CSE 180 Final Project: Differential Expression Analysis Pipeline

Karina Soerjanto, Katie Huang, Cherie Huang, Dennis Cao

Biological Motivation

Often times when tackling a biological question, we would like to know which genes are activated or deactivated in certain pathways because this can help guide a research project. In order to solve this problem, we can use next generation sequencing and perform differential expression analysis. Differential expression analysis, or RNA-seq, attempts to identify the genes that are differentially expressed in distinct groups of samples by looking at the difference in levels of mRNA between the groups of samples.

Usually in a differential expression analysis we have a control sample, which is commonly a healthy sample under normal conditions. We compare this control sample to an experimental sample that has undergone different circumstances than the control. The samples are obtained by taking a snapshot of the levels of mRNA in both the control and experimental specimens. After we obtain the mRNA reads, we align the reads to a reference genome and count the number of reads that are aligned in each gene. After normalizing the counts by taking into account the different lengths of the genes, we are able to determine which genes are highly expressed or lowly expressed between the two samples. Therefore, by studying the levels of mRNA, we are able to identify which genes were differentially expressed at a given time under a given condition, which can lead to several biological insights. Using this information we can target some of these differentially expressed genes for genetic alterations, thus potentially changing the outcome of certain pathways and curing diseases.

For example, if we perform a differential expression analysis on a sample of bacteria under normal conditions against that same strain of bacteria treated with an antibiotic that it's

resistant to, we can find the genes that are differentially expressed, and from there perform further experiments to identify exactly which gene is responsible for the resistance. After discovering the gene of interest, we can develop drugs to alter that gene's expression levels and solve our initial problem of antibiotic resistance. This is just one of the many examples of the power of differential expression analysis in solving current biological problems.

Project Details

Our project automates and streamlines an RNA-seq pipeline by using bash scripting and python modules. The user can run our script giving the input as their samples of interest along with the reference genome of the samples that they have. The script will run several bioinformatics tools that are used in differential expression analysis. After the script is finished executing, the user will have a list of files. Contained in this list is a summary file that shows N of the most differentially expressed genes that are also significant (have a p-value of < 0.05). The main purpose of our program is to simplify the execution of RNA-seq pipelines, which can often have a steep learning curve for people who have never used a command-line environment. Various parameters and formats can also be confusing and overwhelming to new users. With the use of our program, the user can easily give input and using a single command, run an entire pipeline and receive a differential expression analysis of their samples. Given the input and a single command, our program can run an entire pipeline and provide a differential expression analysis of the samples.

Our program will prompt the user with questions regarding their samples. It will ask whether the samples are single or paired-end reads, the insert size if it differs from the default, the reference genome used in the form of a .fasta file, and the samples in the form of .fastq files. Since this program will generate the N most differentially expressed genes that are biologically

significant, it will prompt the user to enter what N should be. Once the program has the information necessary, it enters the first stage of the pipeline and performs the alignment of mRNA reads to the given reference genome using TopHat [1]. Next in the RNA-seq pipeline, it utilizes Cufflinks [2] to assemble the transcripts. This stage outputs a separate file of assembled transcripts for each file. In order to merge these together, Cuffmerge [3] is called on the files. In order to identify the differentially expressed genes, Cuffdiff [2] is called next. This outputs a file containing information regarding the genes, such as the log2 fold change, and p-value. This output is parsed by extracting the top N most differentially expressed genes with a p-value that is less than 0.05. It is written to a user friendly file that can be easily opened and interpreted.

To implement our project, we downloaded the bioinformatics tools mentioned above: TopHat, Cufflinks, Cuffmerge, Cuffdiff and we executed the programs as a pipeline using a bash script. We also used the bash script to print to the terminal to prompt the user and receive input. Once the pipeline finished running, we parse the output files of cuffdiff using python and present to the user the names, log 2 fold changes, and p values of the top N most differentially expressed genes between the input samples.

Project Results

In order to use our program, you would need to download Samtools, Bowtie, Tophat, Cufflinks, Cuffmerge, and Cuffdiff and add the variables bowtie, tophat, and cufflinks to your PATH. Additionally, you need to have Python installed on your machine. Following, to use our pipeline, simply run the script by typing into your terminal:

```
$ ./rna_seq_pipeline.sh
```

Again, you will be prompted to enter several different input files, such as your .fastq samples and your .fasta reference files, and the optional parameters to run the pipeline. Once you have gone through the series of prompts, the program will execute the tools to run the pipeline.

The purpose of our program is to automate and simplify the execution of an RNA-seq pipeline. Without this streamlined program, users would have to enter a command, wait several hours for a command to finish before having to return to the computer and enter the next command. In addition to not having to execute the different tools separately, this program simplifies the pipeline by prompting the users for exactly what is needed, which eliminates the confusion regarding how to execute these tools. It abstracts away the complex underlying details of the programs run, and acts as a user-friendly interface. Due to the fact that RNA-seq data has a very high throughput, the program takes approximately a couple of hours to complete running the pipeline on a realistic dataset. With that being said, our program doesn't deviate far from the average time that it usually takes to complete a differential expression analysis. It merely streamlines the execution of several bioinformatics tools. Some of the drawbacks of our pipeline is the fact that the user cannot specify extra optional parameters if they would like to, as the program will automatically run the pipeline for them. Because we add abstraction to simplify the pipeline, we restrict the users freedom with running these tools. Additionally, our program only allows the user to perform a differential expression analysis between 2 samples, so if the researcher has multiple samples they would have to run this pipeline multiple times.

Ultimately, our pipeline is designed to be for researchers that are new to bioinformatics and do not have the need to specify optional parameters or run complex commands. For someone who is relatively new to computational biology, learning the tools and commands to perform this pipeline can be confusing and overwhelming. By prompting for the various

parameters in the program, we eliminate the confusion of how to format these commands. Therefore, although the users have less freedom in the number of files and the input of optional parameters, overall this program is useful because it automates and simplifies the pipeline greatly.

Team Contributions:
Dennis Cao - Wrote the bash script and contributed to the python script
Karina Soerjanto - Researched the topic and wrote a lot of the writeup. Did a lot of the python portion of the script.
Katie Huang - Edited slides, created README, and handled the logistics of turning in the project.
Cherie Huang - Proofread and added to the writeup and helped test the code.

Notes:
On our own machines, we tested with files retrieved from the NCBI database (https://blast.ncbi.nlm.nih.gov/Blast.cgi) , but were unable to upload it to the Github repo.

For testing purposes, feel free to use the test data we provided within our repo:

- reads_1.fq
- reads_2.fq
- test_ref.fa

References
[1] Trapnell, Cole, Lior Pachter, and Steven L. Salzberg. "TopHat: discovering splice junctions with RNA-Seq." *Bioinformatics* 25.9 (2009): 1105-1111.
[2] Goff, L., C. Trapnell, and D. Kelley. "cummeRbund: Analysis, exploration, manipulation, and visualization of Cufflinks high-throughput sequencing data." *R package version* 2.0 (2012).
[3] Trapnell, Cole. "Cufflinks. cuffmerge (v2)."