

---

README\_loc.md

# Perform differential expression analysis on fibrotic and non-fibrotic patients under 4 different treatments on HPC

---

## Author

---

Sam (Cheng-Hsiang) Lu

Email: [Cheng-Hsiang.Lu@cshs.org](mailto:Cheng-Hsiang.Lu@cshs.org)

## Mentor

---

David Casero

Email: [David.Casero@cshs.org](mailto:David.Casero@cshs.org)

## Background

---

### Inflammatory bowel diseases(IBD)

Inflammatory bowel diseases (IBD) are a group of chronic conditions that cause inflammation and damage to the digestive tract. The two main types of IBD are Crohn's Disease (CD) and Ulcerative Colitis (UC). Both Crohn's disease and ulcerative colitis are chronic conditions, meaning they can last for a lifetime and require ongoing treatment to manage symptoms and prevent complications.

UC affects only the colon and rectum and causes symptoms such as bloody diarrhea, abdominal pain, and a frequent need to pass stools. It can also lead to complications such as inflammation of the skin, eyes, and joints. On the other hand, CD can affect any part of the digestive tract, like small or large intestine, and can cause symptoms such as abdominal pain, diarrhea, weight loss, and fatigue. It can also cause complications such as fistulas (abnormal connections between different parts of the intestine)

One of the most prevalent complication of CD is the onset of fibrotic complications and strictures (narrowing of the intestine)[1]. The molecular mechanisms involved in these phenotypes remain largely unknown, and as a result, there are currently no effective drugs to prevent or treat stricturing CD.

## Induced Pluripotent Stem Cells(iPSCs)

Induced pluripotent stem cells (iPSCs) are a type of stem cell that are generated in the laboratory by reprogramming adult cells, such as skin or blood cells, to a pluripotent state. A pluripotent state means that the cells have the potential to develop into any type of cell in the body, just like embryonic stem cells.

iPSCs offer several advantages as they can be generated from the patient's own cells, avoiding issues with immune rejection, ethical concerns and the need for embryos.

iPSCs can be differentiated into multiple cell and tissue types, and can therefore be used to study the underlying causes of diseases, test new drugs and therapies, and potentially generate replacement tissues or organs for transplantation. As iPSCs possess the same genetic background as the patient they are derived from, they are considered an instrumental tool in the field of personalized and precision medicine[2].

## Aim

---

In this project, we aim to take advantage of iPSC lines to unveil specific signaling pathways specifically affected in patients with fibrotic CD[3]. We will be analyzing RNA-seq data from 19 iPSC lines that were differentiated into gut mesenchymal organoids. This panel comprises 10 iPSC lines derived from Crohn's disease patients that suffered fibrotic complications, and 9 lines from patients with non-fibrotic disease. Each iPSC line was differentiated into mesenchymal organoids in two independent replicates, and each was subjected to 4 different treatments:

- untreated
- TGF $\beta$  (a pro-fibrotic cytokine)[4]
- TNF $\alpha$  (a pro-inflammatory cytokine)

- and the combination of TGF-b+TNF-a

The final RNA-Seq dataset comprised a total of 151 samples(1 library failed). The objective is to

- investigate if the effect of the four different treatments in iPSC-derived organoids recapitulate the expected responses observed in-vivo[5].
- perform differential expression analysis to identify genes that show differential responses between fibrotic and non-fibrotic patients[6].

## Pipelines

---

### In HPC

#### Convert 151 Fastq to Fasta files

First, put all fasq.gz files into one folder and list all fastq files' name in fastqfiles.txt.

```
ls *q.gz > fastqfiles.txt
```

Cut redundant suffix "\_R1\_trimmed" and list all fastq files' name in libraryname.txt and preffix.txt.

```
ls *q.gz | cut -f 1 -d '.' | sed 's/_R1_trimmed//g' >libraryname.txt
```

```
ls *q.gz | cut -f 1 -d '.' | sed 's/_R1_trimmed//g' > preffix.txt
```

Form a table with 3 columns: fastqfiles.txt libraryname.txt preffix.txt.

```
paste fastqfiles.txt libraryname.txt preffix.txt > tofastatable.txt
```

Create small-sized fasta-formatted files. To submit this job to the cluster on HPC, you need to read the file, library, and prefix. Once you have done that, run the script "generatfastaFromFastaqz" which combine the script "DCfastaqTofastaLibraryId.pl". This results in small-sized fasta-formatted files contain only one header and one sequence per read. You can find all scripts in the "scripts" folder.

```
cat tofastatable.txt | awk '{print}' | while read file library preffix ; do
qsub -cwd -o $PWD -e $PWD -l h_data=2048M,h_rt=8:00:00
$HOME/scripts/generatefastaFromFastaqz $file $library $preffix

done
```

Figure 1 shows what each fasta-formatted file look like.

```
>527iP29TNFaM_S124_1
GNAGCAAAGTGGTACCCAAACCTAAGAGCTATTATCCTAAATTCAAAATCTAAAAAAAAACCTTAGAACCTC
>527iP29TNFaM_S124_2
GNGCTCATGCTGTTTCCAGGAGAAAAGTAAGATCCTCAGCCGTATTCGCTTAATATTCATTCTAA
>527iP29TNFaM_S124_3
CNTACCTTGAGTCATTTTTCTATTCTATGCCATAACTAAATTCTGATTAGTTCTCCAATACAATGC
>527iP29TNFaM_S124_4
TNAGTATTGATTGTTAGCGGTGTGGTCGGGTGTGTTATTCTGAATTGGGGAGGTTATGGGTTAATAG
>527iP29TNFaM_S124_5
GNCTAACTAAAGAAGGAAAATGTAGAATTAGGCAGAAATCTCATGAAATCCTCCATGCAATAAGGAGGCTT
>527iP29TNFaM_S124_6
CNCTTAATTGACTAAAGGATGGTAGGTGGTCACATGCAGTCATGTGGATTCTAACATGACATTAGTGAGT
>527iP29TNFaM_S124_7
ANTCTCCTTTGTTTGCAATTGAATAAGGTAAATTCAAGGCTGTGCAGCTTCATTGCCGTTGGTGGTT
>527iP29TNFaM_S124_8
GCCATCCTTCGATTCTCAGGTGTCAGGCATTCTGGTCATTGGTATTGTGATCTTATTGGTCCCTGTAC
>527iP29TNFaM_S124_9
GCCGAAAAACATCAGAGATGGAGGGCCCCAGCAGCAGGAAGACGTCAATGATGCCAGTCTGACATCCAGCGAAC
>527iP29TNFaM_S124_10
GCCGTGAATGCAGGACCATCCAGGTCTCAAAGTCTGTGAGGTTGTTCATATCCAAACAAGGGCCCTGCTGGC
```

### Generate auxiliary files and directories for each sample

Put a list of names of all fasta files in the directory and save them in a text file named "fastafiles.txt".

```
ls *fasta.gz > fastafiles.txt
```

Cut the redundant suffix ".fasta.gz" from the names of all fasta files and generate a new list of file names with the suffix removed in a text file named "targetdirectories.GTF.txt".

```
cat fastafiles.txt | sed 's/.fasta.gz//g' > targetdirectories.GTF.txt
```

Create a separate directory for each sample listed in "fastafiles.txt".

```
cat fastafiles.txt |while read line ; do mkdir ${line}\\.fasta\\.gz/GTFpass1/ ; done
```

### Form the submission script called "sendmyof"

Add a shebang line at the beginning of your script file named "sendmyof" to indicate the interpreter that should be used to execute the script.

```
echo '#!/bin/bash/' > sendmyof
```

The command below runs the "generatesendscriptSingleGTFParam" script with several input parameters to map the RNA-seq data with STAR. The input parameters include the list of target directories containing the input data ("targetdirectories.GTF.txt"), the directory prefix for pass-1 alignments ("GTFpass1"), a parameter file containing settings for STAR alignment ("Parameters.txt"), a prefix for individual submission scripts to HPC ("myof"), the path to the STAR index directory ("~/home/luc/RNASEQ\_MASTER/Hsapiens/GRC38/INDEXES/GRCh38.primary.33.basicselected.STAR2.7.3a/"), the path to the input data directory ("~/home/luc/iPSC/MYOFIBROBLAST/"), the amount of free memory to use ("mem\_free=32G"), and the number of threads to use ("8"). In the end, it will generate a sample-specific sumission script called "processLaneSingleGTFParam" in each sample's folder:

```
./generatesendscriptSingleGTFParam targetdirectories.GTF.txt GTFpass1  
Parameters.txt myof  
~/home/luc/RNASEQ_MASTER/Hsapiens/GRC38/INDEXES/GRCh38.primary.33.basicselected  
~/home/luc/iPSC/MYOFIBROBLAST/ mem_free=32G 8 >> sendmyof
```

Change sendmyof into executable mode and run sendmyof.

```
chmod a+x sendmyof
```

- sendmyof

It will take less than one day to run through 151 samples and generate each sample a folder which contain every output from STAR.

### Create a table summarizing the mapping statistics for each sample

Change directory into one sample file which ends with "GTFpass1". Extract the first column from the mapping statistics file and store it in "temp2.txt".

```
grep "|" 008iP22TGFbM_S71GTFpass1/008iP22TGFbM_S71GTFpass1Log.final.out |  
cut -f 1 -d "|" | sed 's/^ *//g' | awk 'NR>3 {print}' > temp2.txt
```

The first column from the mapping statistics file in Figure 2.

```

Mapping speed, Million of reads per hour
Number of input reads
Average input read length
Uniquely mapped reads number
Uniquely mapped reads %
Average mapped length
Number of splices: Total
Number of splices: Annotated (sjdb)
Number of splices: GT/AG
Number of splices: GC/AG
Number of splices: AT/AC
Number of splices: Non-canonical
Mismatch rate per base, %
Deletion rate per base
Deletion average length
Insertion rate per base
Insertion average length
Number of reads mapped to multiple loci
% of reads mapped to multiple loci
Number of reads mapped to too many loci
% of reads mapped to too many loci
Number of reads unmapped: too many mismatches
% of reads unmapped: too many mismatches
Number of reads unmapped: too short
% of reads unmapped: too short
Number of reads unmapped: other
% of reads unmapped: other
Number of chimeric reads
% of chimeric reads

```

Create an empty temporary file for storing intermediate results.

```

rm tempprev.txt
touch tempprev.txt

```

Extract the total mapped reads from each subsequent mapping statistics file and combine with previous results.

```

ls *pass1/*final.out | while read line ; do
grep "|" $line | cut -f 2 > temp.txt
paste tempprev.txt temp.txt > tempnew.txt
mv tempnew.txt tempprev.txt

```

done

Remove the first column and write the final results to a file called "mappingstatsFirstpass.txt".

```
cut -f 2- tempprev.txt | awk 'NR>3 {print}' > tempnew.txt
mv tempnew.txt tempprev.txt
paste temp2.txt tempprev.txt > mappingstatsFirstpass.txt
```

Figure 3 shows what mappingstatsFirstpass.txt look like.

Mapping speed, Million of reads per hour	452.92	344.45	460.17	487.96	331.34	441.99	560.78	487.93	532.33	351.69
Number of input reads	46046803	40855908	43588067	47982561	39392754	33886137	72122988			
Average input read length	75	75	75	75	75	75	75	75	75	75
Uniquely mapped reads number	42647348	36668193	39982295	43570276	36214207	31289999				
Uniquely mapped reads %	92.62%	89.75%	91.73%	90.80%	91.93%	92.34%	91.60%	90.47%	93.77%	90.29%
Average mapped length	75.21	75.21	75.21	75.20	75.20	75.19	75.21	75.20	75.20	75.24
Number of splices: Total	14107761	11743321	12025357	12627415	11851597	10196588				
Number of splices: Annotated (sjdb)	14029840	11673663	11945482	12536747	11787471	10140891				
Number of splices: GT/AG	14015583	11657575	11928403	12523568	11777228	10127320				
Number of splices: GC/AG	78050	74078	84368	87757	62434	59661	145139	96183	139381	88755
Number of splices: AT/AC	7048	6261	6785	8291	6179	5252	12249	9152	15682	8499
Number of splices: Non-canonical	7080	5407	5801	7799	5756	4355	9871	8901	10933	5153
Mismatch rate per base, %	0.27%	0.28%	0.29%	0.28%	0.28%	0.28%	0.31%	0.28%	0.29%	0.29%
Deletion rate per base	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%
Deletion average length	1.47	1.47	1.47	1.48	1.48	1.47	1.46	1.47	1.48	1.50
Insertion rate per base	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%
Insertion average length	1.61	1.52	1.43	1.43	1.63	1.60	1.47	1.43	1.62	1.42
Number of reads mapped to multiple loci	2398878	3287402	2498197	3093230	2029231	1702575	4120922	3568843	3990452	2580456
% of reads mapped to multiple loci	5.21%	8.05%	5.73%	6.45%	5.15%	5.02%	5.71%	6.70%	4.80%	5.64%
Number of reads mapped to too many loci	222693	207795	314715	343251	343822	264218	498200	397171	262921	547542
% of reads mapped to too many loci	0.48%	0.51%	0.72%	0.72%	0.87%	0.78%	0.69%	0.75%	0.32%	1.20%
Number of reads unmapped: too many mismatches	1269	1089	1322	1446	1105	1001	2452	1565	1799	1824
% of reads unmapped: too many mismatches	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Number of reads unmapped: too short	759644	674719	771365	950053	788663	614921	1410179	1081555	902701	1286763
% of reads unmapped: too short	1.65%	1.65%	1.77%	1.98%	2.00%	1.81%	1.96%	2.03%	1.09%	2.81%
Number of reads unmapped: other	16971	16710	20173	24305	15726	13423	29546	25946	17237	24268
% of reads unmapped: other	0.04%	0.04%	0.05%	0.05%	0.04%	0.04%	0.04%	0.05%	0.02%	0.05%
Number of chimeric reads	0	0	0	0	0	0	0	0	0	0
% of chimeric reads	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

The summary statistics show good rates of unique alignments for all samples.

## Counts

Generate a directory called "COUNTS" and copy all gene count files to this folder and then clean all file names.

```
mkdir COUNTS
cp *pass1/*PerGene* COUNTS/
```

```
ls *tab|while read line ; do mv $line ${line/GTFpass1ReadsPerGene.out/} ;  
done
```

For each count file, extract and create the five count tables.

```
ls *.tab | while read line ; do  
echo $line  
cat $line | awk 'NR==3{print}' | cut -f 2- > ${line/tab/nofeature\.tab}  
cat $line | awk 'NR==4{print}' | cut -f 2- > ${line/tab/ambiguous\.tab}  
cat $line | awk 'NR>4{print}' | cut -f 2 > ${line/tab/nostrand\.tab}  
cat $line | awk 'NR>4{print}' | cut -f 3 > ${line/tab/sense\.tab}  
cat $line | awk 'NR>4{print}' | cut -f 4 > ${line/tab/antisense\.tab}  
done
```

Make a Geneid list from one of the count tables as "countsannot\_GRCh38.primary.Selected.Geneid.txt".

```
ls 008iP22TGFbM_S71.tab | head -1 | while read line; do  
cut -f 1 $line | awk 'NR>4{print}' >  
countsannot_GRCh38.primary.Selected.Geneid.txt  
done
```

Create a file listing the names of all samples as "RBarretTNFATGFBsamples.txt".

```
ls *.sense.tab | sed 's/.sense.tab//g' | tr -s " " "\n" | sed 's/_1//g' >  
RBarretTNFATGFBsamples.txt
```

Make count tables for sense, anti-sense, nostrand, ambiguous, and nofeature reads.

```
# combine all sense counts into RBarretTNFATGFB_sense.ALL.cnt  
paste *.sense.tab > RBarretTNFATGFB_sense.ALL.cnt  
  
# combine all antisense counts into RBarretTNFATGFB_antisense.ALL.cnt  
paste *.antisense.tab > RBarretTNFATGFB_antisense.ALL.cnt  
  
# combine all nostrand counts into RBarretTNFATGFB_nostrand.ALL.cnt  
paste *.nostrand.tab > RBarretTNFATGFB_nostrand.ALL.cnt
```

```
# combine all ambiguous counts into RBarretTNFATGFB_ambiguous.cnt
cat *ambiguous.tab > RBarretTNFATGFB_ambiguous.cnt

# combine all nofeature counts into RBarretTNFATGFB_nofeature.cnt
cat *nofeature.tab > RBarretTNFATGFB_nofeature.cnt
```

Next, I am going to use "RBarretTNFATGFB\_antisense.ALL.cnt" file for the further analysis, as this matrix contains the counts matching the strand-specificity of the RNA-Seq libraries generated in this study.

## In MATLAB

### Transfer data and import annotation

Transfer the counts, annotation, and mappability data to your local laptop.

```
RBarretTNFATGFCnt = textread('RBarretTNFATGFB_antisense.ALL.cnt','');
RBarretsamplesTNFATGFB = textread('RBarretTNFATGFBsamples.txt','%s');
RBarretsampleskeysTNFATGFB = textread('samplekeys_Sam.txt','%s');

% calculate the sum of the counts in RBarretTNFATGFCnt, divides the result
% by 1000000, and rounds the result to the nearest integer.
RBarretTNFATGFBmeta_seqdepth=round(sum(RBarretTNFATGFCnt)/1000000);
```

The following files contain the annotation and gene effective lengths (mappabilities) for the human gene annotation used for alignment, and can be found in the "mappability and R code" folder.

```
Gencode_33_Selected_MappSS=textread('mappability and R
code/gencode.v33.Selected.ReadsPerGene.out.MappSS.txt','');
Gencode_33_Selected_MappUS=textread('mappability and R
code/gencode.v33.Selected.ReadsPerGene.out.MappUS.txt','');
Gencode_33_Selected_Geneid=textread('mappability and R
code/gencode.v33.annotation.Selected.geneid.txt','%s\n');
Gencode_33_Selected_Biotype=textread('mappability and R
code/gencode.v33.annotation.Selected.biotype.txt','%s\n');
Gencode_33_Selected_Genename=textread('mappability and R
code/gencode.v33.annotation.Selected.genename.txt','%s\n');
```

## Compile counts

First, initialize a new variable called RBarretTNFATGFBTPM with the same count data as RBarretTNFATGFBCnt. Then, iterates over each gene in the count data matrix. For each gene, the corresponding row in RBarretTNFATGFBTPM is updated by dividing the count data by the gene effective length from the "Gencode\_33\_Selected\_MappSS", multiplying by 1000, and storing the result in RBarretTNFATGFBTPM.

Finally, iterates over each sample in the TPM data matrix. For each sample, the corresponding column in RBarretTNFATGFBTPM is updated by dividing the values in the column by the sum of the values in the column, multiplying by 1,000,000, and storing the result in RBarretTNFATGFBTPM. This step **normalizes the TPM values** across samples and scales the resulting values to TPM.

```
RBarretTNFATGFBTPM = RBarretTNFATGFBCnt;
for i=1:size(RBarretTNFATGFBCnt,1)
    % divid the gene count matrix RBarretTNFATGFBCnt by
    Gencode_33_Selected_MappSS matrix, which is the sum of the transcript
    length of each gene
    RBarretTNFATGFBTPM(i,:) =
    RBarretTNFATGFBCnt(i,:)/Gencode_33_Selected_MappSS(i)*1000;
end
% set any NaN or Inf values resulting from the normalization process to 0
RBarretTNFATGFBTPM(isnan(RBarretTNFATGFBTPM)) = 0;
RBarretTNFATGFBTPM(isinf(RBarretTNFATGFBTPM)) = 0;
for i=1:size(RBarretTNFATGFBTPM,2)
    % scale the TPM values so that the sum of expression values across each
    sample of the matrix is equal to 1,000,000. This ensures that the
    expression values are comparable across different samples and allows
    meaningful comparisons of gene expression levels between different samples.
    RBarretTNFATGFBTPM(:,i) =
    RBarretTNFATGFBTPM(:,i)/sum(RBarretTNFATGFBTPM(:,i))*1000000;
end
```

## Make the first dendrogram

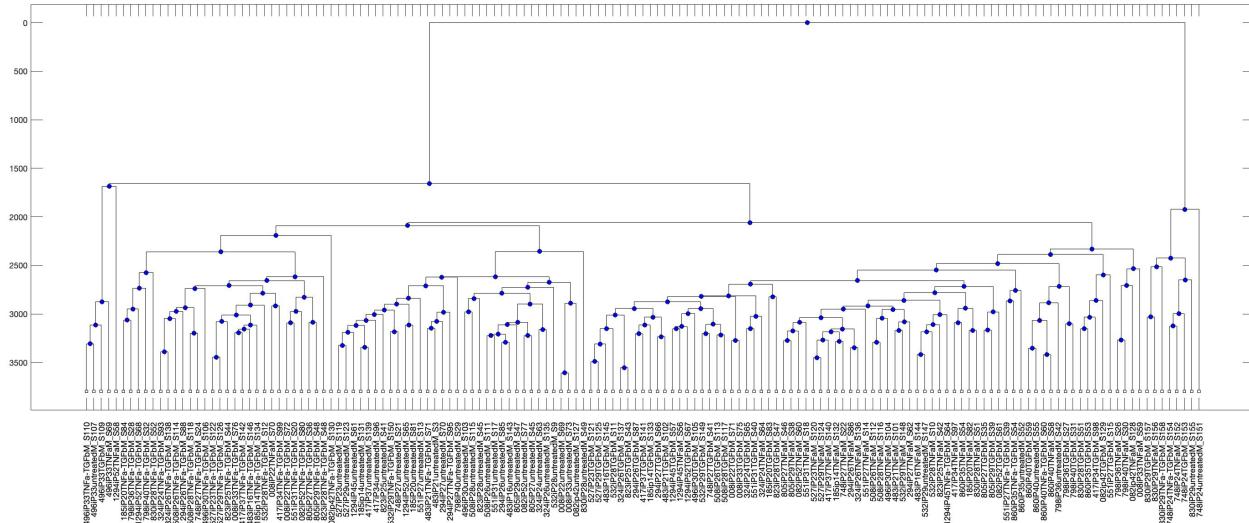
Make a dendrogram to visualize the relationships among samples in the RBarretTNFATGFB dataset based on their gene expression profiles. To downgrade the effect of potential expression outliers in the dendrogram and compute a more robust sample clustering:

1. I generates a random selection of 1,000 genes from the TPM data matrix.

2. Calculates the pairwise distances between the selected genes.
3. Creates a for loop that iterates 9,999 times. For each iteration, a new random selection of 1,000 genes is generated, and the pairwise distances between these genes are added to the previous 'thisdist' calculation.
4. Converts the one-dimensional distance vector 'thisdist' into a distance matrix 'thisdistmat' using the 'squareform' function.
5. Generates a hierarchical clustering tree based on the distance matrix 'thisdistmat'.

Overall, I perform a clustering analysis on a subset of genes in the RBarretTNFATGFB dataset to visualize the relationships among samples based on their gene expression profiles in Figure 4.

```
thisrand = unique(randi([1 size(RBarretTNFATGFBTPM,1)],1,1000));
thisdist = pdist(RBarretTNFATGFBTPM(thisrand,:));
for i=1:9999
    thisrand = unique(randi([1 size(RBarretTNFATGFBTPM,1)],1,1000));
    thisdist = thisdist+pdist(RBarretTNFATGFBTPM(thisrand,:));
end
thisdistmat = squareform(thisdist/10000);
thistree = seqlinkage(thisdistmat,'average', RBarretsamplesTNFATGFB)
plot(thistree, 'ORIENTATION', 'top')
```



A first observation is that the major clusters are formed by samples from the same treatment, with exceptions. Therefore, the Treatment factor seems to be the dominant source of gene expression variation in this experiment.

## All biotypes counts percents

As part of the preliminary quality control, the following estimates the relative contribution of each gene biotype to the expression matrix:

```
% use the unique function and stored the allbiotypes variable.  
allbiotypes = unique(Gencode_33_Selected_Biotype);  
  
% create A cell array allbiotypeslength to store the lengths of each  
biotype name.  
allbiotypeslength = cell(length(allbiotypes),1);  
  
% two new matrices, allbiotypescounts and allbiotypescountspercents, are  
initialized with zeros. These matrices have dimensions (number of unique  
biotypes) x (number of samples in the TPM data). They will be used to store  
the number of reads (counts) and the percentage of total reads (%TPM) for  
each biotype in each sample.  
allbiotypescounts = zeros(length(allbiotypes),size(RBarretTNFATGFBCnt,2));  
allbiotypescountspercents =  
zeros(length(allbiotypes),size(RBarretTNFATGFBCnt,2));  
  
for i=1:length(allbiotypes)  
% finds all the indices of Gencode_33_Selected_Biotype that match the  
current biotype. Then, returned a vector of **indices** where the biotype  
occurs in Gencode_33_Selected_Biotype.  
temp = strmatch(allbiotypes{i}, Gencode_33_Selected_Biotype);  
% Stored the length of the temp vector represents the number of genes with  
the current biotype in the Gencode_33_Selected_Biotype.  
allbiotypeslength{i} = length(temp);  
if length(temp)>1  
% Sum the expression values for all genes with the current biotype across  
all samples. The resulting sums are stored in the corresponding row of the  
allbiotypescounts matrix.  
allbiotypescounts(i,:) = sum(RBarretTNFATGFBCnt(strmatch(allbiotypes{i},  
Gencode_33_Selected_Biotype),:));  
allbiotypescountspercents(i,:) =  
allbiotypescounts(i,:)/sum(RBarretTNFATGFBCnt)*100;  
end  
end  
  
dlmwrite('allbiotypescountspercents.txt',  
allbiotypescountspercents,'delimiter','\t')  
writetable(cell2table(allbiotypes),'allbiotypes.txt','WriteVariableNames',0)
```

Using Excel, create a spreadsheet using "allbiotypes.txt" and "allbiotypescountspercents.txt", and calculate the minimum, maximum, and average values for each biotype in Figure 5. You can access my completed spreadsheet [here](#). Notably, protein\_coding genes exhibit an average of 98.65% among the various biotypes, consistent with my expectations. Moreover, I found no samples with excessive contributions from other biotypes (e.g. mitochondrial and non-coding RNAs), and therefore no library quality issues were found in this step.

Biotypes	min	max	average
IG_C_gene	0.000000000	0.000446580	0.000013340
IG_D_gene	0.000000000	0.000000000	0.000000000
IG_J_gene	0.000000000	0.000013813	0.000000235
IG_V_gene	0.000000000	0.000073058	0.000010604
Mt_rRNA	0.228350000	1.165600000	0.487934371
Mt_tRNA	0.000141280	0.002049800	0.000492376
TEC	0.011838000	0.029033000	0.018976556
TR_C_gene	0.000000000	0.002667100	0.000287637
TR_D_gene	0.000000000	0.000011542	0.000000580
TR_J_gene	0.000000000	0.000076445	0.000009219
TR_V_gene	0.000000000	0.000061956	0.000011904
lncRNA	0.537580000	1.612500000	0.825408079
miRNA	0.002352100	0.014811000	0.004202784
misc_RNA	0.000641580	0.002443400	0.001048973
<b>protein_coding</b>	<b>97.675000000</b>	<b>99.144000000</b>	<b>98.658132450</b>
rRNA	0.000000000	0.000028828	0.000005066
ribozyme	0.000000000	0.000007357	0.000000293
sRNA	0.000000000	0.000004680	0.000000135
scRNA	0.000000000	0.000000000	0.000000000
scaRNA	0.000002616	0.000324330	0.000036518
snRNA	0.000262070	0.001464200	0.000543351
snoRNA	0.001727500	0.004335800	0.002883131
vaultRNA	0.000000000	0.000000000	0.000000000

## Protein coding genes

The "allbiotypes.txt" file contains multiple biotypes. For the next step, I will only retain the "protein\_coding" biotype. I will also remove some gene classes that typically show very noisy or variable gene expression across different samples (e.g histone and ribosomal genes, among others).

```
allbiotypes=unique(Gencode_33_Selected_Biotype);
% finds the 15th unique value, which is protein_coding, of
Gencode_33_Selected_Biotype in the array proteincodingindx.
proteincodingindx = strmatch(allbiotypes{15}, Gencode_33_Selected_Biotype);
biotypeindx = proteincodingindx;

% creates an array additionalgenes contains the indices of genes that have
certain prefixes such as 'MT-', 'H1', 'H2', 'H3', 'H4', 'RPL', or 'RPS' in
their names.
additionalgenes = [strmatch('MT-',Gencode_33_Selected_Genename) ;
strmatch('H1',Gencode_33_Selected_Genename);
strmatch('H2',Gencode_33_Selected_Genename);
strmatch('H3',Gencode_33_Selected_Genename);
strmatch('H4',Gencode_33_Selected_Genename) ;
strmatch('RPL',Gencode_33_Selected_Genename) ;
strmatch('RPS',Gencode_33_Selected_Genename)];

% creates an array nonadditionalgenes with the same length as the
Gencode_33_Selected_Genename array.
nonadditionalgenes = 1:length(Gencode_33_Selected_Genename);
% remove the indices of genes in additionalgenes from the
nonadditionalgenes array.
nonadditionalgenes(additionalgenes) = [];

% mappableindx contains the indices of elements in the
Gencode_33_Selected_MappSS array that are greater than 50.
mappableindx = find(Gencode_33_Selected_MappSS>50);

% a new variable finalIndexGeneric which is the intersection of three other
variables: biotypeindx, nonadditionalgenes, and mappableindx.
finalIndexGeneric =
intersect(biotypeindx,intersect(nonadditionalgenes,mappableindx));
% find the indices of rows in RBarrettTNFATGFBCnt that have a sum greater
than 150 (an average of >1 per sample).
countindx = find(sum(RBarrettTNFATGFBCnt')>150);

% update finalIndexGeneric to be the intersection of finalIndexGeneric and
countindx.
finalIndexGeneric=intersect(finalIndexGeneric,countindx);
```

```
% create a new variable RBarretTNFATGFBCnt_GMask which is a subset of  
RBarretTNFATGFBCnt corresponding to the rows indexed by finalIndexGeneric.  
RBarretTNFATGFBCnt_GMask = RBarretTNFATGFBCnt(finalIndexGeneric,:);  
  
Gencode_33_Selected_Geneid_GMask =  
Gencode_33_Selected_Geneid(finalIndexGeneric);  
Gencode_33_Selected_Genename_GMask =  
Gencode_33_Selected_Genename(finalIndexGeneric);  
Gencode_33_Selected_MappSS_GMask =  
Gencode_33_Selected_MappSS(finalIndexGeneric);  
Gencode_33_Selected_MappUS_GMask =  
Gencode_33_Selected_MappUS(finalIndexGeneric);  
  
% normalize the expression data like we did previously, keeping only the  
filtered set of genes above:  
RBarretTNFATGFBExpression_GMask = RBarretTNFATGFBCnt_GMask;  
for i=1:size(RBarretTNFATGFBExpression_GMask,2)  
RBarretTNFATGFBExpression_GMask(:,i) =  
RBarretTNFATGFBCnt_GMask(:,i)/sum(RBarretTNFATGFBCnt_GMask(:,i))*1000000;  
end  
for i=1:size(RBarretTNFATGFBExpression_GMask)  
RBarretTNFATGFBExpression_GMask(i,:) =  
RBarretTNFATGFBExpression_GMask(i,:)/Gencode_33_Selected_MappSS_GMask(i)*1000;  
end  
RBarretTNFATGFBExpression_GMask(isnan(RBarretTNFATGFBExpression_GMask)) =  
0;  
RBarretTNFATGFBExpression_GMask(isinf(RBarretTNFATGFBExpression_GMask)) =  
0;  
  
% RBarretTNFATGFBCPM_GMask contains the expression data normalized only by  
CPM, using the same normalization method as the code above.  
RBarretTNFATGFBCPM_GMask = zeros(size(RBarretTNFATGFBCnt_GMask));  
for i=1:size(RBarretTNFATGFBCnt_GMask,2)  
RBarretTNFATGFBCPM_GMask(:,i) =  
RBarretTNFATGFBCnt_GMask(:,i)/sum(RBarretTNFATGFBCnt_GMask(:,i))*1000000;  
end  
  
% RBarretTNFATGFBTPM_GMask contains the expression data normalized only by  
TPM.  
RBarretTNFATGFBTPM_GMask = RBarretTNFATGFBCnt_GMask;  
for i=1:size(RBarretTNFATGFBCnt_GMask,1)  
RBarretTNFATGFBTPM_GMask(i,:) =  
RBarretTNFATGFBCnt_GMask(i,:)/Gencode_33_Selected_MappSS_GMask(i)*1000;  
end  
RBarretTNFATGFBTPM_GMask(isnan(RBarretTNFATGFBTPM_GMask)) = 0;  
RBarretTNFATGFBTPM_GMask(isinf(RBarretTNFATGFBTPM_GMask)) = 0;  
for i=1:size(RBarretTNFATGFBTPM_GMask,2)
```

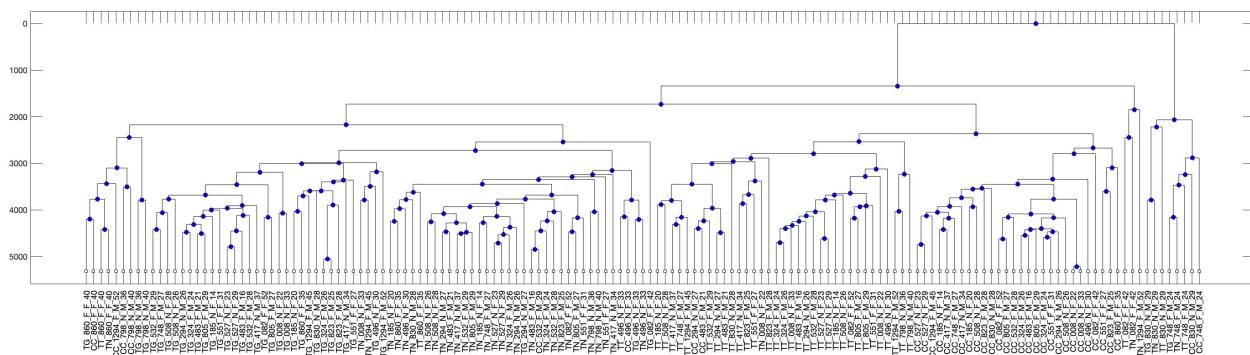
```
RBarretTNFATGFBTPM_GMask(:,i) =
RBarretTNFATGFBTPM_GMask(:,i)/sum(RBarretTNFATGFBTPM_GMask(:,i))*1000000;
end
```

## Dendrogram with only protein coding genes

Perform hierarchical clustering on the filtered gene expression data stored in the variable RBarretTNFATGFBTPM\_GMask:

```
thisrand = unique(randi([1 size(RBarretTNFATGFBTPM_GMask,1)],1,1000));
thisdist = pdist(RBarretTNFATGFBTPM_GMask(thisrand,:)');
for i=1:9999
    thisrand = unique(randi([1 size(RBarretTNFATGFBTPM_GMask,1)],1,1000));
    thisdist = thisdist+pdist(RBarretTNFATGFBTPM_GMask(thisrand,:)');
end
thisdistmat = squareform(thisdist/10000);
thistree = seqlinkage(thisdistmat,'average', RBarretsampleskeysTNFATGFB)
plot(thistree,'ORIENTATION','top')
```

Again, the samples are clustered largely by their treatment status but in a more consistent fashion as compared to the unfiltered dataset in Figure 6.



## The percent of the top 100 genes

Another item for quality control is achieved by calculating the percent of signal attributed to the top 100 expressed genes in each sample based on their transcript per million (TPM) values in the RBarretTNFATGFBTPM\_GMask matrix.

```
% iterates over 151 samples, it first sorts the TPM values of all genes in
descending order and stores the indices of the sorted genes in y. The top
```

```
100 expressed genes in the sample are obtained by selecting the first 100
indices in y, and these indices are appended to a running list of all top
100 indices yall.

yall=[];
for i=1:151
[x y]=sort(RBarretTNFATGFBTPM_GMask(:,i),'descend');
yall=unique([y(1:100); yall]);
top100percent(i)=sum(RBarretTNFATGFBTPM_GMask(y(1:100),i))/1000000;
end
```

I find that, for some samples, the top 100 most-expressed genes accumulate ~40% of the total TPMs for the sample, while the average is ~25%. I will keep track of these number in case those samples show outlier behaviour in downstream analyses.

In the end, we store the Gencode\_33\_Selected\_Geneid\_GMask.txt, Gencode\_33\_Selected\_Genename\_GMask.txt, Gencode\_33\_Selected\_MappSS\_GMask.txt, RBarretTNFATGFBTPM\_GMask.txt, and RBarretTNFATGFBCnt\_GMask.txt for ours further analysis in R.

```
writetable(cell2table(Gencode_33_Selected_Geneid_GMask), 'Gencode_33_Selected_Geneid_GMask')
writetable(cell2table(Gencode_33_Selected_Genename_GMask), 'Gencode_33_Selected_Genename_GMask')
dlmwrite('Gencode_33_Selected_MappSS_GMask.txt',
Gencode_33_Selected_MappSS_GMask, 'delimiter', '\t')
dlmwrite('RBarretTNFATGFBTPM_GMask.txt',
RBarretTNFATGFBTPM_GMask, 'delimiter', '\t')
dlmwrite('RBarretTNFATGFBCnt_GMask.txt',
RBarretTNFATGFBCnt_GMask, 'delimiter', '\t')
```

## Differential expression analysis in R

### Install packages

First, install packages BiocManager, BiocLite, IHW, DESeq2[7], and ggplot2. Then, read in RBarretTNFATGFBCnt\_GMask.txt, RBarretTNFATGFBsamples.txt, samplekeys\_Sam.txt, and Gencode\_33\_Selected\_Genename\_GMask.txt.

```
setwd("/Users/LuC/Desktop/Cedars-
Sinai/PROJECTS/IBD_RNASeq/RBARRETTNFATGFB/")
#setwd("/Users/samuellu/Desktop/Cedars-
Sinai/PROJECTS/IBD_RNASeq/RBARRETTNFATGFB/")

if (!requireNamespace("BiocManager", quietly = TRUE))
```

```

install.packages("BiocManager")

#BiocManager::install("BiocLite")
#BiocManager::install("IHW")
#BiocManager::install("DESeq2")
#install.packages("ggplot2")

library(DESeq2)
library(IHW)
library(ggplot2)
library(ggrepel)

RBarretTNFATGFCntGMask =
as.matrix(read.table("RBarretTNFATGFCnt_GMask.txt"))
sampleNameTNFATGFB = as.matrix(read.table("RBarretTNFATGFBsamples.txt"))
sampleKeyTNFATGFB = as.matrix(read.table("samplekeys_Sam.txt"))
genenames = as.matrix(read.table("Gencode_33_Selected_Genename_GMask.txt"))

```

### Form samplekeys\_Sam.tab

Separate samplekeys\_Sam.txt by "\_" to get samplekeys\_Sam.tab before next step. Here are my code in terminal.

```

#In terminal
#Create an empty file to store the output
touch samplekeys_Sam.tab

#Loop over the sample names and split them by "_"
for sample in $(cat samplekeys_Sam.txt); do
    IFS=_ read -r col1 col2 col3 col4 col5 <<< "$sample"
    echo -e "$col1\t$col2\t$col3\t$col4\t$col5" >> samplekeys_Sam.tab
done

```

### Generate a sampleTableTNFATGFB

The sampleTableTNFATGFB contains the experimental factors: Treatment, iPSC line (Line), fibrotic phenotype (Pheno), Sex, number of iPSC passages (Pass), the combination of phenotype and treatment (Factor), and the combination of line and passages (Batch) in Figure 7.

```

sampleTableTNFATGFB = read.table("samplekeys_Sam.tab")
rownames(sampleTableTNFATGFB)<-sampleKeyTNFATGFB
colnames(sampleTableTNFATGFB)<- c("Treatment","Line","Pheno","Sex","Pass")
sampleTableTNFATGFB$Factor <-
paste(sampleTableTNFATGFB$Treatment,sampleTableTNFATGFB$Pheno,sep="_")
#concatenating the "Line" and "Pass" columns with an underscore separator
sampleTableTNFATGFB$Batch <-
paste(sampleTableTNFATGFB$Line,sampleTableTNFATGFB$Pass,sep="_")
colnames(RBarretTNFATGFBCntGMask) <- sampleKeyTNFATGFB
write.table(sampleTableTNFATGFB,file="sampleTableTNFATGFB.txt", sep = "\t",
col.names = FALSE)

```

	Treatment	Line	Pheno	Sex	Pass	Factor	Batch
TG_008_N_F_22	TG	8	N	F	22	TG_N	8_22
TT_008_N_F_22	TT	8	N	F	22	TT_N	8_22
TN_008_N_F_22	TN	8	N	F	22	TN_N	8_22
CC_008_N_F_22	CC	8	N	F	22	CC_N	8_22
TG_008_N_F_33	TG	8	N	F	33	TG_N	8_33
TT_008_N_F_33	TT	8	N	F	33	TT_N	8_33
TN_008_N_F_33	TN	8	N	F	33	TN_N	8_33
CC_008_N_F_33	CC	8	N	F	33	CC_N	8_33
TG_082_F_F_52	TG	82	F	F	52	TG_F	82_52
TT_082_F_F_52	TT	82	F	F	52	TT_F	82_52
TN_082_F_F_52	TN	82	F	F	52	TN_F	82_52
CC_082_F_F_52	CC	82	F	F	52	CC_F	82_52

## DESeq2 package

Different experimental factors are tested while creating the DESeq object for differential expression, to check if there are significant differences. The RBarretTNFATGFBCntGMaskBatch is created with the **Batch** information specified in the design formula, while the RBarretTNFATGFBCntGMaskFactor is created with the **treatment and phenotype** information specified in the design formula. I also tested if fitting the data to the first principal component (PC1, see below) makes a difference in the first steps.

The DESeq function is used to estimate size factors and dispersion values for the DESeqDataSet objects. Using this object, we first use the varianceStabilizingTransformation function to perform variance stabilizing transformation. This transformation is important for reducing the effect of noise and impose heteroscedasticity in the data, making it more suitable for downstream analyses such as linear modeling and clustering.

```
RBarretTNFATGFBCntGMaskBatch <-
DESeqDataSetFromMatrix(RBarretTNFATGFBCntGMask, colData=
sampleTableTNFATGFB,design= ~Batch)
RBarretTNFATGFBCntGMaskBatch <- DESeq(RBarretTNFATGFBCntGMaskBatch)

RBarretTNFATGFBCntGMaskFactor <-
DESeqDataSetFromMatrix(RBarretTNFATGFBCntGMask, colData=
sampleTableTNFATGFB,design= ~Factor)
RBarretTNFATGFBCntGMaskFactor <- DESeq(RBarretTNFATGFBCntGMaskFactor)

RBarretTNFATGFBCntGMaskPC1 <-
DESeqDataSetFromMatrix(RBarretTNFATGFBCntGMask, colData= pcabatchR,design=
~PC1)
RBarretTNFATGFBCntGMaskPC1 <- DESeq(RBarretTNFATGFBCntGMaskPC1)

RBarretTNFATGFBCntGMaskBatch_vsd <-
varianceStabilizingTransformation(RBarretTNFATGFBCntGMaskBatch,blind=FALSE)
RBarretTNFATGFBCntGMaskFactor_vsd <-
varianceStabilizingTransformation(RBarretTNFATGFBCntGMaskFactor,blind=FALSE)
RBarretTNFATGFBCntGMaskPC1_vsd <-
varianceStabilizingTransformation(RBarretTNFATGFBCntGMaskPC1,blind=FALSE)
```

## Principal Component Analysis (PCA)

```
#perform principal component analysis (PCA) on the variance-stabilized
counts data
pcabatch <- prcomp(t(assay(RBarretTNFATGFBCntGMaskBatch_vsd)))

#give the percentage of variance explained by each principal component
percentVarbatch <- round(100*pcabatch$sdev^2/sum(pcabatch$sdev^2))

#pcabatch$rotation is a matrix containing the loadings of the principal
components.
aloadbatch <- abs(pcabatch$rotation)

#normalize the loadings in aloadbatch so that each column (i.e., PC) sums
to 1.
aloadrelativebatch <- sweep(aloadbatch, 2, colSums(aloadbatch), "/")

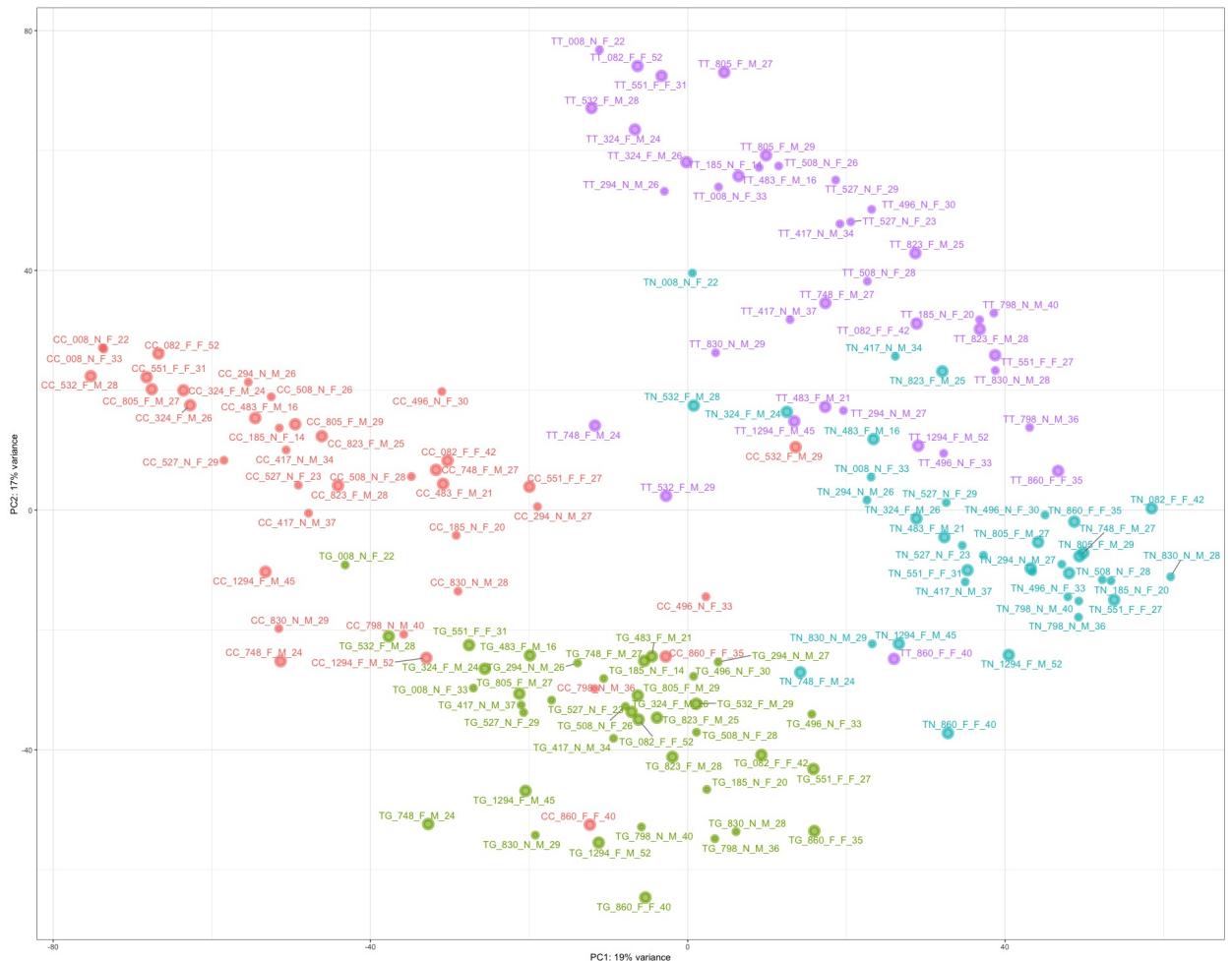
#pcabatch$x is a matrix containing each sample's coordinate on each
principal component
pcabatchALL <- pcabatch$x
pcabatchR<- cbind(pcabatchALL,sampleTableTNFATGFB)
```

## PCA plots

```
ggplot(pcabatchR, aes(PC1, PC2, color= Treatment)) +  
  geom_point(aes(size= Pheno),alpha=0.6,stroke = 3)+  
  xlab(paste0("PC1: ",percentVarbatch[1],"% variance")) +  
  ylab(paste0("PC2: ",percentVarbatch[2],"% variance")) +  
  geom_text_repel(aes(label = sampleKeyTNFATGFB),size=4,box.padding    =  
  0.35, point.padding = 0.5,segment.color = 'grey50')+ theme_bw()
```

The plot shows the clustering of all samples using the first two principal components, **PC1** and **PC2**, colored by **Pheno** variable, with the point size indicating the **Treatment** variable.

In Figure 8, four distinct groups were formed based on their treatment: the CC group (untreated) is located in the right corner, the TG group (treated with TGF-b) is located at the bottom, the TN group (treated with TNF-a) is located in the right corner, and the TT group (treated with both TGF-b and TNF-a) is located at the top. These 4 groups were differentiated based on the combination of both PC1 and PC2, which accounted for 19% and 17% of the variance, respectively.



## A series of bar plots (one for each principal component)

Each bar plot represents the coordinates of all samples on a given principal component. These plots provide a quick visual evaluation of the potential association of each component with specific experimental factors.

```
#the resulting vector coul will contain 12 colors from the "Set3" palette.
library(RColorBrewer)
coul <- brewer.pal(12, "Set3")
#generates colors for a plot based on the batch variable
colors=pcabatchR$Batch
allbatches<-unique(pcabatchR$Batch)
for (i in 1:38){
  colors[pcabatchR$Batch==allbatches[i]]<-coul[i%%12+1]
}

thinlines=c(seq(4,72,8),75,seq(83,151,8))
thicklines=c(seq(8,72,8),79,seq(87,151,8))

#first half of the barplot would be the non-fibrotic group and the second
part would be the fibrotic group
#the order would be CC, TG, TN, TT
```

```

samplesorder=c(4,1,3,2,8,5,7,6,28,25,27,26,32,29,31,30,36,33,35,34,40,37,39,38

#create 38 barplots and saving each of them as a PNG file
for (i in 1:38) {
  filename = paste("PC_",i,".png", sep = "")
  png(filename)

  barplot(pcabatchALL[samplesorder,i],col=colors[samplesorder],las=2,xaxt='n',sp
  for (i in 1:length(thinlines)) {
    abline(v = thinlines[i], col = "black",lty = 3)
  }
  for (i in 1:length(thicklines)) {
    abline(v = thicklines[i], col = "black",lty = 1)
  }
  abline(v = 72, col = "red",lty = 1)
  dev.off()
}

write.csv(aloadrelativebatch,file="aloadrelativeMask_batchmodel_filtered.csv")
write.csv(pcabatch$x,file="pca_batchmodel_x.csv")

```

To facilitate visualization, a red line is drawn at position 72 in order to separate the non-fibrotic group from the fibrotic group. Within each patient, the treatment order would be CC, TG, TN, TT. The color of each bar represents the batch of the sample, with a unique color assigned to each batch. The vertical lines on the plot separate the data for individual iPSC lines and their two batches.

The Figure 9 below represents PC1 for all samples. From this plot, one can see that PC1 corresponds to extreme expression after treatment with TNF-a in all cases, even more than after its combination with TGF-b. Therefore, it seems to indicate that PC1 is associated with an interaction between TNF-a and TGF-b in iPSC mesenchymal organoids, an unexpected finding that warrants further analysis.

## PCA rank matrix

For easier visualization, I next clean the PCA results for exporting into spreadsheets. Take csv files and converts it to the txt files with the second column onwards. It does this by first removing the first row using awk, replacing multiple commas with tabs using tr, and removing the first column using cut.

```

#In terminal
cat aloadrelativeMask_batchmodel_filtered.csv | awk 'NR>1{print}' | tr -s
"," "\t" | cut -f 2- > aloadrelativeMask_batchmodel_filtered.clean.txt

```

```
cat pca_batchmodel_x.csv | awk 'NR>1{print}' | tr -s "," "\t" | cut -f 2- > pca_batchmodel_x.clean.txt
```

Read in the preprocessed data files created in the previous steps and store them in variables pcabatch\_samples and pca\_loadings, respectively.

```
#In Matlab
pcabatch_samples = textread('pca_batchmodel_x.clean.txt','');
pca_loadings =
textread('aloadrelativeMask_batchmodel_filtered.clean.txt');
```

Sort the three columns of pca\_loadings in descending order and store the sorted values in variables x1, x2, and x3, and the corresponding indices in y1, y2, and y3.

```
%x = pca_loading number, y = its index

[x1 y1]=sort(pca_loadings(:,1),'descend');
[x2 y2]=sort(pca_loadings(:,2),'descend');
[x3 y3]=sort(pca_loadings(:,3),'descend');
```

Determine the rank of each row in the original order for the first three principal components and store the ranks in a matrix pcarankmatrix.

```
[x y z]=intersect(1:length(y1),y1);
pcarankmatrix(:,1)=z;
[x y z]=intersect(1:length(y2),y2);
pcarankmatrix(:,2)=z;
[x y z]=intersect(1:length(y3),y3);
pcarankmatrix(:,3)=z;

%contains the rank of each feature in the original order for the first
three principal components

dlmwrite('pcarankmatrix.txt', pcarankmatrix,'delimiter','\t')
```

To efficiently manage our data with a single glance, I have organized it into an Excel spreadsheet using a combination of command line, Excel, and R.

# Spreadsheet

The Figure 10 (patients) built on excel contains patients order, patients id, phenotypes, and sex. You can visit the sheet by clicking [here](#).

ID	Phenotype	Gender		1	008IP22TGFbM_571	N	F	TG_008_N_F_22		TG_008_N_F_22	TG	N	F	22_TG_N	8_22
16	082	Bifibrotic		2	008IP22TGFbM_572	N	F	TG_008_N_F_22		TG_008_N_F_22	TG	N	F	22_TN_N	8_22
12	324	Bifibrotic	M	3	008IP22TGFbM_570	N	F	TN_008_N_F_22		TN_008_N_F_22	TN	N	F	22_TN_N	8_22
13	483	Bifibrotic	M	4	008IP22UntreatedM_569	N	F	CC_008_N_F_22		CC_008_N_F_22	CC	N	F	22_CC_N	8_22
17	532	Bifibrotic	M	5	008IP33TGFbM_575	N	F	TG_008_N_F_33		TG_008_N_F_33	TG	N	F	33_TG_N	8_33
11	551	Bifibrotic	F	6	008IP33TGFbM_576	N	F	TG_008_N_F_33		TG_008_N_F_33	TG	N	F	33_TG_N	8_33
19	748	Bifibrotic	M	7	008IP33TGFbM_559	N	F	TN_008_N_F_33		TN_008_N_F_33	TN	N	F	33_TN_N	8_33
10	805	Bifibrotic	M	8	008IP33UntreatedM_573	N	F	CC_008_N_F_33		CC_008_N_F_33	CC	N	F	33_CC_N	8_33
14	823	Bifibrotic	M	9	082IP52TGFbM_561	F	F	TG_082_F_F_52		TG_082_F_F_52	TG	F	F	52_TG_F	8_52
15	860	Bifibrotic	F	10	082IP52TGFbM_580	F	F	TT_082_F_F_52		TT_082_F_F_52	TT	F	F	52_TT_F	8_52
18	1294	Bifibrotic	M	11	082IP52TGFbM_560	F	F	TN_082_F_F_52		TN_082_F_F_52	TN	F	F	52_TN_F	8_52
6	008	Non-fibrotic	F	12	082IP52UntreatedM_577	F	F	CC_082_F_F_52		CC_082_F_F_52	CC	F	F	52_CC_F	8_52
3	185	Non-fibrotic	F	13	082IP52TGFbM_5129	F	F	TG_082_F_F_42		TG_082_F_F_42	TG	F	F	42_TG_F	8_42
4	294	Non-fibrotic	M	14	082IP52TGFbM_5130	F	F	TT_082_F_F_42		TT_082_F_F_42	TT	F	F	42_TT_F	8_42
2	417	Non-fibrotic	M	15	082IP52TGFbM_5128	F	F	TN_082_F_F_42		TN_082_F_F_42	TN	F	F	42_TN_F	8_42
7	496	Non-fibrotic	F	16	082IP52UntreatedM_5127	F	F	CC_082_F_F_42		CC_082_F_F_42	CC	F	F	42_CC_F	8_42
5	508	Non-fibrotic	F	17	1294IP52TGFbM_557	F	F	TG_1294_F_M_45		TG_1294_F_M_45	TG	F	M	45_TG_F	1294_45
8	527	Non-fibrotic	F	18	1294IP52TGFbM_564	F	F	TT_1294_F_M_45		TT_1294_F_M_45	TT	F	M	45_TT_F	1294_45
1	798	Non-fibrotic	M	19	1294IP52TGFbM_556	F	F	TN_1294_F_M_45		TN_1294_F_M_45	TN	F	M	45_TN_F	1294_45
9	830	Non-fibrotic	M	20	1294IP52UntreatedM_561	F	F	CC_1294_F_M_45		CC_1294_F_M_45	CC	F	M	45_CC_F	1294_45
				21	1294IP52TGFbM_567	F	F	TG_1294_F_M_52		TG_1294_F_M_52	TG	F	M	52_TG_F	1294_52
				22	1294IP52TGFbM_568	F	F	TT_1294_F_M_52		TT_1294_F_M_52	TT	F	M	52_TT_F	1294_52
				23	1294IP52TGFbM_558	F	F	TN_1294_F_M_52		TN_1294_F_M_52	TN	F	M	52_TN_F	1294_52
				24	1294IP52UntreatedM_565	F	F	CC_1294_F_M_52		CC_1294_F_M_52	CC	F	M	52_CC_F	1294_52
				25	185IP20TGFbM_583	N	F	TG_185_N_F_20		TG_185_N_F_20	TG	N	F	20_TG_N	8_20
				26	185IP20TGFbM_584	N	F	TT_185_N_F_20		TT_185_N_F_20	TT	N	F	20_TT_N	8_20
				27	185IP20TGFbM_582	N	F	TN_185_N_F_20		TN_185_N_F_20	TN	N	F	20_TN_N	8_20
				28	185IP20UntreatedM_581	N	F	CC_185_N_F_20		CC_185_N_F_20	CC	N	F	20_CC_N	8_20
				29	185IP14TGFbM_5133	N	F	TG_185_N_F_14		TG_185_N_F_14	TG	N	F	14_TG_N	8_14
				30	185IP14TGFbM_5134	N	F	TT_185_N_F_14		TT_185_N_F_14	TT	N	F	14_TT_N	8_14
				31	185IP14TGFbM_5132	N	F	TN_185_N_F_14		TN_185_N_F_14	TN	N	F	14_TN_N	8_14
				32	185IP14UntreatedM_5131	N	F	CC_185_N_F_14		CC_185_N_F_14	CC	N	F	14_CC_N	8_14
				33	294IP26TGFbM_587	N	M	TG_294_N_M_26		TG_294_N_M_26	TG	N	M	26_TG_N	294_26
				34	294IP26TGFbM_588	N	M	TT_294_N_M_26		TT_294_N_M_26	TT	N	M	26_TT_N	294_26
				35	294IP26TGFbM_586	N	M	TC_294_N_M_26		TC_294_N_M_26	TC	N	M	26_TC_N	294_26
				36	294IP26TGFbM_585	N	M	CC_294_N_M_26		CC_294_N_M_26	CC	N	M	26_CC_N	294_26
				37	294IP26TGFbM_565	N	M	TG_294_N_M_27		TG_294_N_M_27	TG	N	M	27_TG_N	294_27
				38	294IP27TGFbM_595	N	M	TT_294_N_M_27		TT_294_N_M_27	TT	N	M	27_TT_N	294_27
				39	294IP27TGFbM_562	N	M	TN_294_N_M_27		TN_294_N_M_27	TN	N	M	27_TN_N	294_27
				40	294IP27UntreatedM_570	N	M	CC_294_N_M_27		CC_294_N_M_27	CC	N	M	27_CC_N	294_27
				41	324IP24TGFbM_565	F	M	TG_324_F_M_24		TG_324_F_M_24	TG	F	M	24_TG_F	324_24
				42	324IP24TGFbM_593	F	M	TT_324_F_M_24		TT_324_F_M_24	TT	F	M	24_TT_F	324_24
				43	324IP24TGFbM_564	F	M	TN_324_F_M_24		TN_324_F_M_24	TN	F	M	24_TN_F	324_24
				44	324IP24UntreatedM_563	F	M	CC_324_F_M_24		CC_324_F_M_24	CC	F	M	24_CC_F	324_24
				45	324IP26TGFbM_5137	F	M	TG_324_F_M_26		TG_324_F_M_26	TG	F	M	26_TG_F	324_26
				46	324IP26TGFbM_5138	F	M	TT_324_F_M_26		TT_324_F_M_26	TT	F	M	26_TT_F	324_26
				47	324IP26TGFbM_5136	F	M	TN_324_F_M_26		TN_324_F_M_26	TN	F	M	26_TN_F	324_26
				48	324IP26UntreatedM_5135	F	M	CC_324_F_M_26		CC_324_F_M_26	CC	F	M	26_CC_F	324_26

The Figure 11 (allbiotypes\_percent) includes the names and percentages of all biotypes, along with their respective minimum, maximum, and average values, providing us with a comprehensive overview. You can visit the sheet by clicking [here](#).

```
#In terminal
paste allbiotypes allbiotypescountspercents >
combine_allbiotypes_percents.txt

#In R
#allbiotypes_percents
sheet2_1 <- list("Biotypes")
sheet2_2 <- sampleKeyTNFATGFB
combined_sheet2 <- c(sheet2_1, sheet2_2)
combined_spreadsheet2 <-
as.matrix(read.table("combine_allbiotypes_percents.txt"))
colnames(combined_spreadsheet2) <- combined_sheet2
write.table(combined_spreadsheet2,file="combined_spreadsheet2.txt", sep =
"\t", row.names = FALSE)
#add their respective minimum, maximum, and average values on Excel
```

Biotypes	min	max	average	TG_008_N_F_22	TT_008_N_F_22	TN_008_N_F_22	CC_008_N_F_22	TG_008_N_F_33	TT_008_N_F_33	TN_008_N_F_33	CC_008_N_F_33
IG_C_gene	0.000000000	0.000446580	0.000013340	0.000000000	0.000000000	0.000000000	0.000000000	0.000003012	0.000011577	0.000000000	0.000000000
IG_D_gene	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
IG_J_gene	0.000000000	0.000013813	0.000000235	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
IG_V_gene	0.000000000	0.000073058	0.000010604	0.000012755	0.000010996	0.000008912	0.000002556	0.000015061	0.000008269	0.000010462	0.000011520
Mt_rRNA	0.228350000	1.165600000	0.487934371	0.495080000	0.584550000	0.439920000	0.826140000	0.508050000	0.550660000	0.508120000	0.859590000
Mt_tRNA	0.000141280	0.002049800	0.000492376	0.000214290	0.000274910	0.000199040	0.000345110	0.000240970	0.000219970	0.000237130	0.000336390
TEC	0.011838000	0.029033000	0.018976556	0.017090000	0.016613000	0.016773000	0.021052000	0.017133000	0.019370000	0.017489000	0.021066000
TR_C_gene	0.000000000	0.002667100	0.000287637	0.000002551	0.000000000	0.000005942	0.000000000	0.000018073	0.000011577	0.000007182	0.000009216
TR_D_gene	0.000000000	0.000011542	0.000000580	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
TR_J_gene	0.000000000	0.000076445	0.000009219	0.000000000	0.000000000	0.000000000	0.000010225	0.000009036	0.000000000	0.000006975	0.000002304
TR_V_gene	0.000000000	0.000061956	0.000011904	0.000000000	0.0000008247	0.000005942	0.000010225	0.000003012	0.000013231	0.000006975	0.000016129
lncRNA	0.537580000	1.612500000	0.825408079	0.832110000	0.782160000	0.759180000	1.060300000	0.858370000	0.828130000	0.792760000	1.102900000
miRNA	0.002352100	0.014811000	0.004202784	0.004000100	0.004865900	0.004536300	0.003474100	0.003707900	0.004582900	0.005147200	0.003327100
misc_RNA	0.000641580	0.002443400	0.001048973	0.001140300	0.001300300	0.001265500	0.001250100	0.001015100	0.001283400	0.001227500	0.001101300
protein_codi	<b>97.675000000</b>	<b>99.144000000</b>	<b>98.658132450</b>	<b>98.647000000</b>	<b>98.607000000</b>	<b>98.775000000</b>	<b>98.084000000</b>	<b>98.608000000</b>	<b>98.592000000</b>	<b>98.671000000</b>	<b>98.008000000</b>
rRNA	0.000000000	0.000028828	0.000005066	0.000000000	0.0000008247	0.0000008912	0.000000000	0.0000003012	0.000000000	0.000010462	0.000002304
ribozyme	0.000000000	0.000007357	0.000000293	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
sRNA	0.000000000	0.000004680	0.000000135	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.0000001654	0.000000000
scRNA	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
scaRNA	0.000002616	0.000324330	0.000306518	0.000012755	0.000024742	0.000062386	0.000040902	0.000024097	0.000031424	0.000027898	0.000052994
snRNA	0.000262070	0.001464200	0.000543351	0.000413270	0.000519580	0.000534740	0.000682550	0.000590380	0.000583820	0.000606780	0.000580630
snoRNA	0.001727500	0.004335800	0.002883131	0.002719400	0.002650100	0.002857900	0.003037000	0.002970000	0.003486400	0.002978100	0.003205000
vaultRNA	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000

The Figure 12 is the main sheet that includes Genename, Geneid, Mapp, PC1, PC2, PC3, and patient's TPM values.

#In terminal

```
paste Gencode_33_Selected_Genename_GMask.txt
Gencode_33_Selected_Genename_GMask.txt Gencode_33_Selected_Geneid_GMask.txt
Gencode_33_Selected_MappSS_GMask.txt pcarankmatrix.txt > combine_test.txt
```

#In R

#spreadsheet

```
sheet3_1 <- list("Genename","Genename","Geneid","Mapp","PC1","PC2","PC3")
sheet3_2<- sampleKeyTNFATGFB
combined_headers <- c(sheet3_1, sheet3_2)
combined_spreadsheet <- as.matrix(read.table("combine_test.txt"))
colnames(combined_spreadsheet) <- combined_headers
combined_spreadsheet <-
combined_spreadsheet[order(combined_spreadsheet[,1]),] #sort by the first column
write.table(combined_spreadsheet,file="combined_spreadsheet.txt", sep = "\t", row.names = FALSE)
```

You can sort this sheet with PC1, PC2, and so on to see the correlation between the experimental factors and the expression level of each gene, which allows a quick identification of genes and gene classes more associated with the dominant sources of gene expression variability in this experiment.

# DESeq analysis for pairwise comparisons

## Reorder data by phenotypes and patients (lines)

```
samplesorder=c(4,1,3,2,8,5,7,6,28,25,27,26,32,29,31,30,36,33,35,34,40,37,39,38)
```

Add the PC1 coordinate of each sample into the experimental design, to be used as factor in the model

```
BarretMyofCnt=RBarretTNFATGFCntGMask[,samplesorder]
sampleTableMyof=cbind(sampleTableTNFATGFB[samplesorder,],pcabatchALL[samplesor
sampleTableMyof$PC1 <- scale(sampleTableMyof$PC1 , center = TRUE)
```

```
#normalized and batch-corrected expression matrix
#using the variance stabilizing transformation
BarretMY0_Batch_vsdf=RBarrettTNFATGFBCntGMaskBatch_vsdf[,samplesorder]
```

```
write.csv(assay(BarretMY0_Batch_vs), file="BarretMY0_Batch_vs.csv")
```

## Check gene name duplicates

```
Genenames =  
as.list(read.table("Gencode 33 Selected Genename GMask.txt", header=FALSE, as.is=TRUE))
```

```
#Find duplicate gene names
duplicated_genes <- Genenames$V1[duplicated(Genenames$V1)]
if(length(duplicated_genes) > 0){
  cat("Duplicate gene names found:", paste(duplicated_genes, collapse = ","))
} else {
  cat("No duplicate gene names found.")
}

#Duplicate gene names found: TBCE, ATXN7, AHRR, MATR3, HSPA14, TMSB15B

#Find duplicate gene names at indices
duplicated_indices <- which(duplicated(Genenames$V1))
if(length(duplicated_indices) > 0){
  cat("Duplicate gene names found at indices:", paste(duplicated_indices,
collapse = ", "))
} else {
  cat("No duplicate gene names found.")
}
#Duplicate gene names found at indices: 1530, 3024, 4133, 4576, 7638, 15459
```

```
#Correct the duplicated gene names
#by appending "_1" to the end of each duplicate gene name
Genenames$V1[1530] <- "TBCE_1"
Genenames$V1[3024] <- "ATXN7_1"
Genenames$V1[4133] <- "AHRR_1"
Genenames$V1[4576] <- "MATR3_1"
Genenames$V1[7638] <- "HSPA14_1"
Genenames$V1[15459] <- "TMSB15B_1"
```

### First round of pairwise comparisons:

All cell lines UNTREATED vs All cell lines TGF $\beta$

All cell lines UNTREATED vs All cell lines TNF $\alpha$

All cell lines UNTREATED vs All cell lines TNF $\alpha$ /TGF $\beta$

We model the data correcting for PC1 and Line (patient-specific expression) and then test for the treatment effect

```
#The factor "Batch" is included as a covariate to account for potential
```

```
batch effects,
#and the factor "Treatment" is included as the variable of interest for
differential expression analysis.
```

```
RBarretMYOFCntGMaskTreatment <- DESeqDataSetFromMatrix(BarretMyofCnt,
colData= sampleTableMyof, design= ~ Batch + Treatment)
RBarretMYOFCntGMaskTreatment <- DESeq(RBarretMYOFCntGMaskTreatment)
```

The filterFun argument specifies a multiple testing correction method to apply to the results, in this case the independent hypothesis weighting (IHW) method.

```
#Calculate differential expression results for the pairwise comparisons of
#the TG vs CC, TN vs CC, and TT vs CC treatment groups, respectively.
```

```
#The resulting output for each comparison will contain a table of genes
#with their corresponding LFCs, p-values, and adjusted p-values based on
the specified multiple testing correction method.
```

```
RBarretMYOFCntGMaskTreatment_TG <-
results(RBarretMYOFCntGMaskTreatment, contrast=c("Treatment", "TG",
"CC"), filterFun=ihw)
RBarretMYOFCntGMaskTreatment_TN <-
results(RBarretMYOFCntGMaskTreatment, contrast=c("Treatment", "TN",
"CC"), filterFun=ihw)
RBarretMYOFCntGMaskTreatment_TT <-
results(RBarretMYOFCntGMaskTreatment, contrast=c("Treatment", "TT",
"CC"), filterFun=ihw)
```

Visually check the names of the most significant genes (very low adjusted p-value)

```
Genenames$V1[which(RBarretMYOFCntGMaskTreatment_TG$padj<0.00000000000001)]
Genenames$V1[which(RBarretMYOFCntGMaskTreatment_TN$padj<0.00000000000001)]
Genenames$V1[which(RBarretMYOFCntGMaskTreatment_TT$padj<0.00000000000001)]
```

### Second round of pairwise comparisons:

Untreated NON-FIBROTIC cell lines vs Untreated FIBROTIC cell lines

TNF $\alpha$ /TGF $\beta$  NON-FIBROTIC cell lines vs TNF $\alpha$ /TGF $\beta$  FIBROTIC cell lines

TGF $\beta$  NON-FIBROTIC cell lines vs TGF $\beta$  FIBROTIC cell lines

## TNF $\alpha$ NON-FIBROTIC cell lines vs TNF $\alpha$ FIBROTIC cell lines

We model the data correcting for Line (patient-specific expression) and then test for the Factor (treatment+phenotype combination) effect.

```
RBarretMYOFCntGMaskFactor <- DESeqDataSetFromMatrix(BarretMyofCnt, colData=sampleTableMyof, design= ~ Line + Factor)
RBarretMYOFCntGMaskFactor <- DESeq(RBarretMYOFCntGMaskFactor)
```

```
#Compute differential expression analysis results for the four contrasts of interest "CC_F" vs "CC_N", "TG_F" vs "TG_N", "TN_F" vs "TN_N", and "TT_F" vs "TT_N" in the dataset.
```

```
RBarretMYOFCntGMaskFactor_CC_Pheno <-
results(RBarretMYOFCntGMaskFactor, contrast=c("Factor", "CC_F",
"CC_N"), filterFun=ihw)
RBarretMYOFCntGMaskFactor_TG_Pheno <-
results(RBarretMYOFCntGMaskFactor, contrast=c("Factor", "TG_F",
"TG_N"), filterFun=ihw)
RBarretMYOFCntGMaskFactor_TN_Pheno <-
results(RBarretMYOFCntGMaskFactor, contrast=c("Factor", "TN_F",
"TN_N"), filterFun=ihw)
RBarretMYOFCntGMaskFactor_TT_Pheno <-
results(RBarretMYOFCntGMaskFactor, contrast=c("Factor", "TT_F",
"TT_N"), filterFun=ihw)
```

```
Genenames$V1[which(RBarretMYOFCntGMaskFactor_CC_Pheno$padj<0.01)]
Genenames$V1[which(RBarretMYOFCntGMaskFactor_TG_Pheno$padj<0.01)]
Genenames$V1[which(RBarretMYOFCntGMaskFactor_TN_Pheno$padj<0.01)]
Genenames$V1[which(RBarretMYOFCntGMaskFactor_TT_Pheno$padj<0.01)]
```

## Export and process results

```
write.csv(as.data.frame(RBarretMYOFCntGMaskTreatment_TG), file="RBarretMYOFCntG
write.csv(as.data.frame(RBarretMYOFCntGMaskTreatment_TN), file="RBarretMYOFCntG
write.csv(as.data.frame(RBarretMYOFCntGMaskTreatment_TT), file="RBarretMYOFCntG
write.csv(as.data.frame(RBarretMYOFCntGMaskFactor_CC_Pheno), file="RBarretMYOFC
write.csv(as.data.frame(RBarretMYOFCntGMaskFactor_TG_Pheno), file="RBarretMYOFC
write.csv(as.data.frame(RBarretMYOFCntGMaskFactor_TN_Pheno), file="RBarretMYOFC
write.csv(as.data.frame(RBarretMYOFCntGMaskFactor_TT_Pheno), file="RBarretMYOFC
```

## Pairwise results shreadsheet

Paste Gencode\_33\_Selected\_Geneid\_GMask.txt,  
Gencode\_33\_Selected\_Genename\_GMask.txt,  
Gencode\_33\_Selected\_MappSS\_GMask.txt,  
RBarretMYOFCntGMaskFactor\_CC\_Pheno\_test.txt,  
RBarretMYOFCntGMaskFactor\_TG\_Pheno\_test.txt,  
RBarretMYOFCntGMaskFactor\_TN\_Pheno\_test.txt,  
RBarretMYOFCntGMaskFactor\_TT\_Pheno\_test.txt,  
RBarretMYOFCntGMaskTreatment\_TG\_test.txt,  
RBarretMYOFCntGMaskTreatment\_TN\_test.txt, and  
RBarretMYOFCntGMaskTreatment\_TT\_test.txt to formulate a shreadsheet.

#In terminal

```
tail -n +2 RBarretMYOFCntGMaskFactor_CC_Pheno.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskFactor_CC_Pheno_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskFactor_TG_Pheno.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskFactor_TG_Pheno_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskFactor_TN_Pheno.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskFactor_TN_Pheno_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskFactor_TT_Pheno.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskFactor_TT_Pheno_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskTreatment_TG.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskTreatment_TG_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskTreatment_TN.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskTreatment_TN_test.txt
```

```
tail -n +2 RBarretMYOFCntGMaskTreatment_TT.txt | cut -d"," -f3,7 >  
RBarretMYOFCntGMaskTreatment_TT_test.txt
```

```
paste Gencode_33_Selected_Geneid_GMask.txt  
Gencode_33_Selected_Genename_GMask.txt Gencode_33_Selected_MappSS_GMask.txt  
RBarretMYOFCntGMaskFactor_CC_Pheno_test.txt  
RBarretMYOFCntGMaskFactor_TG_Pheno_test.txt  
RBarretMYOFCntGMaskFactor_TN_Pheno_test.txt  
RBarretMYOFCntGMaskFactor_TT_Pheno_test.txt  
RBarretMYOFCntGMaskTreatment_TG_test.txt  
RBarretMYOFCntGMaskTreatment_TN_test.txt  
RBarretMYOFCntGMaskTreatment_TT_test.txt >  
Barret_Myofibroblast_TGFTNF_PAIRWISEResults.txt
```

```
sed 's/,/\t/g' Barret_Myofibroblast_TGFTNF_PAIRWISEResults.txt >
```

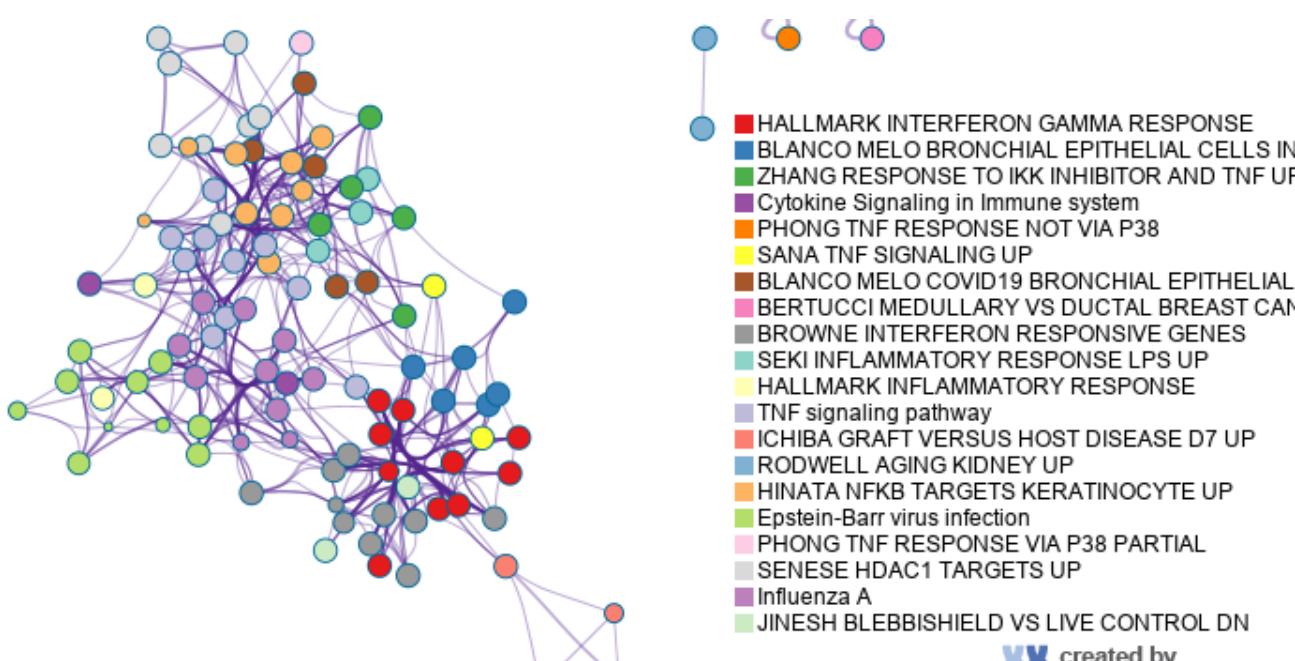
## Barret\_Myofibroblast\_TGFTNF\_PAIRWISEResults.txt

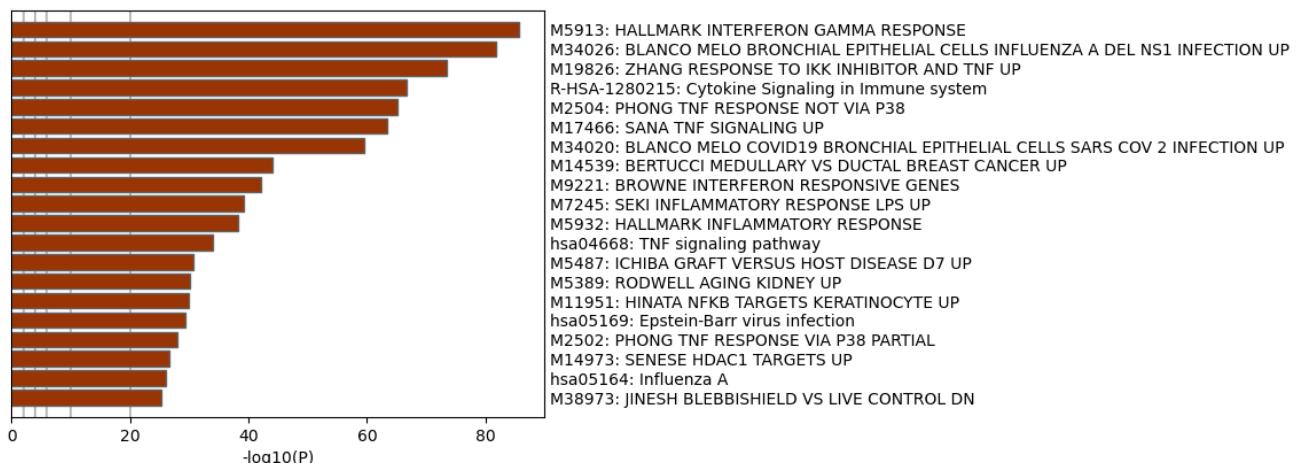
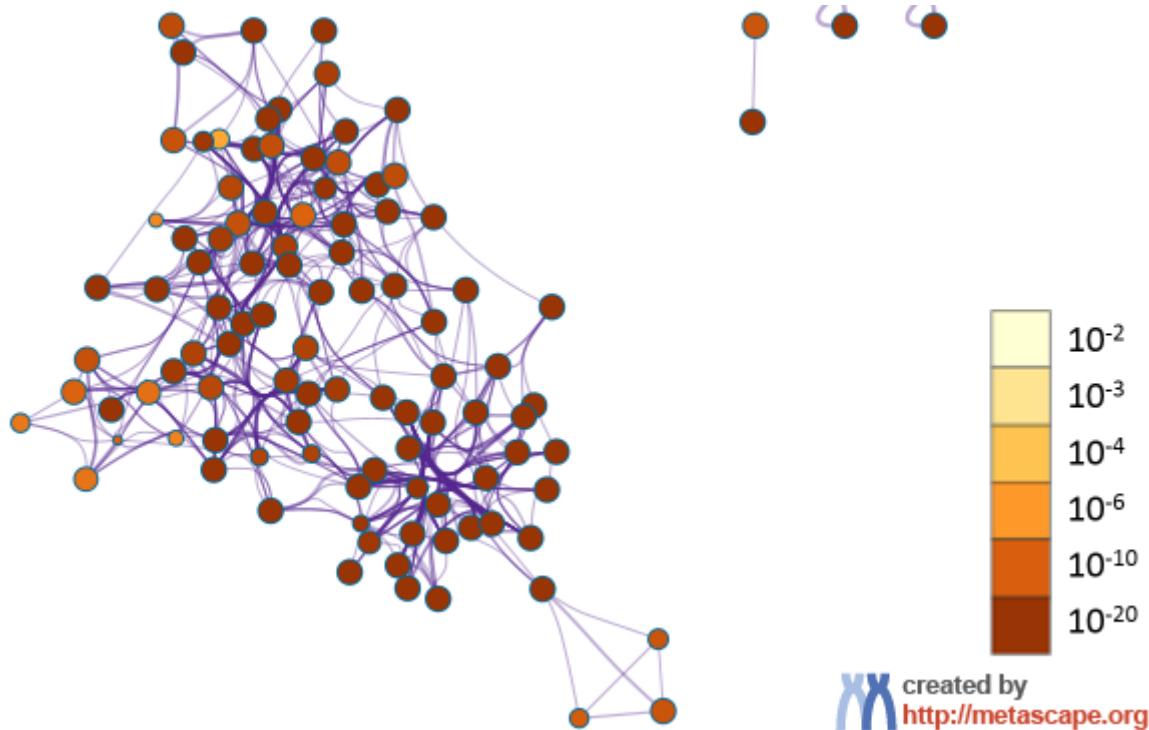
In Figure 13, it shows a clear formatted sheet containing Gene ID, Gene name, Mappability, fold change, and adjusted p-value. Then, you can sort each adjusted p-value column to focus on genes you are interested in. You can find the pairwise results spreadsheet [here](#).

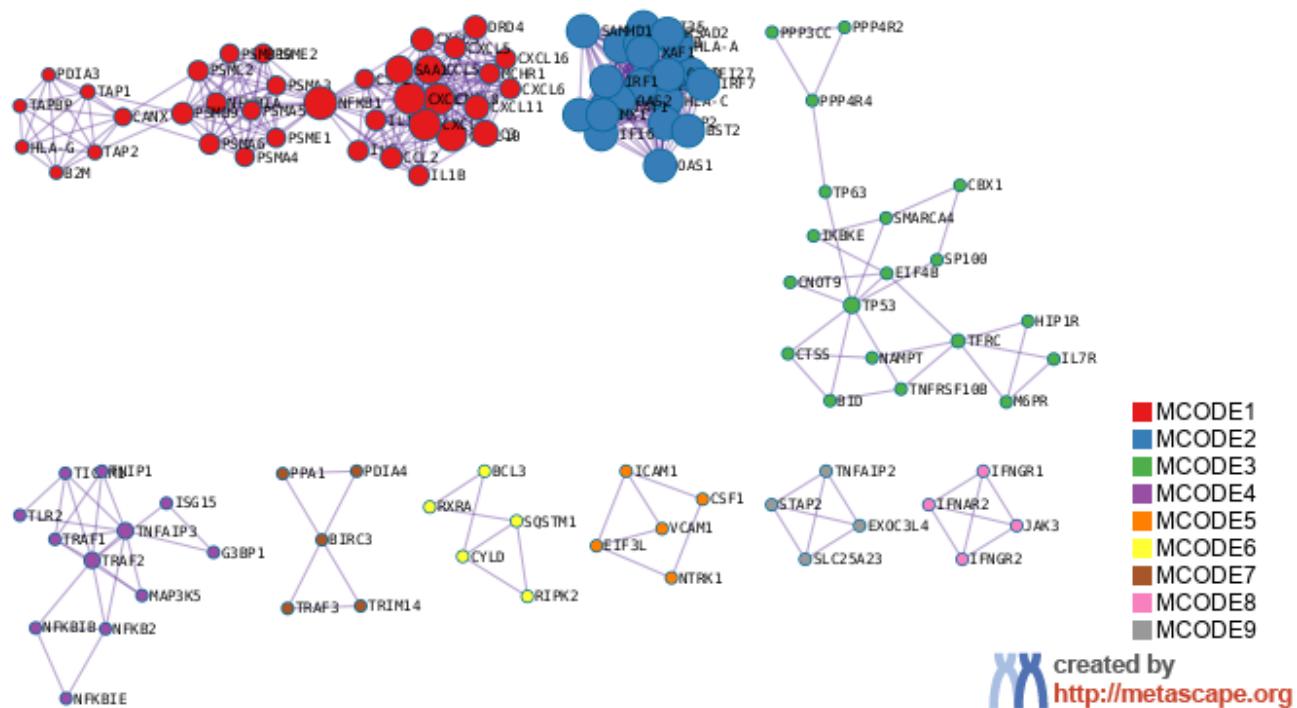
Gene ID	Gene name	Mappability	Untreated NON-FIBROTIC cell lines vs Untreated FIBROTIC cell lines		TGFβ NON-FIBROTIC cell lines vs TGFβ FIBROTIC cell lines		TNFα NON-FIBROTIC cell lines vs TNFα FIBROTIC cell lines		TNFα/TGFβ NON-FIBROTIC cell lines vs TNFα/TGFβ FIBROTIC cell lines		All cell lines UNTREATED vs All cell lines TGFβ		All cell lines UNTREATED vs All cell lines TNFα		All cell lines UNTREATED vs All cell lines TNFα/TGFβ	
			fold change	adjusted p-value	fold change	adjusted p-value	fold change	adjusted p-value	fold change	adjusted p-value	fold change	adjusted p-value	fold change	adjusted p-value	fold change	adjusted p-value
ENSG00000187634.12	SAMD11	4521	0.596611691	1	0.226814098	1	0.378338171	1	0.919392421	0.336037463	5.20E-43	0.081053744	0.817710494	-0.73358626	2.77E-18	
ENSG00000188976.11	NOC2L	2644	0.068821384	1	-0.000431352	1	0.040269268	1	0.191217181	0.145174491	2.36E-07	0.020558933	5.95E-14	0.10246128	0.000280382	
ENSG00000187961.14	KLHL17	2896	0.172917015	0.832705293	0.136320927	0.989972294	0.08539404	1	0.208050404	0.61896006	0.327769911	2.15E-17	0.127232327	0.001907649	-0.138919428	0.0050619
ENSG00000187583.11	PLEKHN1	3613	-0.090864126	0.827806587	-0.028993874	1	-0.160520142	0.807752678	-0.134336653	0.824087967	3.48E-39	2.162682366	4.62E-57	0.10101939	7.40E-15	
ENSG00000187642.9	PERM1	3296	-2.18570595	0.040394778	0.216761266	1	0.058135914	0.368492077	0.975846392	-0.598395081	0.081212314	-1.5051354176	0.000323565	-0.870103019	0.017890754	
ENSG00000188290.10	HES4	1133	0.153300669	1	0.122066297	1	0.218157143	0.887299992	-0.134392474	0.589823313	0.20649624	0.07818297	0.937081074	8.76E-19	0.371064905	0.000713975
ENSG00000187608.10	ISG15	861	-0.102148425	1	0.024595762	1	0.021694639	1	-0.279257826	0.186993258	1.601800299	8.39E-25	0.379882001	7.91E-89		
ENSG00000188157.15	AGRN	7570	-0.160468497	1	-0.075395693	1	-0.193898673	0.995982556	-0.163035751	0.584145053	1.72E-18	1.150947439	2.61E-68	1.20353386	3.41E-75	
ENSG00000131591.17	C1orf159	4519	-0.03309248	1	0.024669475	1	-0.027725627	1	-0.088624678	0.768141664	1.08E-17	0.084932533	0.012031563	0.278258925	1.32E-18	
ENSG00000186869.14	TNFRSF18	1359	0.886403284	0.823129353	0.446386643	0.96514689	0.117902521	1	0.33864822	1	-1.177370894	4.28E-06	0.54518563	4.38E-123	5.677301524	1.23E-135
ENSG00000186827.11	TNFRSF4	976	-0.514160059	0.664970317	-0.612637224	0.816114951	-0.860106585	0.46831063	-0.253356711	0.760286957	0.333418897	0.330752548	4.4995402	3.17E-56	5.52886735	1.09E-85
ENSG00000078808.17	SDF4	2308	0.049625524	1	0.057274721	1	0.016151239	1	0.003560533	1	0.142337521	0.000149013	0.571263300	2.52E-58	0.93214946	2.09E-155
ENSG00000176022.7	B3GALT6	2706	0.085310973	1	0.056899424	1	0.097685666	0.722016938	0.028759473	5.52E-30	0.330182287	1.35E-51	0.31198219	1.12E-46		
ENSG00000184163.3	C1QTF12	935	0.004339553	0.059031507	0.01697861	1	0.123619172	0.971573052	0.059605593	0.795960145	0.027763009	0.684228446	0.27003154	0.000399295	0.252410656	0.001009205
ENSG00000160087.20	UBE2J2	4052	-0.018781721	1	-0.005616432	1	0.06902966	1	0.041120288	1	0.118911555	1.34E-05	0.359863102	6.77E-43	0.40201098	6.95E-54
ENSG00000162572.21	SCNN1D	4334	-0.072212678	1	-0.244865516	0.815898111	-0.181899086	0.825135208	-0.206986633	0.693142693	0.172373458	0.028499787	-0.37164395	6.50E-07	-0.34198392	5.80E-06
ENSG00000131584.19	ACAP3	3785	0.127844752	1	0.12791254	1	0.016340476	1	0.169561477	0.52509675	-0.195760833	1.71E-07	-0.433311159	1.06E-32	-0.523980776	3.23E-47
ENSG00000169972.12	PUS1L	1230	0.073770166	1	0.05698265	1	0.092046079	1	0.062409185	0.877004056	0.184795617	3.34E-07	0.094698149	0.008115119	-0.0585175	0.124565207
ENSG00000127054.20	INTS11	4763	-0.008556898	1	0.00764043	1	-0.014888971	1	-0.010872238	1	5.18E-09	-0.137611785	2.15E-15	-0.142479203	1.37E-16	
ENSG00000224051.7	CPTP	2406	0.015653893	1	0.053973742	1	0.047680865	1	0.024181163	1	0.027714475	0.191445222	-0.004764667	1	-0.167596923	5.95E-15
ENSG00000169962.5	TAS1R3	3376	0.013513716	0.941888124	0.05216788	0.992719658	0.107758312	1	-0.176170461	0.856267365	0.341631389	0.003406764	0.17222661	0.1300722	-0.061516795	0.653158608
ENSG00000107404.20	DVL1	3522	-0.044768884	1	-0.03015666	1	-0.050538025	1	0.051143906	1	0.107387432	7.55E-05	0.012170496	0.797610809	-0.220666329	2.76E-17
ENSG00000162576.16	MXRA8	3257	0.191313096	1	0.219490333	1	0.118577359	1	0.100133923	1	0.453824966	2.56E-17	-0.058839373	0.385826016	-0.429853477	1.31E-15
ENSG00000175756.13	AURKAIP1	1520	-0.028795706	1	-0.037998442	1	0.087237629	1	-0.052937789	1	0.141171957	3.74E-05	0.245885007	1.31E-13	0.150746674	7.28E-06
ENSG00000221978.12	CCNL2	4211	0.028521234	1	-0.005812613	1	-0.015013085	1	0.035759196	1	-0.118972028	6.80E-06	-0.32796888	5.25E-38	-0.323340944	1.81E-37
ENSG00000242485.6	MRLP20	1324	-0.07393538	1	-0.042639929	1	0.091903192	1	-0.117280624	0.754724517	0.157619322	8.45E-05	0.28739454	1.32E-08	0.155903354	0.000148887
ENSG00000235098.8	ANKRD65	2831	0.375416773	0.560677801	0.437437002	0.633271303	0.424234269	0.842296769	0.280125353	0.837926079	0.341581005	0.0002452026	0.56395942	0.574281575	0.025426855	0.749726248
ENSG00000205116.3	TMEM88B	390	0.1478490154	0.384576027	-0.032535513	1	0.185967779	1	-0.11236015	1	0.1037645284	0.021560888	0.298207391	0.342404805	0.29319792	0.394244566
ENSG00000179403.12	VWA1	5140	0.112084592	1	-0.12164659	1	-0.160268467	1	0.016458299	1	0.970974507	1.69E-22	-0.021517027	0.93781096	-0.217782531	0.038846874
ENSG00000171456.10	ATAN2C	2421	0.762421664	0.219490314	0.118577357	0.923084002	0.792327692	0.792327662	1.132776602	1.112776602	0.792327662	0.792327662	0.792327662	0.792327662	0.792327662	0.792327662

## Pathway analysis

Go to [Metascape](#) and enter the top 500 sorted gene list. After customizing your analysis, you can get their pathway analysis result.







## Future works

To further our understanding of complex signaling pathways, we propose integrating data from both the myofibroblast experiment and the epithelial experiment, both of which involve treatments. By combining these datasets, we aim to identify novel ways to distinguish between cellular responses and gain insight into potential mechanisms underlying these responses. Specifically, I plan to leverage this combined dataset to model responses from induced pluripotent stem cell (iPSC) lines, which could provide a powerful tool for studying complex signaling pathways. By doing so, we hope to enhance our understanding of the intricate interplay between various signaling pathways and pave the way for future research in this field.

## References

1. D'Alessio, S., et al., Revisiting fibrosis in inflammatory bowel disease: the gut thickens. *Nat Rev Gastroenterol Hepatol*, 2022. 19(3): p. 169-184.
2. Brooks, I.R., et al., Functional genomics and the future of iPSCs in disease modeling. *Stem Cell Reports*, 2022. 17(5): p. 1033-1047.
3. Workman, M.J., et al., Modeling Intestinal Epithelial Response to Interferon-gamma in Induced Pluripotent Stem Cell-Derived Human Intestinal Organoids. *Int J Mol Sci*, 2020. 22(1).
4. Ihara, S., Y. Hirata, and K. Koike, TGF-beta in inflammatory bowel disease: a key regulator of immune cells, epithelium, and the intestinal microbiota. *J Gastroenterol*,

2017. 52(7): p. 777-787.
5. Wang, Q., et al., Applications of human organoids in the personalized treatment for digestive diseases. *Signal Transduct Target Ther*, 2022. 7(1): p. 336.
  6. Carcamo-Orive, I., et al., Analysis of Transcriptional Variability in a Large Human iPSC Library Reveals Genetic and Non-genetic Determinants of Heterogeneity. *Cell Stem Cell*, 2017. 20(4): p. 518-532 e9.
  7. Anders, S. and W. Huber, Differential expression analysis for sequence count data. *Genome Biol*, 2010. 11(10): p. R106.