Project 2: Transcriptional Profile of Mammalian Cardiac Regeneration with mRNA-Seq

Mary T. Yohannes, Eetu Eklund, Salam Al-Abdullatif, Evie Wan

3/11/2020

Introduction

Shortly after birth, mammalian cardiomyocytes exit the cell cycle and hypertrophy of existing myocytes controls the remaining heart growth process. Prior to the first week of life, neonatal mice can regenerate their hearts in response to injuries. However, this ability is lost after the first week. There is evidence suggesting that adult cardiomyocytes with terminal differentiation still hold some limited capacity to self-renew as a result of injury. Nonetheless, this ability to repair itself is insufficient to replace the loss of functional myocardium. It has been shown that neonatal mice can fully recover and regenerate their heart after resection of the left ventricular apex. O'Meara et al. profiled gene expression during the process of mouse-cardiac-myocyte differentiation to compare the transcriptional signature to a cardiac-myocyte-explant model where the myocytes completely lose the fully differentiated phenotype. The study's objective was to analyze transcriptional data in order to determine and confirm potential regulators of heart regeneration. Additionally, the study aimed to identify whether myocytes (muscle cells) played a role in the regeneration process through the reversion of transcriptional phenotype into a less differentiated state. Our goal in this project was to complete a similar analysis on RNA-seq data from mice samples using tools such as `FASTQC`, `TopHat`, `Samtools`, `RSeQC`, `Cufflinks`, `Cuffdiff`, and `DAVID` to process, analyze and understand the reason behind cardiac regeneration of neonatal at early stages of development and its loss at later stages.

In a single run, large quantities of sequences are generated by modern high throughput sequencers. Prior to analyzing these sequences quality control (QC) checks are performed to affirm the absence of problems and biases in the data which can impact the succeeding results. Problems and biases that arise due to defects in the sequencer and/or starting materials can be identified by using `FastQC`. In this project, `FastQC` was run in command line mode to extract quality measures. Moreover, `TopHat` was used to align the sample files (`P0_1_1.fastq and P0_1_2.fastq`) to a reference sequence (`mm9.fa`, mouse genome) and `RSeQC` package was used for different quality control metrics on the alignment file created by `Tophat`. `Cufflinks` was used for differential expression analysis of the aligned reads created by `TopHat` and `Cuffdiff` was used to find any significant changes in the transcript expression between the two neonatal (P0) samples and the two adult (Ad) samples. This helped us determine differentially expressed genes between adult and young mice which could be further studied as potential markers of cardiac regeneration.

Data

All the samples used in this study except for one were downloaded and processed prior to the start of the project. The one remaining sample (`GSM1570702 (vP0_1)`) was downloaded from NCBI GEO Series `GSE64403`. The sample was stored in a file named `SRR1727914.sra`, short read archive (SRA) format, and had a size of 1.1 GB.

Data extraction:

Downloaded the file `SRR1727914.sra` into the SCC server and renamed the SRA file before converting to FASTQ file:

```
wget https://sra-downloadb.be-md.ncbi.nlm.nih.gov/sos1/sra-pub-
run-5/SRR1727914/SRR1727914.1

mv SRR1727914.1 P0_1.sra
```

Made SRA tools available and extracted the SRA formatted file to FASTQ file using `qsub`:

```
module load sratoolkit

run_extract.qsub file content:
     fastq-dump --split-files
     /projectnb/bf528/users/group_5/project_2/samples/P0_1.sra -
     O /projectnb/bf528/users/group_5/project_2/samples

qsub run_extract.qsub
```

Quality Control:

After the two FastQ files were extracted, they were processed, and the quality measures were obtained using the `FastQC` package available on SCC:

```
module load fastqc

fastqc -o
/projectnb/bf528/users/group_5/project_2/samples/fastqc_output
/projectnb/bf528/users/group_5/project_2/samples/P0_1_1.fastq
/projectnb/bf528/users/group_5/project_2/samples/P0_1_2.fastq
```

Three `RSeQC` utilities and `Samtools Flagstat` were used as quality control metrics. The following programs were used with the `TopHat` output file (`accepted_hits.bam`) as an input. `Samtools Flagstat` was used on this alignment file to count the number of alignments of each FLAG type (pass or fail alignment). `geneBody_Coverage.py` was used to create a line graph that shows the RNA-Seq coverage over a gene body (this program required an additional input file: `mm9.bed`). `inner_distance.py` was used to create a graph that shows the density of mRNA insert sizes. `bam_stat.py` was used to determine the number of uniquely mapped genes based on mapping quality, which is the probability that a read is misplaced. `inner_distance.py` and `bam_stat.py` were run locally and were done quickly whereas `geneBody_Coverage.py` was run on the cluster and took about 30 minutes to complete.

Methods

Two FASTQ files (`P0_1_1.fastq, P0_1_2.fastq`) were aligned against the mouse reference genome mm9 using `TopHat`.

```
tophat --no-novel-juncs -G
/project/bf528/project_2/reference/annot/mm9.gtf --segment-length=20 -
-segment-mismatches=1 -r 200 -p 16 -o P0_1_tophat
/project/bf528/project_2/reference/mm9
/projectnb/bf528/users/group_5/project_2/samples/P0_1_1.fastq
/projectnb/bf528/users/group_5/project_2/samples/P0_1_2.fastq
```

Option descriptions:

--no-novel-juncs = only looks for the reads across junctions specified in the .gtf file
-G = used with .gtf/gff3 annotation file to extract the transcript sequences (based on annotation file) and uses bowtie to align the reads to a virtual transcriptome. The reads that did not map to the transcriptome are converted to genomic mappings and then merged with the novel mappings and junctions of the tophat output file.
--segment-length = each read is cut up into segments of at least this length
--segment-mismatches = allows up to these many mismatches in each segment alignment
-r = expected inner distance between the read pairs
-p = how many threads used to align reads
-o = directory name of outputs
Last two inputs are the paired FASTQ files in order

This took about an hour to run and produced a BAM file that included all the original reads and any alignments discovered by `TopHat`. `Samtools Flagstat`, `geneBody_Coverage.py`, `inner_distance.py`, and `bam_stat.py` were run on this alignment file as described above in the data section.

`Cufflinks` was run on the `accepted_hits.bam` file for differential expression analysis. This program output a `genes.fpkm_tracking` file which contains the estimated gene-level expression values in FPKM. FPKM (fragments per kilobase of exon model per million reads mapped) is a normalized form of the estimated gene expression based on the RNA-seq data. `Cufflinks` ran on the cluster and took about an hour to complete.

```
cufflinks --compatible-hits-norm -G
/project/bf528/project_2/reference/annot/mm9.gtf -b
/project/bf528/project_2/reference/mm9.fa -u -o P0_1_cufflinks -p 16
/projectnb/bf528/users/group_5/project_2/P0_1_tophat/accepted_hits.bam
```

Option descriptions:

--compatible-hits-norm = only counts fragments compatible with reference transcript towards number of fragments used in FPKM
-G = used with .gtf/gff3 annotation file to extract the transcript sequences (based on annotation file) and uses bowtie to align the reads to a virtual transcriptome. The reads that did not map to the transcriptome are converted to genomic mappings and then merged with the novel mappings and junctions of the tophat output file.
-b = runs bias detection and correction algorithm to improve the accuracy of transcript abundance estimates
-u = cufflinks estimates to more accurately weigh reads that map to multiple locations
-o = output file directory name
-p = number of threads used to align reads
Last input is the accepted_hits.bam alignment file

`Cuffdiff` was run on all four samples that were involved in the study (`P0_1, P0_2, Ad_1, Ad_2`) to determine significant changes in transcript expression. This output a file known as `genes.fpkm_tracking` which included the differential expression statistics comparing the two conditions (`P0` and `Ad`). `Cuffdiff` was ran on the cluster and took about 3 hours to complete.

```
P0BAM =
/projectnb/bf528/users/group_5/project_2/P0_1_tophat/accepted_hits.bam
SAMPLEDIR = /project/bf528/project_2/data/samples/
P0REPS = $P0BAM,$SAMPLEDIR/P0_2/accepted_hits.bam
ADREPS =
$SAMPLEDIR/Ad_1/accepted_hits.bam,$SAMPLEDIR/Ad_2/accepted_hits.bam
LABEL = "P0,Ad"
OUTDIR = cuffdiff_out
FASTA = /project/bf528/project_2/reference/mm9.fa

cuffdiff -p 16 -L $LABEL -u -b $FASTA -o $OUTDIR
$SAMPLEDIR/merged_asm/merged.gtf $P0REPS $ADREPS
```

Option descriptions:

- -p = number of threads used to align reads
- -L = specify labels for each sample
- -u = cufflinks estimates to more accurately weigh reads that map to multiple locations
- -b = runs bias detection and correction algorithm to improve the accuracy of transcript abundance estimates
- -o = sets name of output directory

The `Cuffdiff` differential expression of the day 0 postnatal mice vs the adult mice was loaded into R as a data frame and sorted by increasing q-value. The data was then subset into up-regulated and down-regulated genes by their log2 fold change and genes were further subset by significance as determined by the `Cuffdiff` output. The up-regulated and down-regulated genes were then written into a text file and uploaded to `DAVID` for functional annotation clustering. The gene ontology terms `GOTERM_BP_FAT`, `GOTERM_MF_FAT` and `GOTERM_CC_`FAT were selected for clustering and the top annotations were selected for comparison.

The FPKM values of genes specific to the most prominent GO terms discovered in the analysis were plotted using `genes.fpkm_tracking` tables. There were eight samples in total and seven of them were provided in the project directories (`Ad_1, Ad_2, P0_2, P4_1, P4_2, P7_1, P7_2`). The FPKM table for sample P0 was obtained from the `cuffdiff` output and the genes of interest and their FPKM values were selected from `genes.fpkm_tracking`. These values were then plotted using `ggplot2` to recreate the plots shown in the paper.

`DAVID` analysis results were compared based on common GO terms. The `DAVID` analysis results of the paper were found in the supplemental materials and compared to the `DAVID` output generated by the analyst. The updated csv tables were generated (`DAVID_up_top5_updated.csv` and `DAVID_up_down5_updated.csv`) and they contained the GO terms that overlapped with the paper's result. Lastly, a clustered heatmap was generated with the most differentially expressed genes between P0 and Ad.

To generate a heatmap, first 1000 most differentially expressed genes were selected based on FPKM values. The most positive and the most negative FPKM values were selected to represent the most up-regulated and down-regulated genes. Duplicated genes were deleted, and only unique values were kept.

Results

The extraction process of the SRA formatted file resulted in two FastQ files: forward and reverse reads. The format of the files was inspected by accessing the first few lines of both files and ensuring that the texts in the header fields of both files were the same. Additionally, the outputs from the FastQ tool included a number of html and image files that could be downloaded and inspected.

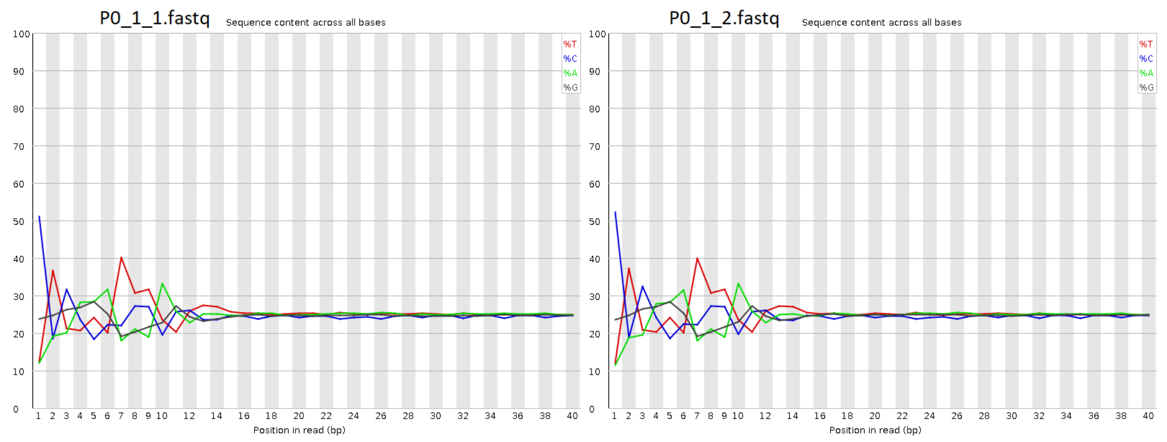

Figure 1. "Per base sequence content" FastQ output for the two fastq files

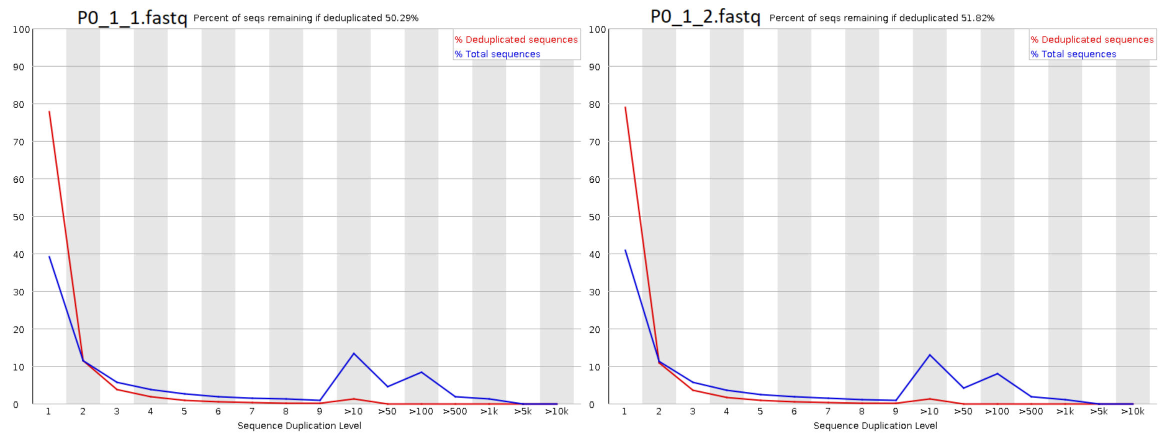

Figure 2. "Sequence Duplication Levels" FastQ output for the two fastq files

Output 1: All 49,706,999 reads passed quality controls and all 65.32% of reads were properly aligned. Below is the output from Samtools Flagstat for reads that pass/fail alignment.

```
samtools flagstat P0_1_tophat/accepted_hits.bam
```

```
49706999 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
49706999 + 0 mapped (100.00%:-nan%)
49706999 + 0 paired in sequencing
25089027 + 0 read1
24617972 + 0 read2
32466938 + 0 properly paired (65.32%:-nan%)
47843662 + 0 with itself and mate mapped
1863337 + 0 singletons (3.75%:-nan%)
5098744 + 0 with mate mapped to a different chr
704916 + 0 with mate mapped to a different chr (mapQ>=5)
```

Output 2: There were 49,706,999 uniquely mapped reads and none of them failed the quality control.

```
bam_stat.py -i ../P0_1_tophat/accepted_hits.bam
```

```
#==================================================
#All numbers are READ count
#==================================================
Total records:                  49706999
QC failed:                  0
Optical/PCR duplicate:            0
Non primary hits                8317665
Unmapped reads:                 0
mapq < mapq_cut (non-unique):       2899954
mapq >= mapq_cut (unique):        38489380
Read-1:                     19409941
Read-2:                     19079439
Reads map to '+':               19236824
Reads map to '-':               19252556
Non-splice reads:               33099839
Splice reads:                   5389541
Reads mapped in proper pairs:       27972916
Proper-paired reads map to different chrom:4
```

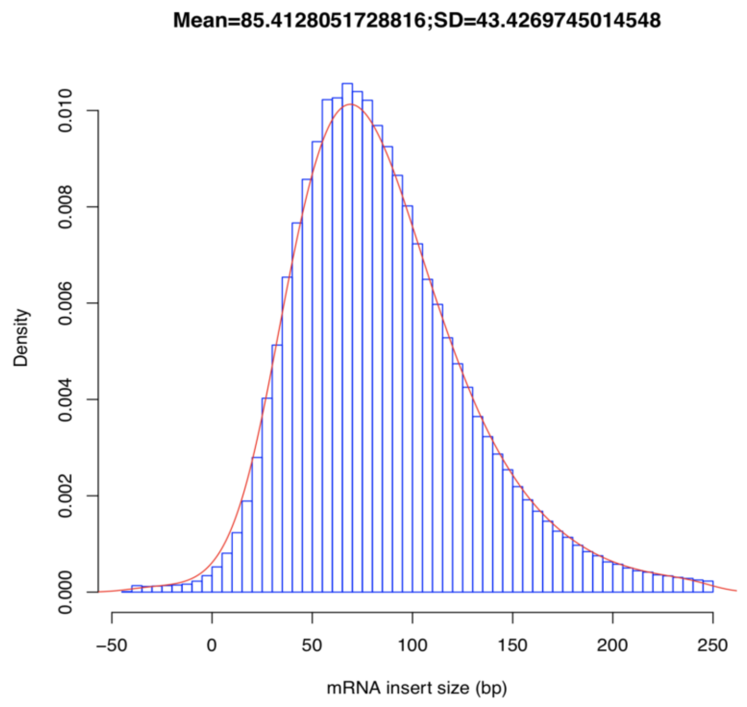

Figure 3. Density of the distances between reads generated from `inner_distance.py`

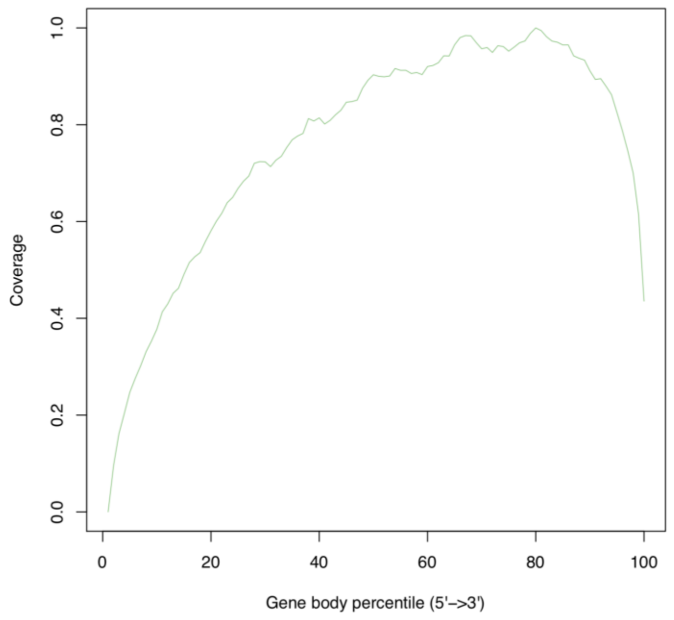

Figure 4. RNA-Seq reads coverage over the gene body generated from `geneBody_coverage.py`

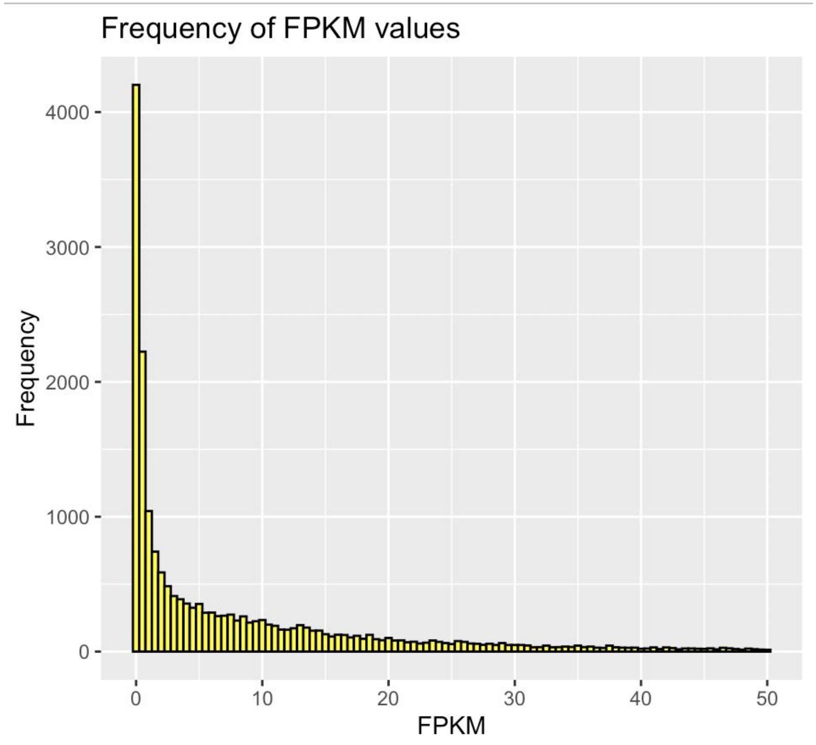

Figure 5. FPKM frequency generated from `genes.fpkm_tracking`

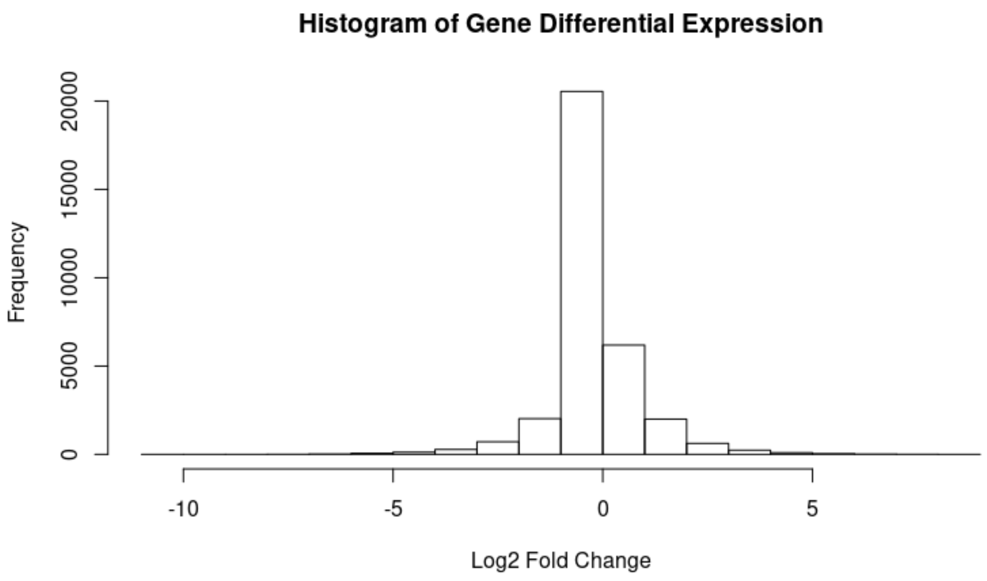

Figure 6. The frequency of log2 fold change across the list of differentially expressed genes

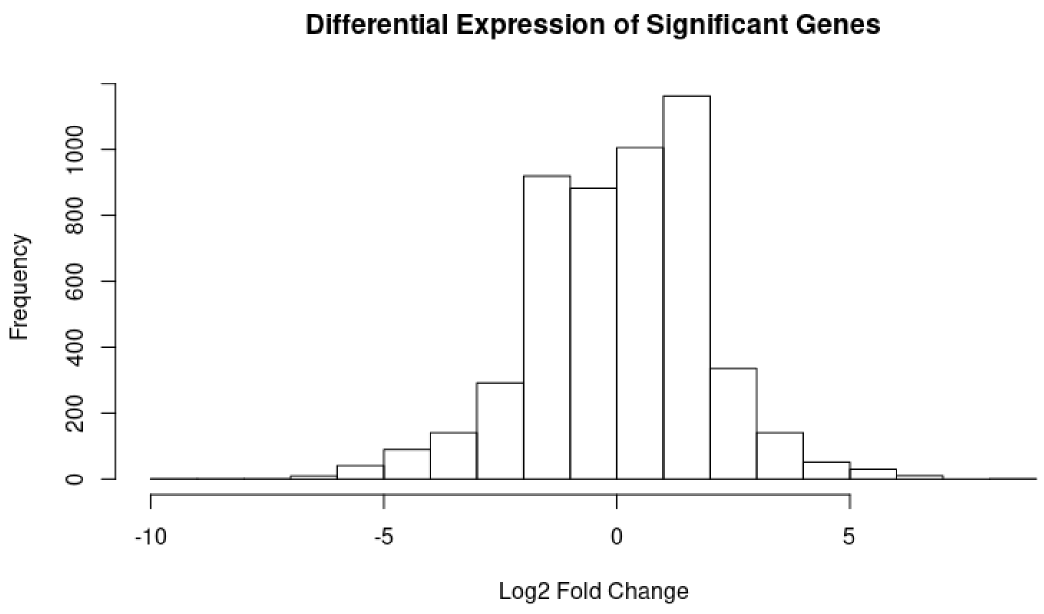

Figure 7. The distribution of log2 fold change among only significant genes as determined by `cuffdiff`

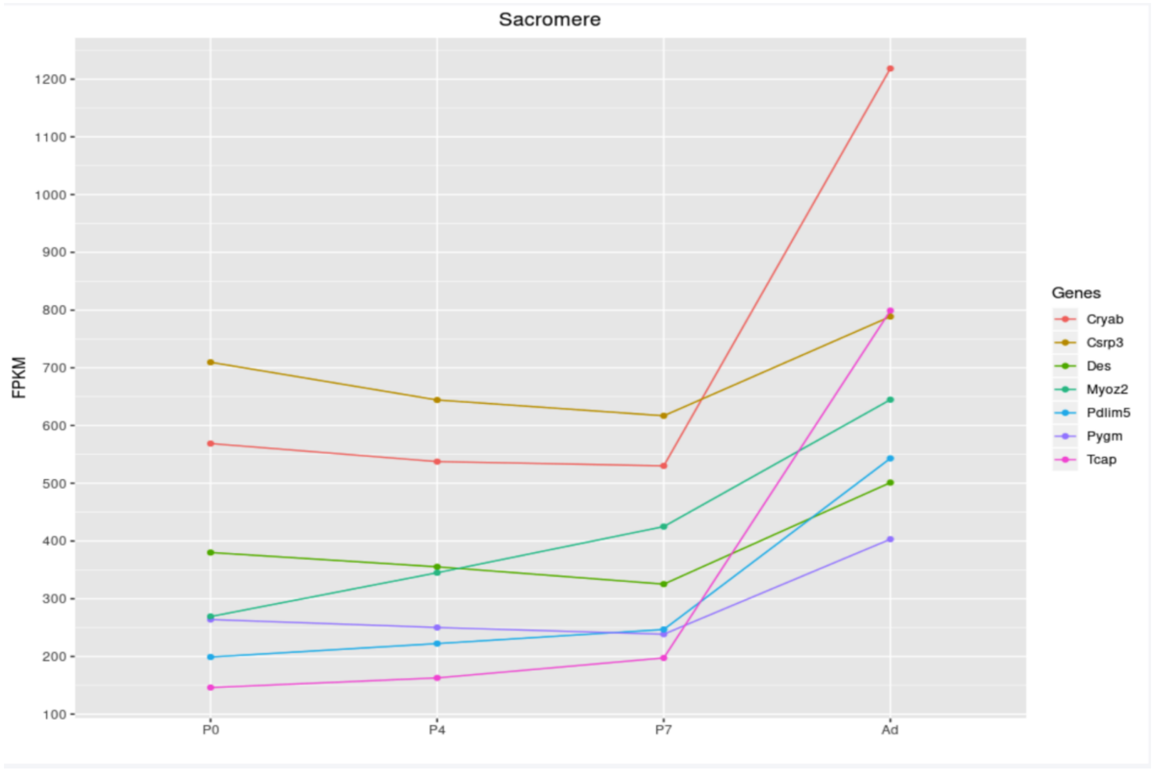

Figure 8. FPKM values of representative Sarcomere genes were significantly differentially expressed during in-vivo maturation.

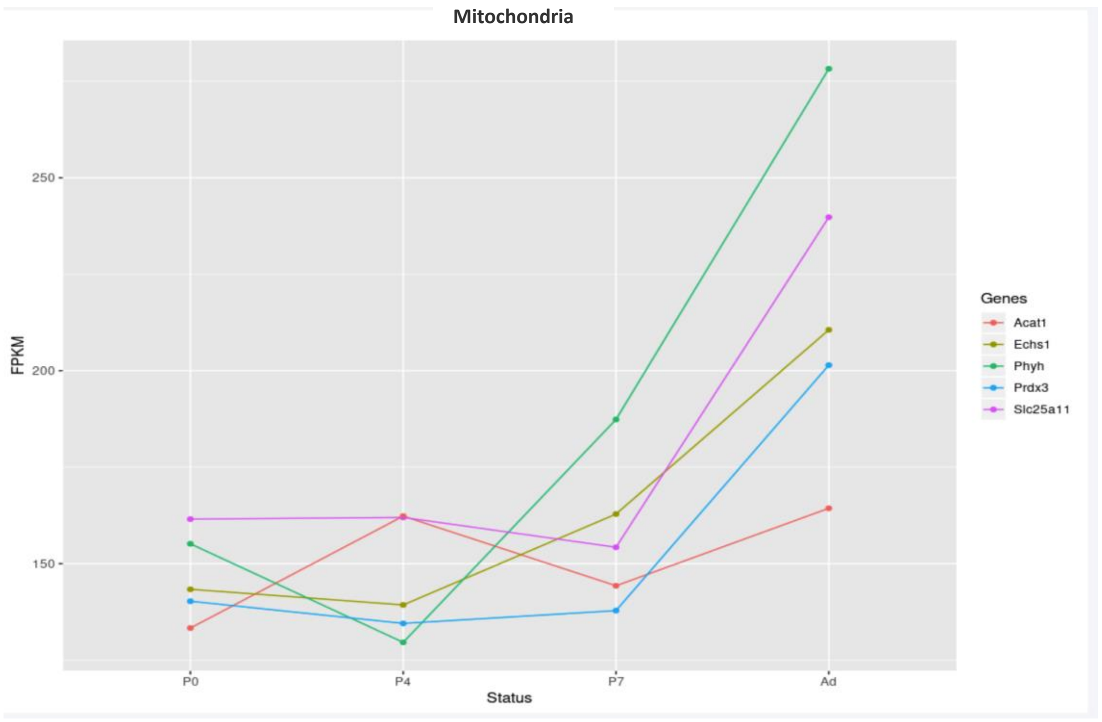

Figure 9. FPKM values of representative Mitochondria genes were significantly differentially expressed during in-vivo maturation

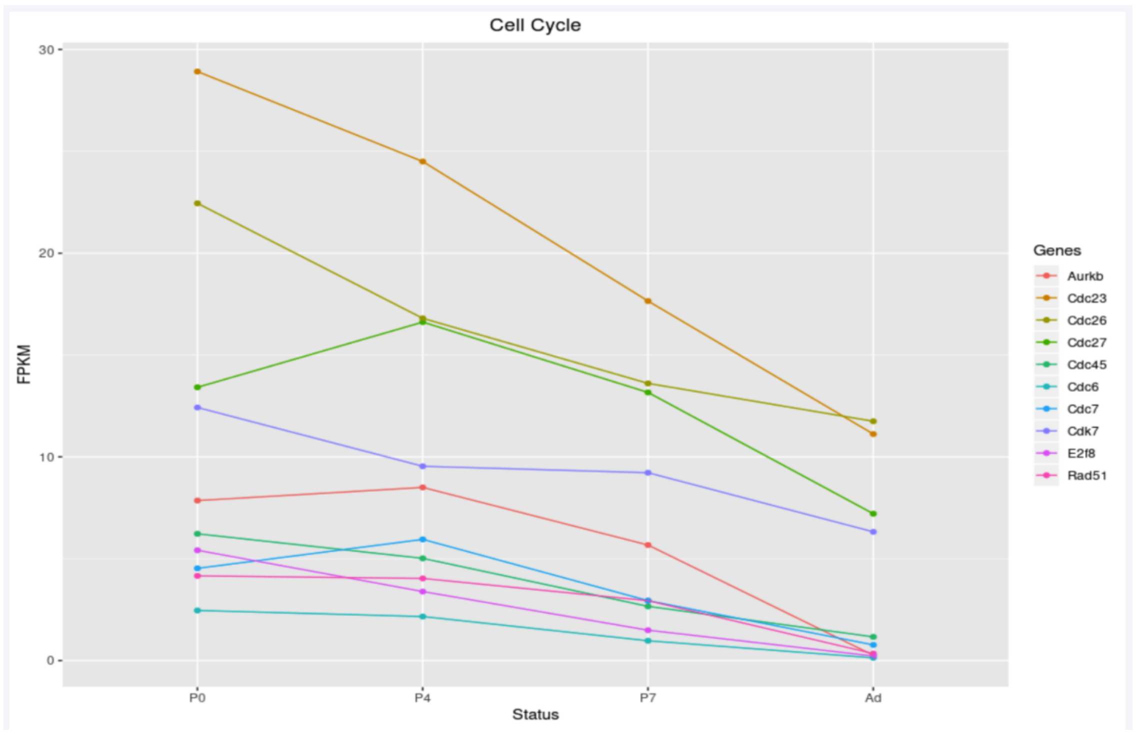

Figure 10. FPKM values of representative Cell Cycle genes significantly differentially expressed during in-vivo maturation

After comparing the GO terms obtained from DAVID analysis with the paper's results, the following GO terms were unique to our analysis and were not found in the paper.

Down-regulated GO terms that were not found in the paper's result:

GO:0000278~mitotic cell cycle
GO:1903047~mitotic cell cycle process
GO:0019219~regulation of nucleobase-containing compound metabolic process
In Cluster3, only the last GO term was found in the paper's result
GO:0051276~chromosome organization
GO:1902589~single-organism organelle organization

Up-regulated GO terms were not found in the paper's results:

Cluster1: GO:0005739~mitochondrion
Cluster 2: GO:0042278~purine nucleoside metabolic process
GO:0046128~purine ribonucleoside metabolic process
Cluster3: GO:0098798~mitochondrial protein complex
GO:0098800~inner mitochondrial membrane protein complex
GO:1990204~oxidoreductase complex
GO:0070469~respiratory chain (this cluster was not found in paper's results)
Cluster4: GO:0006082~organic acid metabolic process
GO:0019752~carboxylic acid metabolic process
GO:0043436~oxoacid metabolic process
GO:0032787~monocarboxylic acid metabolic process
GO:0044255~cellular lipid metabolic process (this cluster was not found in paper's results)
Cluster5: GO:0043230~extracellular organelle
GO:0070062~extracellular exosome
GO:1903561~extracellular vesicle

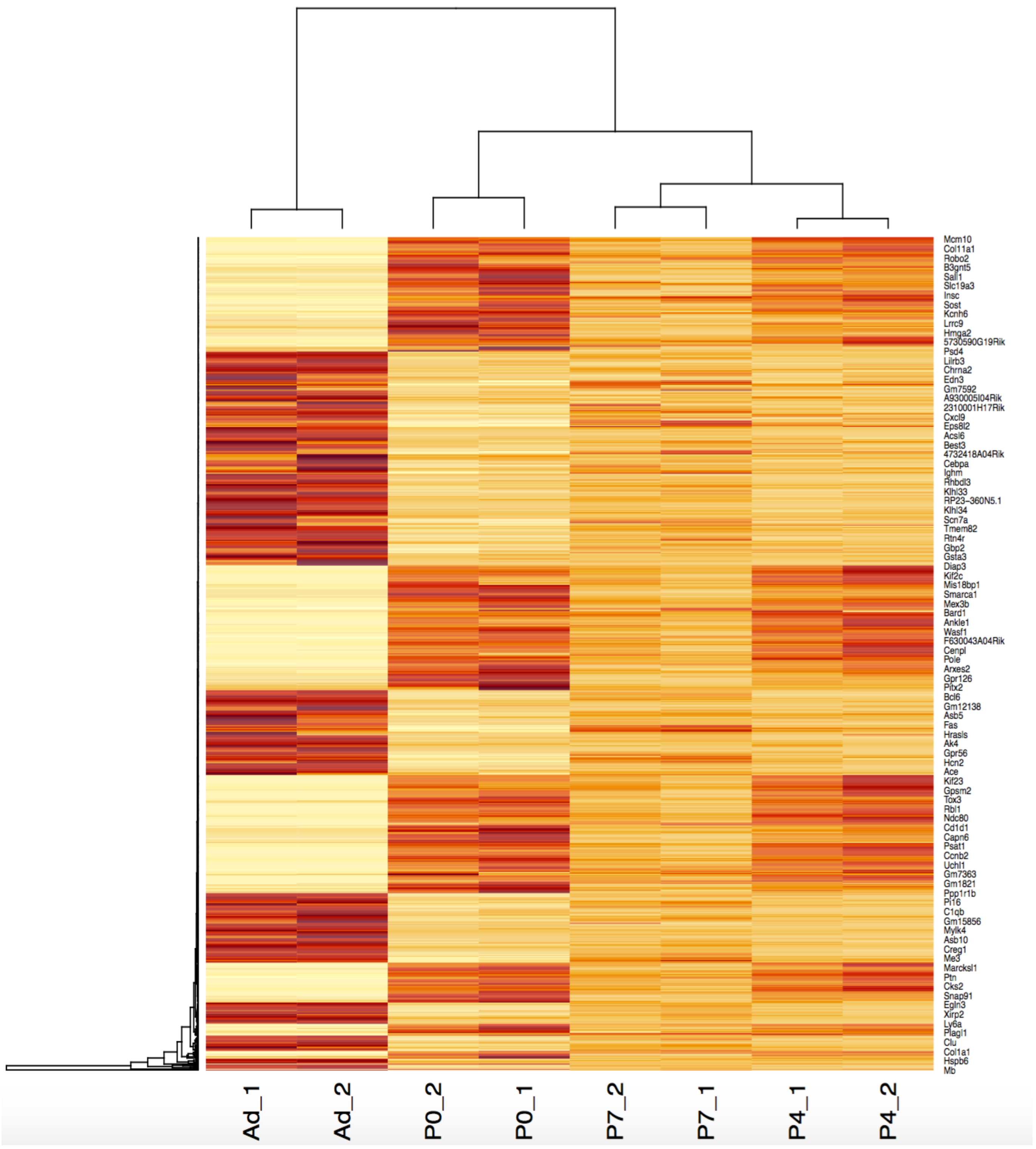

Figure 11. Clustered heatmap showing eight samples with the most differentially expressed genes between P0 and Ad samples

Discussion

According to the metrics read quality obtained from the FastQC run, in both files, a total of 21,577,562 sequences with length of 40 were processed. None of these sequences were overrepresented or flagged for poor quality, and had an overall %GC of 49. Both files failed “Per Base Sequence Content” (Figure 1) and showed warning for “Sequence Duplication Levels” (Figure 2) while passing the rest of the quality matrices.

“Per Base Sequence Content” shows the relative amount of A, T, C and G at each position across all reads in the genome. It failed in both files because there is >20% difference between A and T, and G and C towards the beginning read positions as shown in Figure 1; a difference in proportion between G and C is seen at the first position while a difference between A and T is seen at the 7th position. At the 13th position, in both files, the lines start to parallel with each other.

“Sequence Duplication Levels” shows the relative number of sequences with different degrees of duplication. According to Figure 2, the percent of sequences remaining if duplicated is 50.29% for `P0_1_1.fastq` and 51.82% for `P0_1_2.fastq`. In both plots, there is a fluctuation of duplication levels across reads with approximately 13-14% of the reads are duplicated >10 times and 9% are duplicated between >100 times. The warning indicates that non-unique sequences make up >20% of the total.

The quality control metrics `Samtools Flagstat` and `RSeQC` program `bam_stat.py` determined that all our reads pass quality control and that 65.32% of reads were properly aligned (Output 1, Output 2). This allows us to be more confident in our results by telling us there weren't any major problems when collecting the data. The data is acceptable, so it’s ready for further analysis. The inner distance, or the distance between the reads, was centered around 85bp with a skew to the right (due to larger introns) and a standard deviation of 43bp (Figure 3). Most RNA-seq reads tend to be in the second half of the gene (Figure 4). RNA-seq Coverage decreases towards 3’ and 5’ ends due to the way sequencing technology works. Overall, quality control metrics determined no sources of error or contamination in the data.

For further analysis, FPKM values (Fragments per Kilobase of transcript per Million reads sequenced) were calculated using the `Cufflinks` program. This gives us a normalized estimation of the RNA-seq gene expression data. FPKM values =0 and <50 were filtered out. Median FPKM was 2.94 with a mean of 7.88. FPKM values are mostly around 0-2, but it skews heavily to the right (Figure 5). This means that typically there are 3 fragments per kilobase of the transcripts for every million reads sequenced. This normalization removes potential sources of error in the analysis that are due to factors like gene size and composition.

The `cuffdiff` output produces a list of differentially expressed genes with their log2 fold change, q-value, p-value, and a boolean which indicates significance. The gene list was subset to contain only significant genes, where we found 5188 significant genes. From the list of significant genes, the up-regulated and down-regulated genes were identified by their log2 fold change. We found 2757 up-regulated genes and 2431 down-regulated genes. From the two subset gene lists, two files were created: one containing only the significant up-regulated genes and the other containing only significant down-regulated genes. These gene lists were then uploaded to `DAVID` for functional annotation clustering, and the GO terms relating to fat were selected for comparison. The top GO terms for the top 5 enriched clusters were selected for further analysis.

Plots of FPKM values of genes representative of Sarcomere, Mitochondria, and Cell Cycle were recreated and the results obtained align with the original study. In Figure 8, representative Sarcomere genes show an increase in FPKM values during in-vivo maturation process. Even though some genes, such as Csrp3 and Pygm, show a decrease in FPKM values from P0 to P7, all Sarcomere genes had a steep increase in FPKM values from P7 to Ad. This indicates that the representative Sarcomere genes were primarily up-regulated during the development from postnatal to Adult.

Figure 9 shows general trend of upregulation of mitochondrial genes during in-vivo maturation. Similar to representative Sarcomere genes, the biggest increase in FPKM values happened during P7 to Adult maturation stage.

It can be seen in Figure 10 that the FPKM values of Cell Cycle representative genes show a decreasing trend from P0 to Adult maturation. This pattern shows downregulation of Cell Cycle genes during in-vivo maturation. This further reinforces the study’s conclusion that differentially expressed genes during in-vivo differentiation showed primarily up-regulation of sarcomere and mitochondrial related genes and down-regulation of cell cycle genes.

Comparing `DAVID` analysis results to that of the original study, there are some GO terms that were unique to our analysis and absent in the study’s analysis results. For our project, the `DAVID` analysis was based on genes differentially expressed between just two samples - P0 and Ad. In the original study, `DAVID` analysis involved both in-vivo and in-vitro genes. The difference in GO terms can be validated due to the difference in analysis method. However, both the original paper and our analysis show that cell cycle genes were down-regulated while mitochondrial genes were up-regulated.

Lastly, the heatmap (Figure 11) shows the most differentially expressed genes between P0 and Ad in the eight samples. The method used to produce the heatmap might have introduced errors because it is unclear which value was kept after using dedupe functions in R. If a gene has different FPKM values in a sample, it might be more reasonable to take the average instead of randomly choosing one of the values. Nonetheless, we can see clearly four clusters in the heatmap for P0, P4, P7, and Ad; P0 and Ad samples show the most distinguished patterns - genes that are up-regulated in Ad (shown in red) are down-regulated in P0 (shown in yellow), and vice versa. This further supports that genes are differentially expressed in the prenatal and adult stage of maturation and thus leading to loss of regeneration function of the heart.

Conclusion

Differentially expressed genes during in-vivo differentiation primarily showed down-regulation of cell cycle genes and up-regulation of sarcomere and mitochondrial related genes. This was further supported by DAVID analysis of GO terms and the heatmap provided additional visualization for differential expression. The most prominent expression pattern was between P0 and Ad samples. This reflects the loss of cardiac regeneration function in adult mice since differential gene expressions lead to changes in cell structure and functions. Our project was able to reproduce the results obtained in the original study and demonstrate that genes are differentially expressed during prenatal and adult stage of mice maturation leading to changes in the ability to regenerate hearts in response to injury.

References

Andrews, S. (2010). FastQC: A Quality Control Tool for High Throughput Sequence Data [Online]. Available online at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Huang DW, Sherman BT, Lempicki RA. Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources. Nature Protoc. 2009;4(1):44-57. [PMID: 19131956]

Huang DW, Sherman BT, Lempicki RA. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. Nucleic Acids Res. 2009;37(1):1-13. [PMID: 19033363]

Li H.*, Handsaker B.*, Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. *Bioinformatics*, 25, 2078-9. [PMID: 19505943]

Liguo Wang, Shengqin Wang, Wei Li, RSeQC: quality control of RNA-seq experiments. *Bioinformatics*, Volume 28, Issue 16, 15 August 2012, Pages 2184–2185, https://doi.org/10.1093/bioinformatics/bts356

O'Meara, C. C., Wamstad, J. A., Gladstone, R. A., Fomovsky, G. M., Butty, V. L., Shrikumar, A., … Lee, R. T. (2015). Transcriptional Reversion of Cardiac Myocyte Fate During Mammalian Cardiac Regeneration. *Circulation Research*, *116*(5), 804–815. doi: 10.1161/circresaha.116.304269

Roberts, A., Trapnell, C., Donaghey, J. *et al.* Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome Biol* 12, R22 (2011). https://doi.org/10.1186/gb-2011-12-3-r22

Trapnell, C., Williams, B., Pertea, G. *et al.* Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol* 28, 511–515 (2010). https://doi.org/10.1038/nbt.1621

Trapnell, C., Hendrickson, D., Sauvageau, M. *et al.* Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat Biotechnol* 31, 46–53 (2013). https://doi.org/10.1038/nbt.2450

Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics* doi:10.1093/bioinformatics/btp120

Wang, L., Wang, S., & Li, W. (2012). RSeQC: quality control of RNA-seq experiments. *Bioinformatics* (Oxford, England), 28(16), 2184–2185. http://doi.org/10.1093/bioinformatics/bts356

Project code

output.geneBodyCoverage.R (programmer)

```
accepted_hits <-
c(0.0,0.09534506996799529,0.16189770771969983,0.2038682722021854,0.2468802098
1740155,0.2756490281304558,0.301600707787393,0.33082842295334364,0.3527885658
3320084,0.3774367580314821,0.41313153416380805,0.42956838507664286,0.45162140
930923034,0.4620870912236567,0.49066306266027937,0.5152097150746435,0.5270835
727520745,0.5357947416560273,0.5594283231452459,0.5808484947710947,0.60071487
15639803,0.6170210588789063,0.6388867461458174,0.6497947951808746,0.668393892
5000669,0.683192725658395,0.6940000220396431,0.7201941377698086,0.72372677769
39922,0.7233284898594028,0.7133634226935908,0.7266800898587731,0.734845777597
9623,0.7531142802718432,0.7689379568306364,0.7766652130997341,0.7820822425021
922,0.8127173463010395,0.8077458325396438,0.8142428044501188,0.80151333634541
79,0.8090225575746317,0.8204886818561787,0.8300649067486962,0.846478930888402
3,0.8478288590234234,0.8508097207419804,0.8755885765381703,0.8920207046703578
,0.9032577740905106,0.8999935455331154,0.8990489894036545,0.9006822843775139,
0.9161793019415351,0.9124585379215672,0.9125474836237581,0.905942674888503,0.
9079758319571676,0.9036151311594899,0.9202251506960591,0.9224314763884581,0.9
281759519157959,0.9424568534631363,0.9420577784984391,0.9651246263099813,0.97
99423505908986,0.9842392938498377,0.9837473375324101,0.9696175649657834,0.956
9786168234892,0.9595344282837888,0.9493647072899268,0.9632488952628936,0.9615
274417169511,0.9520354397459774,0.9602845632766023,0.9692318712129203,0.97302
58383329214,0.9887668662303867,1.0,0.9945766735566789,0.9817338587164427,0.97
2825120155411,0.9704597941812194,0.9648223683485538,0.9649719230690518,0.9423
498037684641,0.9373900969586867,0.933525288128976,0.9115470412566374,0.893596
5391463416,0.8952353440309563,0.8791936324322792,0.8619570573298343,0.8252209
080647777,0.7879101538209656,0.7459994112266793,0.6997610272992464,0.61443691
0734723,0.43548681635782305)

pdf("output.geneBodyCoverage.curves.pdf")
x=1:100
icolor =
colorRampPalette(c("#7fc97f","#beaed4","#fdc086","#ffff99","#386cb0","#f0027f
"))(1)
plot(x,accepted_hits,type='l',xlab="Gene body percentile (5'->3')",
ylab="Coverage",lwd=0.8,col=icolor[1])
dev.off()
```

output.inner_distance_plot.R (programmer)

```
out_file = 'output'
pdf('output.inner_distance_plot.pdf')
fragsize=rep(c(-247.5,-242.5,-237.5,-232.5,-227.5,-222.5,-217.5,-212.5,-
207.5,-202.5,-197.5,-192.5,-187.5,-182.5,-177.5,-172.5,-167.5,-162.5,-157.5,-
152.5,-147.5,-142.5,-137.5,-132.5,-127.5,-122.5,-117.5,-112.5,-107.5,-102.5,-
97.5,-92.5,-87.5,-82.5,-77.5,-72.5,-67.5,-62.5,-57.5,-52.5,-47.5,-42.5,-
37.5,-32.5,-27.5,-22.5,-17.5,-12.5,-7.5,-
2.5,2.5,7.5,12.5,17.5,22.5,27.5,32.5,37.5,42.5,47.5,52.5,57.5,62.5,67.5,72.5,
77.5,82.5,87.5,92.5,97.5,102.5,107.5,112.5,117.5,122.5,127.5,132.5,137.5,142.
5,147.5,152.5,157.5,162.5,167.5,172.5,177.5,182.5,187.5,192.5,197.5,202.5,207
.5,212.5,217.5,222.5,227.5,232.5,237.5,242.5,247.5),times=c(0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,49,631,549,5
91,635,692,778,1071,1614,2467,3810,5820,8914,13177,18980,24179,30826,36146,40
412,44101,48215,48389,49792,49007,48154,45683,43616,40804,37813,34090,30615,2
8166,24895,22351,20048,17184,15222,13515,11981,10325,9036,7915,6941,5967,5365
,4584,3964,3560,2956,2727,2362,2092,1954,1691,1592,1445,1344,1189,1081))
frag_sd = sd(fragsize)
frag_mean = mean(fragsize)
frag_median = median(fragsize)
write(x=c("Name","Mean","Median","sd"), sep="   ", file=stdout(),ncolumns=4)
write(c(out_file,frag_mean,frag_median,frag_sd),sep=" ",
file=stdout(),ncolumns=4)
hist(fragsize,probability=T,breaks=100,xlab="mRNA insert size
(bp)",main=paste(c("Mean=",frag_mean,";","SD=",frag_sd),collapse=""),border="
blue")
lines(density(fragsize,bw=10),col='red')
dev.off()
```

analysis.R (analyst)

```
# Read the file as tab separated values
gene_exp <- as.data.frame(read.csv("../cuffdiff_out/gene_exp.diff", sep =
"\t"))
gene_exp <- gene_exp[order(gene_exp$p_value, decreasing = FALSE),]

# Create a histogram
hist(gene_exp$log2.fold_change., breaks = 15, xlab = "Log2 Fold Change", main
= "Histogram of Gene Differential Expression")

sig_genes <- subset(gene_exp, significant == "yes")
hist(sig_genes$log2.fold_change., breaks = 15, xlab = "Log2 Fold Change",
main = "Differential Expression of Significant Genes")

sig_genes_up <- subset(sig_genes, log2.fold_change. > 0)
sig_genes_down <- subset(sig_genes, log2.fold_change. < 0)
cat("The number of up-regulated significant genes: ", dim(sig_genes_up)[1])
cat("The number of down-regulated significant genes: ",
dim(sig_genes_down)[1])
write.table(sig_genes_up$gene, "sig_genes_up.csv", row.names = FALSE,
col.names = FALSE, quote = FALSE)
write.table(sig_genes_down$gene, "sig_genes_down.csv", row.names = FALSE,
col.names = FALSE, quote = FALSE)
write.table(sig_genes_up$gene, "sig_genes_up.txt", row.names = FALSE,
col.names = FALSE, quote = FALSE)
write.table(sig_genes_down$gene, "sig_genes_down.txt", row.names = FALSE,
col.names = FALSE, quote = FALSE)
```

biologist.Rmd (Biologist)

```
library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
setwd("/projectnb/bf528/users/group_5/project_2/evie")
fpkm <- read.table('fpkm_matrix.csv', header=TRUE, sep="\t")
Gdiff <-
read.table("/projectnb/bf528/users/group_5/project_2/cuffdiff_out/gene_exp.di
ff", header=TRUE)
dav_down5 <- read.csv("DAVID_down_top5.csv")
dav_up5 <- read.csv("DAVID_up_top5.csv")
sig_down <- read.csv("sig_genes_down.csv")

#read fpkm tracking tables
P0_2 <-
read.csv("/project/bf528/project_2/data/samples/P0_2/genes.fpkm_tracking",
header=TRUE, sep="\t") #%>% rename(FPKM_P02 = FPKM)
P0_1 <-
read.csv("/projectnb/bf528/users/group_5/project_2/evie/genes.fpkm_tracking",
header=TRUE, sep="\t")
P4_1 <-
read.csv("/project/bf528/project_2/data/samples/P4_1/genes.fpkm_tracking",
header=TRUE, sep="\t")
P4_2 <-
read.csv("/project/bf528/project_2/data/samples/P4_2/genes.fpkm_tracking",
header=TRUE, sep="\t")
P7_1 <-
read.csv("/project/bf528/project_2/data/samples/P7_1/genes.fpkm_tracking",
header=TRUE, sep="\t")
P7_2 <-
read.csv("/project/bf528/project_2/data/samples/P7_2/genes.fpkm_tracking",
header=TRUE, sep="\t")

Ad1_ftable <-
read.csv("/project/bf528/project_2/data/samples/Ad_1/genes.fpkm_tracking",
header=TRUE, sep="\t")
Ad2_ftable <-
read.csv("/project/bf528/project_2/data/samples/Ad_2/genes.fpkm_tracking",
header=TRUE, sep="\t")
```

Sacromere

```
library(tibble)
#Sacromere Gene names
genes_sac <- c("Pdlim5", "Pygm", "Myoz2", "Des", "Csrp3", "Tcap", "Cryab")
genes_mit <- c("Mpc1", "Prdx3", "Acat1", "Echs1", "Slc25a11", "Phyh")
genes_cc <- c("Cdc7", "E2f8", "Cdk7", "Cdc26", "Cdc6", "Cdc27", "Bora",
"Cdc45", "Rad51", "Aurkb", "Cdc23")

#filter adult sacromere genes

sacromere_ad1 <- Ad1_ftable %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
sacromere_ad2 <- Ad2_ftable %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

ad_com <- cbind(sacromere_ad1, sacromere_ad2$FPKM)
ad_com$mean <- rowMeans(ad_com)
ad_mean <- ad_com %>% rownames_to_column("Genes")
ad_sacro <- data.frame("Genes" = ad_mean$Genes, "FPKM" = ad_mean$mean,
"Status" = c("Ad", "Ad", "Ad","Ad", "Ad", "Ad","Ad"))
#filter P0 sacromere genes
sacromere_p0 <- P0_1 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
sacromere_p02 <- P0_2 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
p0_com <- cbind(sacromere_p0, sacromere_p02$FPKM)

p0_com$mean <- rowMeans(p0_com)
p0_mean <- p0_com %>% rownames_to_column("Genes")

p0_sacro <- data.frame("Genes" = p0_mean$Genes, "FPKM" = p0_mean$FPKM,
"Status" = c("P0", "P0","P0", "P0","P0", "P0","P0"))
#filter P4
sacromere_p41 <- P4_1 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
sacromere_p42 <- P4_2 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p4_com <- cbind(sacromere_p41, sacromere_p42$FPKM)
p4_com$mean <- rowMeans(p4_com)
p4_mean <- p4_com %>% rownames_to_column("Genes")
p4_sacro <- data.frame("Genes" = p4_mean$Genes, "FPKM" = p4_mean$mean,
"Status" = c("P4", "P4", "P4", "P4","P4", "P4","P4"))
#filter P7
sacromere_p71 <- P7_1 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
sacromere_p72 <- P7_2 %>% filter(gene_short_name %in% genes_sac) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p7_com <- cbind(sacromere_p71, sacromere_p72$FPKM)
```

```r
p7_com$mean <- rowMeans(p7_com)
p7_mean <- p7_com %>% rownames_to_column("Genes")
p7_sacro <- data.frame("Genes" = p7_mean$Genes, "FPKM" = p7_mean$mean,
"Status" = c("P7", "P7", "P7", "P7","P7", "P7","P7"))

sac_fin <- bind_rows(p0_sacro, p4_sacro, p7_sacro, ad_sacro)
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to
character
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Mitochondria

```r
#filter P0 mitochondira genes

mit_p01 <- P0_1 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

mit_p02 <- P0_2 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p0mit_com <- cbind(mit_p01, mit_p02)
p0mit_com$mean <- rowMeans(p0mit_com)
p0mit_mean <- p0mit_com %>% rownames_to_column("Genes")
p0_mit <- data.frame("Genes" = p0mit_mean$Genes, "FPKM" = p0mit_mean$FPKM,
"Status" = c("P0","P0","P0","P0","P0"))
#filter P4 mitochondria

mit_p41 <- P4_1 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

mit_p42 <- P4_2 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p4mit_com <- cbind(mit_p41, mit_p42)
p4mit_com$mean <- rowMeans(p4mit_com)
p4mit_mean <- p4mit_com %>% rownames_to_column("Genes")

p4_mit <-  data.frame("Genes" = p4mit_mean$Genes, "FPKM" = p4mit_mean$FPKM,
"Status"=c("P4", "P4","P4", "P4","P4"))
```

```
#filter P7 mitochondria
mit_p71 <- P7_1 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

mit_p72 <- P7_2 %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p7mit_com <- cbind(mit_p71, mit_p72)
p7mit_com$mean <- rowMeans(p7mit_com)
p7mit_mean <- p7mit_com %>% rownames_to_column("Genes")

p7_mit <-  data.frame("Genes" = p7mit_mean$Genes, "FPKM" = p7mit_mean$FPKM,
"Status"=c("P7", "P7","P7", "P7","P7"))
#filter Ad mitchondria
mit_ad1 <- Ad1_ftable %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
mit_ad2 <- Ad2_ftable %>% filter(gene_short_name %in% genes_mit) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

admit_com <- cbind(mit_ad1, mit_ad2$FPKM)
admit_com$mean <- rowMeans(admit_com)
admit_mean <- admit_com %>% rownames_to_column("Genes")
ad_mit <- data.frame("Genes" = admit_mean$Genes, "FPKM" = admit_mean$mean,
"Status" = c("Ad", "Ad","Ad", "Ad","Ad"))
mit_fin <- bind_rows(p0_mit, p4_mit, p7_mit, ad_mit)
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to
character
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Cell Cycle

```
#filter P0 cell cycle genes

cc_p01 <- P0_1 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

cc_p02 <- P0_2 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p0cc_com <- cbind(cc_p01, cc_p02)
```

```
p0cc_com$mean <- rowMeans(p0cc_com)
p0cc_mean <- p0cc_com %>% rownames_to_column("Genes")
p0_cc <- data.frame("Genes" = p0cc_mean$Genes, "FPKM" = p0cc_mean$FPKM,
"Status" = c("P0","P0","P0","P0","P0"))
#filter P4 cell cycle genes

cc_p41 <- P4_1 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

cc_p42 <- P4_2 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p4cc_com <- cbind(cc_p41, cc_p42)
p4cc_com$mean <- rowMeans(p4cc_com)
p4cc_mean <- p4cc_com %>% rownames_to_column("Genes")

p4_cc <-  data.frame("Genes" = p4cc_mean$Genes, "FPKM" = p4cc_mean$FPKM,
"Status"=c("P4", "P4","P4", "P4","P4"))
#filter P7 cc
cc_p71 <- P7_1 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

cc_p72 <- P7_2 %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

p7cc_com <- cbind(cc_p71, cc_p72)
p7cc_com$mean <- rowMeans(p7cc_com)
p7cc_mean <- p7cc_com %>% rownames_to_column("Genes")

p7_cc <-  data.frame("Genes" = p7cc_mean$Genes, "FPKM" = p7cc_mean$FPKM,
"Status"=c("P7", "P7","P7", "P7","P7"))
#filter Ad mitchondria
cc_ad1 <- Ad1_ftable %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")
cc_ad2 <- Ad2_ftable %>% filter(gene_short_name %in% genes_cc) %>%
select(gene_short_name, FPKM) %>% column_to_rownames("gene_short_name")

adcc_com <- cbind(cc_ad1, cc_ad2$FPKM)
adcc_com$mean <- rowMeans(adcc_com)
adcc_mean <- adcc_com %>% rownames_to_column("Genes")
ad_cc <- data.frame("Genes" = adcc_mean$Genes, "FPKM" = adcc_mean$mean,
"Status" = c("Ad", "Ad","Ad", "Ad","Ad","Ad", "Ad","Ad", "Ad","Ad"))
cc_fin <- bind_rows(p0_cc, p4_cc, p7_cc, ad_cc)
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to
character
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Plots

```
library(ggplot2)

#Sacromere
sac_fin$Status = as.factor(sac_fin$Status)
sac_fin$Status <- factor(sac_fin$Status, levels=c("P0", "P4", "P7","Ad"))
ggplot(sac_fin, aes(x=Status, y=FPKM, col=Genes, group=Genes)) +geom_point()+
geom_path()+labs(x="", y="FPKM")+scale_y_continuous(breaks=seq(100,1300,100))
+ggtitle("Sacromere")+theme(plot.title = element_text(hjust = 0.5))
#Mitochondria
mit_fin$Status = as.factor(mit_fin$Status)
mit_fin$Status <- factor(mit_fin$Status, levels=c("P0", "P4", "P7","Ad"))
ggplot(mit_fin, aes(x=Status, y=FPKM, col=Genes, group=Genes)) +geom_point()+
geom_path()+ scale_y_continuous(breaks=seq(0,280,50)) + theme(plot.title =
element_text(hjust = 0.5))+ggtitle("Mitochondria")
#Cell Cycle
cc_fin$Status = as.factor(cc_fin$Status)
cc_fin$Status <- factor(cc_fin$Status, levels=c("P0", "P4", "P7","Ad"))
ggplot(cc_fin, aes(x=Status, y=FPKM, col=Genes, group=Genes)) +geom_point()+
geom_path()+ scale_y_continuous(breaks=seq(0,50,10)) + theme(plot.title =
element_text(hjust = 0.5))+ggtitle("Cell Cycle")
gdiff2 <- Gdiff %>% filter(Gdiff$significant == 'yes' & Gdiff$significant ==
'yes' & Gdiff$log2.fold_change. !="Inf" & Gdiff$log2.fold_change. !="-Inf")
%>% arrange(log2.fold_change.)

head500 <- gdiff2 %>% head(gdiff2[unique(gdiff2$gene),]$log2.fold_change.,
n=500)
tail500 <- gdiff2 %>% tail(gdiff2[unique(gdiff$gene),]$log2.fold_change.,
n=500)
diff1000 <- rbind(head500,tail500)

#1000 genes
diff1000df <- data.frame("Genes"=diff1000$gene, "log fold change" =
diff1000$log2.fold_change.)
#diff1000_2 <- diff1000[, c(gene, "log2.fold_change.")]
library(tidyverse)
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
## ── Attaching packages
───────────────────────────────────────────────────────────────────────────────
────── tidyverse 1.2.1 ──
```

```
## ✔ tidyr   0.8.3     ✔ purrr   0.3.2
## ✔ readr   1.3.1     ✔ stringr 1.4.0
## ✔ tidyr   0.8.3     ✔ forcats 0.4.0
## ── Conflicts ──────────────────────────────────────────────────────────────────
──────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
#FPKM values


P01_fpkm <- P0_1 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P01_temp <- P01_fpkm[!duplicated(P01_fpkm$gene_short_name), ] %>% rename(P0_1
= FPKM)
#rownames(P01_temp) <- NULL
#P01_fin <- P01_temp %>% column_to_rownames("gene_short_name") %>%
rename(P0_1 = FPKM)

P02_fpkm <- P0_2 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P02_temp <- P02_fpkm[!duplicated(P02_fpkm$gene_short_name), ] %>% rename(P0_2
= FPKM)

P42_fpkm <- P4_2 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P42_temp <- P42_fpkm[!duplicated(P42_fpkm$gene_short_name), ] %>% rename(P4_2
= FPKM)

P41_fpkm <- P4_1 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P41_temp <- P41_fpkm[!duplicated(P02_fpkm$gene_short_name), ] %>% rename(P4_1
= FPKM)

P71_fpkm <- P7_1 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P71_temp <- P71_fpkm[!duplicated(P71_fpkm$gene_short_name), ] %>% rename(P7_1
= FPKM)

P72_fpkm <- P7_2 %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
P72_temp <- P72_fpkm[!duplicated(P71_fpkm$gene_short_name), ] %>% rename(P7_2
= FPKM)

ad1_fpkm <- Ad1_ftable %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
ad1_temp <- ad1_fpkm[!duplicated(ad1_fpkm$gene_short_name), ] %>% rename(Ad_1
= FPKM)
```

```r
ad2_fpkm <- Ad2_ftable %>% filter(gene_short_name %in% diff1000df$Gene) %>%
select(gene_short_name,FPKM)
ad2_temp <- ad2_fpkm[!duplicated(ad2_fpkm$gene_short_name), ] %>% rename(Ad_2
= FPKM)

#concatenated table
concat04 <- inner_join(P01_temp, P02_temp, by="gene_short_name") %>%
inner_join(., P41_temp, by="gene_short_name") %>% inner_join(., P42_temp,
by="gene_short_name") %>% inner_join(., P71_temp, by="gene_short_name") %>%
inner_join(., P72_temp, by="gene_short_name") %>% inner_join(., ad1_temp,
by="gene_short_name") %>% inner_join(., ad2_temp, by="gene_short_name")
concat_temp <- concat04[!duplicated(concat04$gene_short_name), ]
rownames(concat_temp) <- NULL
concat_fin <- concat_temp %>% column_to_rownames("gene_short_name")


#heatmap

heatmap(as.matrix(concat_fin))
```