Auburn University Bioinformatics Bootcamp 2018

Small Group Exercise #8:

Examining patterns of differentially expressed (DE) transcripts

Getting Started

1. In this exercise, we will be examining patterns of differentially expressed (DE) transcripts in the *Lamellibrachia luymesi* transcriptome, with an emphasis on how different methodological approaches can influence the results and interpretation. We will also relate the results back to the transcriptome annotation that was done with Trinotate earlier in the week. Nearly all of this exercise would be the same if you were conducting DE analyses across treatments. The primary difference would be in the final analyses, which would use statistical tests for DE. First, create a working directory in your home directory:

- `cd` (to take you to the top level of your home directory)
- `mkdir express_example` (create the working directory)

1. Now `cd` into the new directory and get ready to copy three (3) data files that we need into the directory. The first two (2) files are named `Lamellibrachia_luymesi_transcriptomic_sub2.5M_L001_R1_001.fastq` and `Lamellibrachia_luymesi_transcriptomic_sub2.5M_L001_R2_001.fastq` and are located in `/home/shared/biobootcamp/data/Lamellibrachia_luymesi_sequence_reads_for_assembly`. The third file is named `Lamellibrachia_luymesi_all_reads_NORM_TRI_05_2015.fasta` and located in `/home/shared/biobootcamp/data/Lamellibrachia_luymesi_finished_genomic_transcriptomic_assemblies`. Grab those now. Can you guess from the name how the FASTQ files were generated?

2. You will also need to edit three ASC queue scripts for this exercise (you might want to start asking yourself what's different about the three runs). Those are named `Bowtie_Express_R*` (with * representing different ending names to the scripts). You can copy them from `/home/shared/biobootcamp/data/example_ASC_queue_scripts`, using a wildcard (i.e., *) in your `cp` command to get all three scripts.

Creating an index

4. At the heart of Bowtie is the Burrows-Wheeler transform, which is used to compress the size of the genome and also makes searching the genome faster. You will use the bowtie-build

command to pre-process (or index) the FASTA file to make searches faster and also compress the size of the reference.

Load Bowtie into your workspace using `module load bowtie`. Then execute the following, substituting the ALL_CAP with the appropriate information:

```
bowtie-build --offrate 1 NAME_OF_FASTA_FILE.fasta BASENAME_OF_BOWTIE_INDEX
```

- For convenience, set the BASENAME_OF_BOWTIE_INDEX to something short (no spaces). For example, replace BASENAME_OF_BOWTIE_INDEX with LLNORMTRI.

- What do you think the `--offrate 1` option is doing during the creation of the index? HINT: run `bowtie-build` without options to find out or use your google-fu.

- Once the index is created (might take a minute or two), you will see a number of new files ending in `*.ebwt` in the current directory if you do a “long listing”. Note that Bowtie names its indices differently than Bowtie2 and BWA (and no, they are NOT interchangeable).

Editing your scripts

5. Now open all three (3) scripts at the same time using `nano Bowtie_Express_R*`. Read through the comments of each script to understand what they are doing and why they are different from each other. Now modify each script by adding the names of the FASTQ files, the FASTA file and the index basename where appropriate (indicated by ALL_CAPS in the scripts). Lastly, note the parameters at the bottom of each script (they are all the same) for submission to the ASC queue.

Submitting your scripts

7. Now you are ready to submit each of the three (3) scripts in succession to the ASC queue system using the noted parameters. Once the third script has been submitted, check their status using `squeue`.

8. As the three (3) jobs are “cooking”, make a copy of the Trinotate annotation report for the transcriptome FASTA file that we are working with. You can copy one named `Lamellibrachia_luymesi_all_reads_NORM_Trinotate_Annotation_Report_May2015.tab` from `/home/shared/biobootcamp/data/Lamellibrachia_luymesi_Trinotate_annotation/Lamellibrachia_luymesi_Trinotate_annotation_finished/`. Do that now by copying it into your working directory.

9. You’ll also be using a new Linux tool in this exercise, `diff`. Spend a few minutes now

reading over its manual page to understand what it does and why it might be useful in what we are doing:

- `man diff`

10. Check the status of your jobs with `squeue` and see if any have been completed or exited due to errors you might have introduced into your script(s). In the case of the latter, read the output of the particular job logfile, correct the script and resubmit it to the queue.

11. Once the job(s) have completed, you will find that new directories have been created in the current working directory with names similar to those of the scripts. Change directories into any of these and do a long listing to see that each contains just two files. One of them, named `params.xprs`, contains various parameters estimated by the program eXpress, which calculated estimates of differences in expression among transcripts. The other file, named `results.xprs`, contains that information that we are interested in (i.e., the expression levels per transcript calculated by eXpress). Let's do a `head` of the file:

    - `head results.xprs`
    - Note that the file is tab-delimited with 15 columns of information. Which CLI tool works well with data in such a format? HINT: you used it extensively in the parsing of the Trinotate annotation report.
    - The file also has a header denoting information that is in each column. Some of them are self-explanatory while others are somewhat cryptic. More information regarding the content of each column can be found in the manual for eXpress at the program's website:
        - http://bio.math.berkeley.edu/eXpress/manual.html

> **Once all three (3) jobs have successfully completed, we are going to parse the result.xprs reports to 1) determine how similar or different the results of the three (3) separate runs are and 2) use the Trinotate report in an attempt to place annotation to transcripts. We will train our attention on those transcripts that appear to be most highly expressed for the transcriptome of this particular individual of Lamellibrachia luymesi. We'll utilize some variants of commands you have seen previously as well as new one in some simple, but powerful, pipeline.**

Parsing the output

12. To start off, change directories into `express_using_R1_R2/`. While there are 15 columns of data, we will focus on just three (3) and filter those out from the rest: target_id, fpkm and tpm (what's in these latter two (2) columns and why focus on them?). When we filter these three (3) out, we would like the output to be in a tab-delimited format. Lastly, since we are establishing our pipeline, we don't need to run all of the data through, but rather a subset (say

the first 10 lines). From the header of the `results.xprs` file, identify which column number are those for target_id, fpkm and tpm. Once you have those numbers, insert them in place of the “X”s in the command below sequentially and execute the command in your Terminal:

```
head results.xprs | awk 'OFS="\t" {print $X, $X, $X}'
```

- What’s OFS doing in the `awk` command? Consult your `awk` cheat sheet for an explanation.
- A subset of the header will be at the top of the output, so be sure that those match with target_id, fpkm and tpm. If not, the column numbers you provided to `awk` are not correct, go back and modify them.
- Now that we have our subset of data, we can sort them numerically either on the FPKM or TPM columns. So add a pipe to the command in (a) above to send it to `sort`.
    - `sort -r -k2`
    - What are the options `-r -k2` doing to the function of the command? Consult the manual page of `sort` to find out.
    - If you look really closely at the output from `sort`, its not doing EXACTLY what we want because the values are NOT being correctly sorted. To fix this, we will need to modify the above command to include the `-g` option like so:
    - `sort -r -k2 -g`
    - Is that better now? What was actually going on with the command and why did the inclusion of `-g` fix it? Yes, go back and read the manual page again.
- Now that the pipeline is behaving as expected, let’s run it against the entire dataset and redirect the Top 10 results to a text file. The final filter and redirect command would look like (don't forget to replace the "X"s):

```
cat results.xprs | awk 'OFS="\t" {print $X, $X, $X}' | sort -r -k2 | head > ../R1_R2_Top_10_FPKM.txt
```

- Notice that we are redirecting the output to a file one-directory level above where we are currently working. The reason for this will become evident shortly. Also, notice how we used `cat` to stream the entire contents of the file for processing but then used `head` at the end to just “skim” off the Top 10. INFO: you could have taken any number off the top using `head`, but how would you have done that?

    - For fun, try running the above command again with two modifications: without the redirect to the text file and changing the `-k2` to `-k3` in the sort command.
        - Remember what that particular option controls and notice that the two lists don't change at all. What does that tell you about the relationship between FPKM and TPM?

13. Now that we have one set of output dealt with, go back and do the final filter and redirect to a text file command (above in Step 12) for the `results.xprs` files in `express_using_R1_only/` and `express_using_R2_only/`. Name the files you redirect your output to as `R1_only_Top_10_FPKM.txt"` and `R2_only_Top_10_FPKM.txt`. The redirects should send the output to a file one-directory level above where you executed the command.

> **OK, now comes the fun part: actually looking at the Top 10 lists and comparing how similar the contents are. **

14. Change directories into `express_example` and do a long listing to check that the three files (i.e., `R1_R2_Top_10_FPKM.txt`, `R1_only_Top_10_FPKM.txt`, and `R2_only_Top_10_FPKM.txt`) are there. `cat` each one sequentially and examine the lists? Do they look the same or different? Consult with other members of your groups on what their lists look like and what their #1 most expressed transcript is? Is it the same or different across the group?
15. You have probably come to the conclusion that this isn't the most efficient way of looking at the data. This is where `diff` comes into play. We are going to use it to compare the transcript names for each list in a pairwise fashion. `diff` will pick up ANY and ALL difference between lines and since the FPKM and TPM values likely differ across files while transcript names and their order in the list could remain constant, we are going to want to isolate those separately. Thus, use `awk` to extract the target_id columns from `R1_R2_Top_10_FPKM.txt`, `R1_only_Top_10_FPKM.txt`, and `R2_only_Top_10_FPKM.txt` and redirect them to files named `R1_R2_Top_10_names.txt`, `R1_only_Top_10_names.txt`, and `R2_only_Top_10_names.txt`, respectively. Do that now (we'll also be using these name lists for something else shortly….).
16. Once you have your three (3) lists of names, compare them in the following manner:
    - `diff -y R1_only_Top_10_names.txt R2_only_Top_10_names.txt`
    - What does the output look like? What do any symbols like pipe (i.e., |), < or > appear to be telling you in relation to the contents of each list? Talk among your group regarding the results of others.
17. Now let's compare the contents of one of the '*only*' files to that of the '*R1_R2*' file:
    - `diff -y R1_only_Top_10_names.txt R1_R2_Top_10_names.txt`
    - What does the output look like now? Is it much "busier"? Again, what do any symbols like pipe (i.e., |), < or > appear to be telling you in relation to the contents of each list? Talk among your group regarding the results of others.
18. While there are similarities between lists, there are also very clear differences. Remember, we used the EXACT SAME DATA, with the '*only*' files corresponding to quantified expression level using the FASTQs in a single-end fashion while the '*R1_R2*' file used the

FASTQs in a paired-end manner. While both ways are ACCEPTABLE, they are NOT IDENTICAL, so CONSISTENCY WITHIN AN EXPERIMENT IS IMPORTANT!! The other reason for a preference toward single end reads can be found in the statistics Bowtie generated during the mapping. Let's look at those now. You'll need to be in the directory where the job logfiles are (this should be the directory you were in when you submitted the job) - the command below also assumes your job names are in the form BowtieExpressR*:

- `grep -A 3 '^# reads processed:' BowtieExpressR*`
  - "Hey, wait, what's this new option to `grep`? I'll go look it up on the manual page along with the `-B` and `-C` options." Good deal.
- Notice that while there are some minor differences in the percentage of reads mapping when either using the R1 or R2 files, there is a large difference when R1 and R2 are used in a paired-end fashion. WHY DO YOU THINK THIS IS THE CASE? Talk among the group to form hypotheses to explain this.

> **What might be the identities of these highly expressed genes? This is where the name lists that we created above can prove useful.**

19. Our usage of `grep` to this point has been supplying it with a regular expression on the command line. However, it will also accept regular expressions coming from a file. Let's look at how that is done by supplying the contents of our name list(s) to `grep` for searching across the Trinotate annotation report:

    - `grep -f R1_R2_Top_10_names.txt Lamellibrachia_luymesi_all_reads_NORM_Trinotate_Annotation_Report_May2015.tab`

    - This might not very human readable, but just focus on field #3 (based on the tab separator) and, within that field, field #6 (based on a "^" separator).

    - Now isolate the gene names themselves using the techniques you utilized for parsing the Bacteria taxonomic information in Example #5 (Trinotate). Go back to those notes for details. HINT: the section of Ex. #5 (Trinotate) that you are looking for is where multiple `awk` commands were used with different field separators (i.e., the `-F` option).

> **Do any of the potential gene names in the resulting list look familiar and/or interesting, particularly in relation to the biology of Lamellibrachia luymesi? Google some of the names to read find out their potential function. Are there also transcripts that have no annotation associated with them? What might that fact represent (and remember that these are some of the most expressed "genes" identified from this particular experiment)? Talk within your group about this.**