS2     BJ48hS3 BJOSKM96hS1 BJOSKM96hS2 BJOSKM96hS3
##   0.6275458   0.7373130   0.9251530   1.3171609   1.2857707   1.4402985
```

```r
# Extract size factors
original_size_factors <- sizeFactors(tf_dds)
new_size_factors <- sizeFactors(tf_dds_copy)

# Create a plot for comparison
plot(original_size_factors, new_size_factors,
     xlab = "Original Size Factors",
     ylab = "New Size Factors for TFs",
     main = "Comparison of Size Factors",
     pch = 19, col = "blue")
abline(0, 1, col = "red")  # Add a y=x line to aid in visual comparison
```



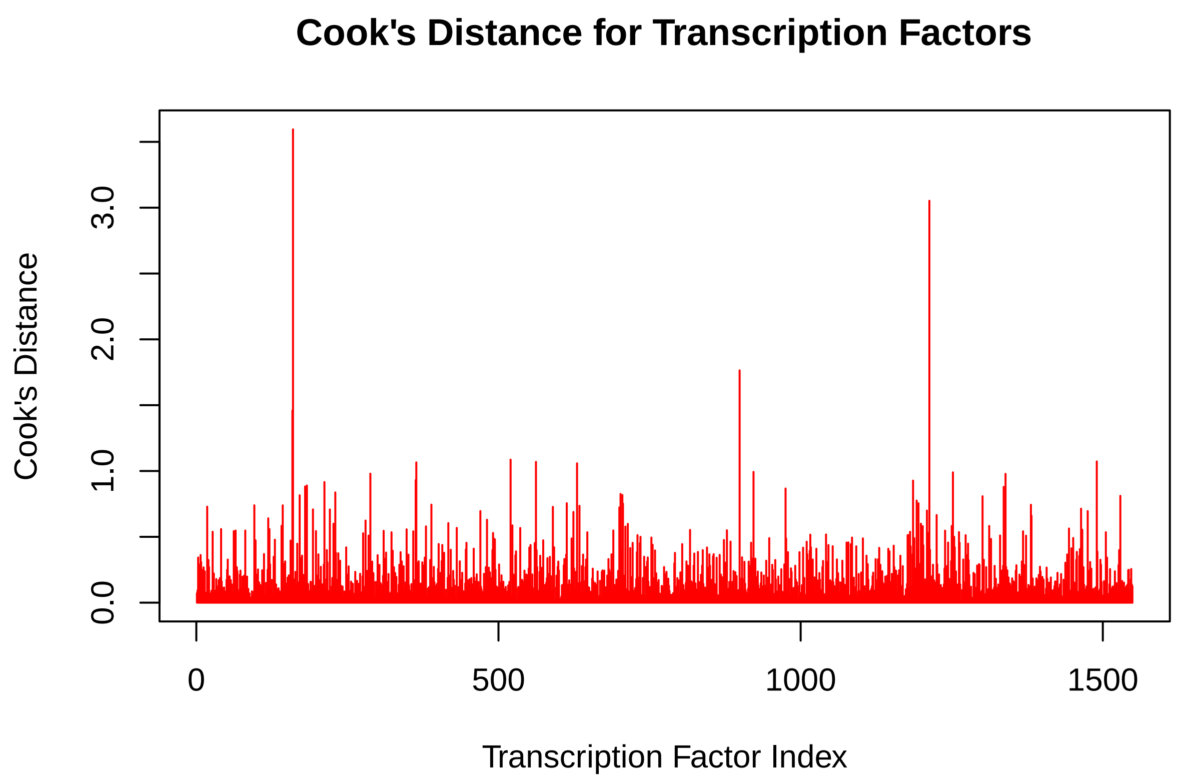
The $x = y$ line here represents where points would fall if we observed no size factor change. The first three (reading from left to right) dots represent my pre-Yamanaka samples, and the last three represent my post-Yamanaka samples. The pre-yamanaka size factors decrease, as compared to the original, and the post-ones increase.

My concern is that a small number of TFs, expressed highly enough, could make everything else look down-regulated unfairly. So I use Cook's distance, which measures the influence of each TF gene on the computed size factors, to see if I have any serious outliers shifting the balance of my normalization.

```
# Calculate Cook's distances for each gene
dds_fit <- DESeq(tf_dds_copy, fitType="mean")
```

```
## using pre-existing size factors

## estimating dispersions

## found already estimated dispersions, replacing these

## gene-wise dispersion estimates

## mean-dispersion relationship

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
## final dispersion estimates

## fitting model and testing
```

```
cooks_distances <- assays(dds_fit)[["cooks"]]
mean_cooks <- rowMeans(cooks_distances, na.rm = TRUE)

# Plot Cook's distances to identify highly influential genes
plot(mean_cooks, type='h',
     main="Cook's Distance for Transcription Factors",
     xlab="Transcription Factor Index", ylab="Cook's Distance",
     col="red")
```



Purely visually, we can see that two to three different transcription factor genes have vastly higher Cook's Distance scores than the rest.

```
# Determine the 99th percentile threshold for Cook's distance
threshold_cooks <- quantile(mean_cooks, 0.99)

# Identify TFs whose Cook's distance exceeds the 95th percentile
high_influence_tfs <- names(mean_cooks)[mean_cooks > threshold_cooks]

# Report Cook's distances for these TFs
high_influence_cooks <- mean_cooks[high_influence_tfs]

# Create a named vector to display results more clearly
```

```
results <- setNames(high_influence_cooks, high_influence_tfs)

# Print the results
print(results)
```

```
## ENSG00000152284 ENSG00000168874 ENSG00000079263 ENSG00000250312 ENSG00000113196
##       1.4587939       3.5951188       0.9162087       0.9801568       0.9310597
## ENSG00000039600 ENSG00000128573 ENSG00000185697 ENSG00000165244 ENSG00000170345
##       1.0654001       1.0865443       1.0696070       1.0585645       1.7643680
## ENSG00000188779 ENSG00000197124 ENSG00000198597 ENSG00000188227 ENSG00000213793
##       0.9931610       0.9272831       3.0527208       0.9898413       0.9782261
## ENSG00000160224
##       1.0727197
```

Setting a 99th percentile cutoff, we see that, even among the top 1th percentile of influential TF genes, we are seeing one TF (ENSG00000168874/ATOH8), that contributes almost 4 times more to size factor calculation than the bottom of this quantile.

I am deciding that this is enough evidence for not redoing normalization with size factors specific to just the TF genes. I will adjust my p-values in my transcription factor DESeq2 object to just reflect the sample size of the TF gene counts, however.

Multiple testing method choice:

Knowing that - among TFs - the differential expression normalization assumption that most genes are not differentially expressed between conditions isn't always true with TFs, I also wanted to choose a stringent p value adjustment method. BH - which, from the literature, seems to be something of a default with RNA-seq data, was finding approximately 50% of all of my TFs to be significantly differentially expressed. Given OSKM iPSC induction takes two to three weeks, that was much higher than my prior expectation, so I ended up going with the Holm-Bonferroni correction from some online research.

Even with the relative stringency of the Holm-Bonferroni correction, we are still observing 17.7% of our TFs as being significantly differentially expressed.

Choosing BP for my GO anlysis:

From some online research, BP was volunteered as best covering most of the functional categories I would be looking for among TFs after OSKM induction.

Data Summary Table:

Dataset Description	Source	Key Characteristics and Findings
Raw sequencing data	SRA Database	Human fibroblast cells (BJ line), control and OSKM-treated samples at different time points post-seeding.
Quality control checks	FastQC, MultiQC	Custom BGI adapter sequences used due to BGI sequencing platform; high duplicate rate noted but mitigated by high aligned read pairs.

Dataset Description	Source	Key Characteristics and Findings
Alignment and quantification of reads	STAR, featureCounts	Paired-end reads, stranded library; reads aligned to human hg38 reference genome, quantified using featureCounts.
Differential expression analysis	DESeq2	Analysis conducted with three replicates per condition; adjustment for multiple comparisons using Holm-Bonferroni method.
Gene Ontology (GO) enrichment analysis	CP, Revigo	Used to understand biological processes promoted or inhibited by Yamanaka factors.
Gene Set Enrichment Analysis (GSEA)	fgsea	Identification of significantly upregulated (upreprogramome) and downregulated (downreprogramome) gene sets in relation to Yamanaka factor induction.
Venn diagrams	VD	Visualization of overlap between up/down-reprogramome and significantly regulated genes in control/OSKM data.
Principal Component Analysis (PCA)	DESeq2 `plotPCA`	Clear separation between conditions observed; significant variation attributed to treatment.
Enhanced Volcano plots	Volcano	Visualization of differential gene expression, highlighting significant changes post-OSKM treatment.
Cook's distance assessment	DESeq2 `cooks`	Influential gene detection to evaluate impact on normalization; one gene (ATOH8) notably influential.

Bibliography:

Cevallos, Ricardo R., et al. "Human Transcription Factors Responsive to Initial Reprogramming Predominantly Undergo Legitimate Reprogramming during Fibroblast Conversion to Ipscs." Nature News, Nature Publishing Group, 12 Nov. 2020, www.nature.com/articles/s41598-020-76705-y.

Hochedlinger, Konrad. "A Molecular Roadmap of Reprogramming Somatic Cells into iPS Cells." Cell, www.cell.com/cell/pdf/S0092-8674(20)30610-3.pdf?_returnURL=https:/linkinghub.elsevier.com/retrieve/pii/S0092867420306103?showall=true. Accessed 23 Apr. 2024.

Lambert, Samuel A. Cell, Cell, www.cell.com/cell/pdf/S0092-8674(20)30610-3.pdf?_returnURL=https:/linkinghub.elsevier.com/retrieve/pii/S0092867420306103?showall=true. Accessed 23 Apr. 2024.

Mikkelsen, Tarjei S., et al. "Dissecting Direct Reprogramming through Integrative Genomic Analysis." Nature News, Nature Publishing Group, 28 May 2008, www.nature.com/articles/nature07056.