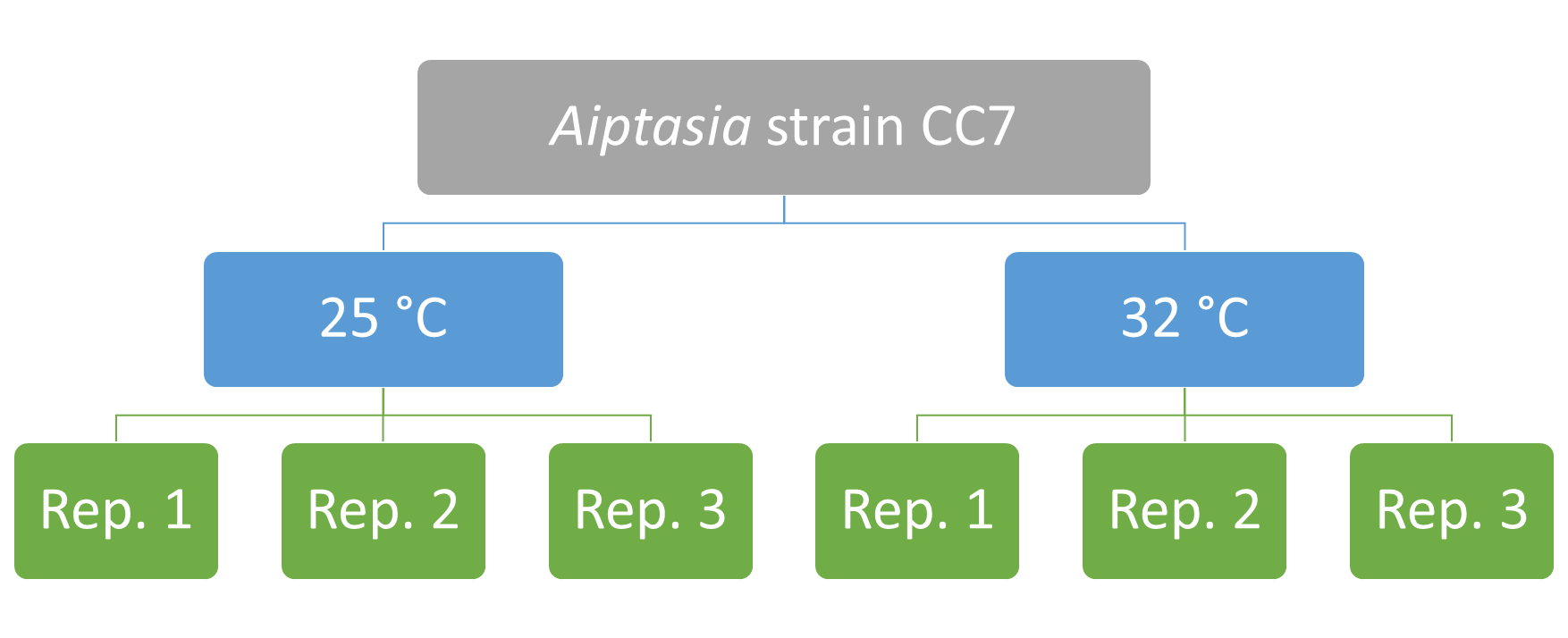
**Stats in R: Day 4, hands-on session**

**Transcriptomics, from raw reads, to real results!**

Note: all commands are case-sensitive. “ssh” will work; “Ssh” would not. If you have weird crashes, double check whether you have typed the commands exactly as shown.

1. **Understanding the experimental setup, and what we’re doing**

We’re dealing with a heat stress experiment performed on Aiptasia strain CC7. The normal temperature is 25 °C, the stress temperature is 32 °C. There are three replicates per temperature.



We’re interested in seeing which genes are differentially expressed under heat stress (using kallisto and sleuth).

Once we have a bunch of differentially expressed genes, we want to see what kind of biological function is associated with these genes, i.e. which genes are heat-stress-related genes? We use topGO for this.

kallisto is written in C and C++; sleuth and topGO are written in R (which helps me fulfil the R in “Stats in R”!)

1. **Logging in to a server via the command line**

On Mac OS X, it’s called “Terminal”.

Type

**ssh stats@lithium.kaust.edu.sa**

then press ENTER.

Type

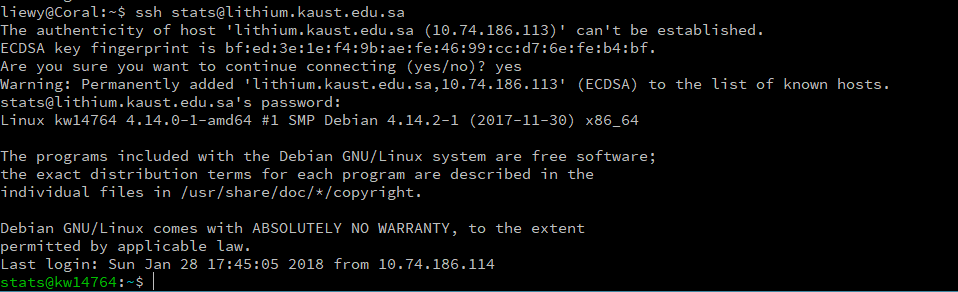
**yes**

in the “Are you sure you want to continue connecting”, then press ENTER.

Type

**stats**

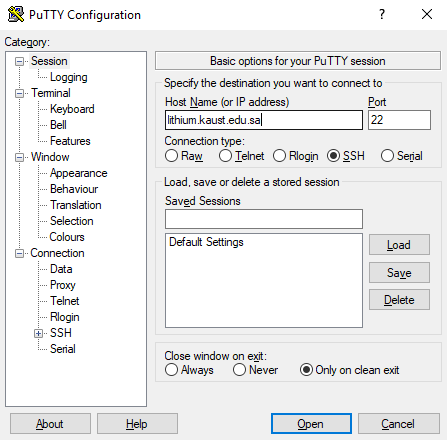
in the password column (without the quotes). You won’t see the cursor moving, that’s normal. Press ENTER.



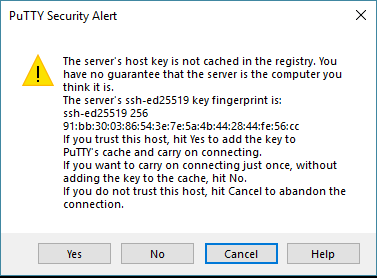
You’re done here, skip the Windows-specific stuff below and go to (2).

On Windows, it’s called “PuTTY”.

Fill in the hostname (lithium.kaust.edu.sa), then click Open.



Say Yes to this.



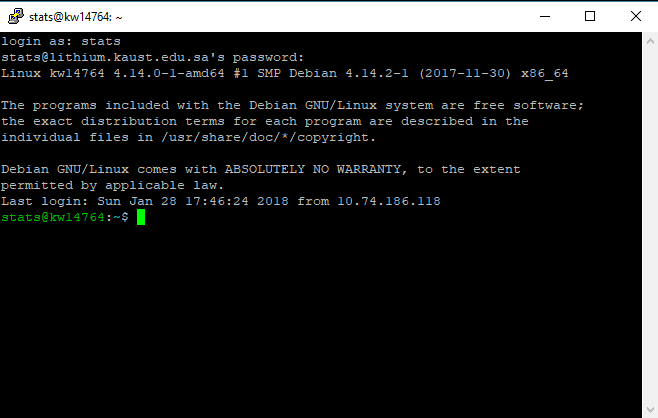
Type

**stats**

in “login as”, press ENTER; then type

**stats**

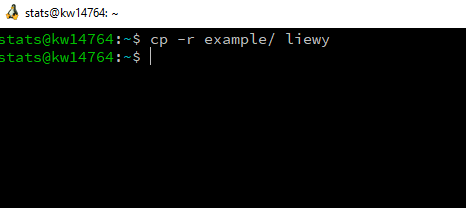
in the password field, and press ENTER.



1. **Make a copy of the example files provided to your own personal folder**

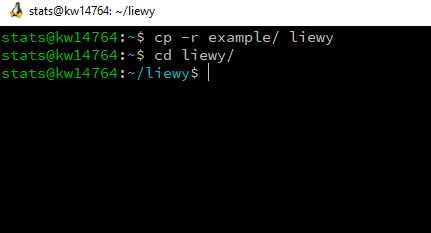
**cp -r example/ <your KAUST username>**

So, as my username is “liewy”, I’ll write



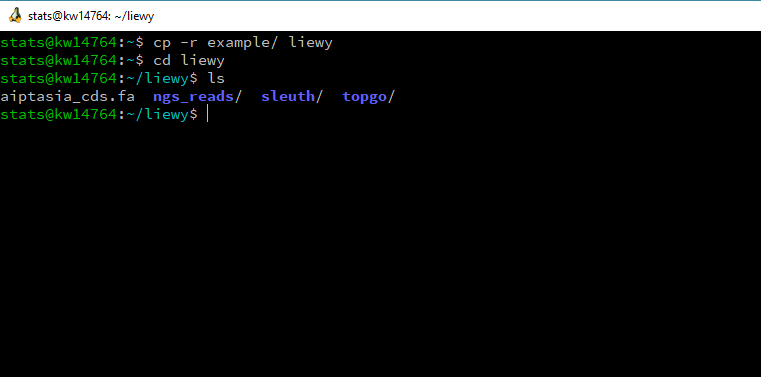
Then enter your own directory

**cd <your KAUST username>**



1. **Exploring your data (briefly)**

**ls** shows you the contents of your folder.



Hmm, let’s look at the contents of aiptasia\_cds.fa.

**less aiptasia\_cds.fa**

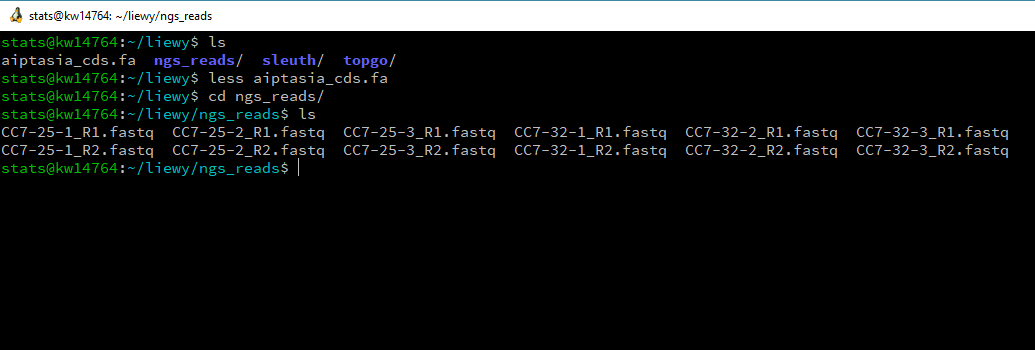


This gives you a brief look at your data. Press **q** to get out (“quit”).

Let’s look at the folder “ngs\_reads”.

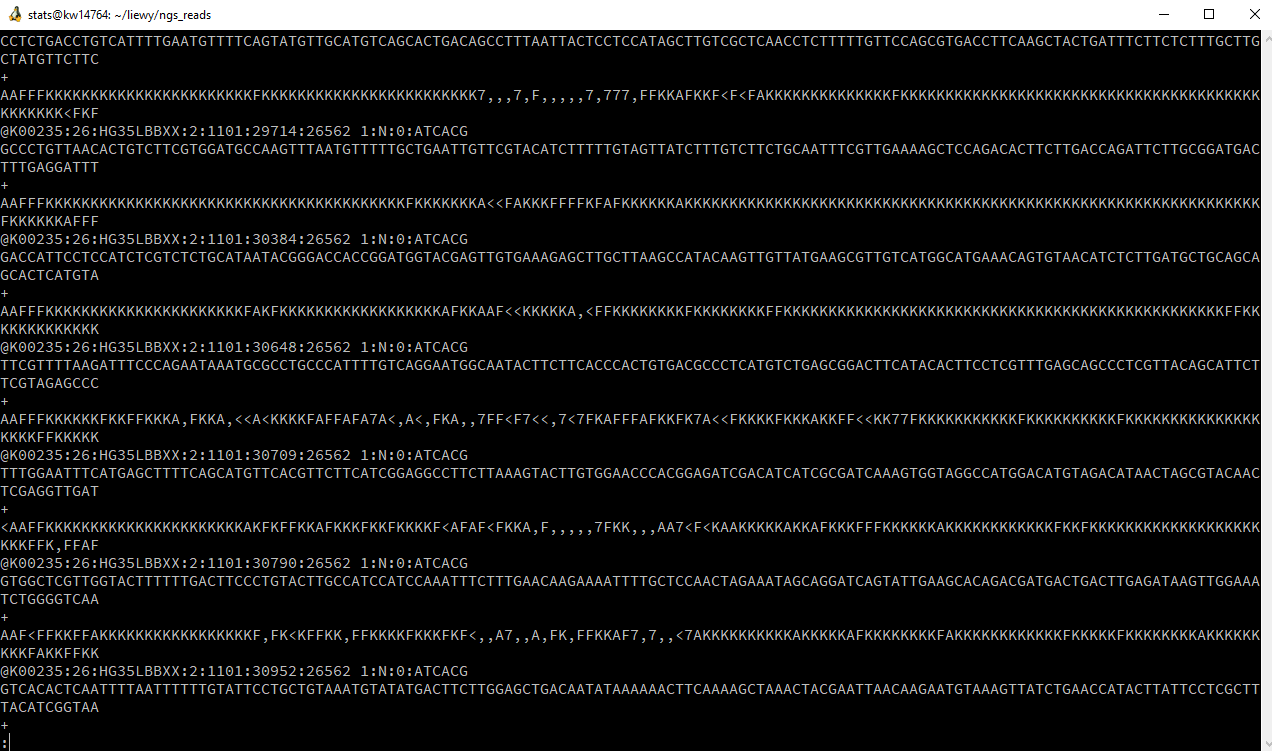
**cd ngs\_reads**

**ls**



Ooh, FASTQs. How does a FASTQ file look like?

**less CC7-25-1\_R1.fastq**



Yep, this is what bioinformaticists mean when they “deal with NGS data”. This data was generated from a next-generation sequencer (Illumina 2000 for this case). Whee.

Again, **q** to quit.

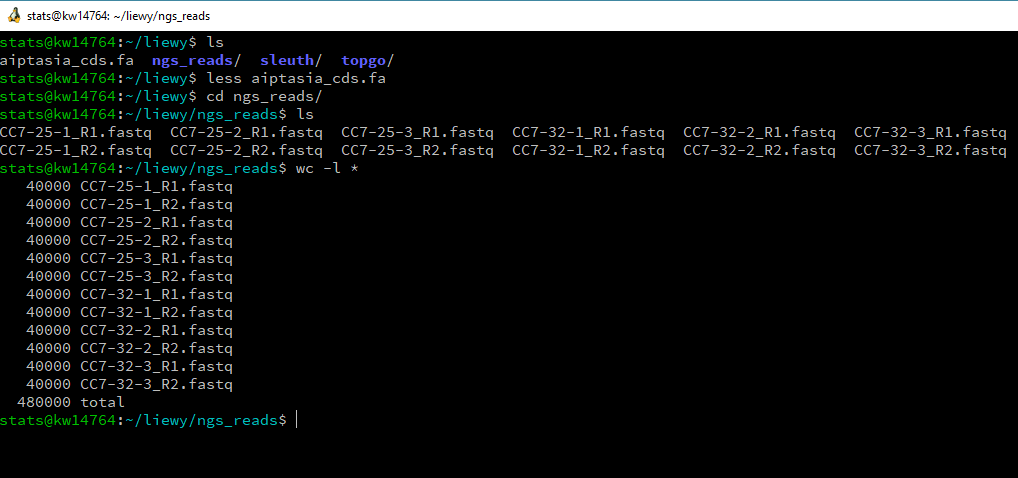
Let’s do a quick count of the number of lines of your files.

**wc -l \***

wc = “word count”

-l = “number of lines”. This is what we call a “flag”. Flags are optional parameters, and flags are specific to your program. -l might mean something else in another program.

\* = “all files”

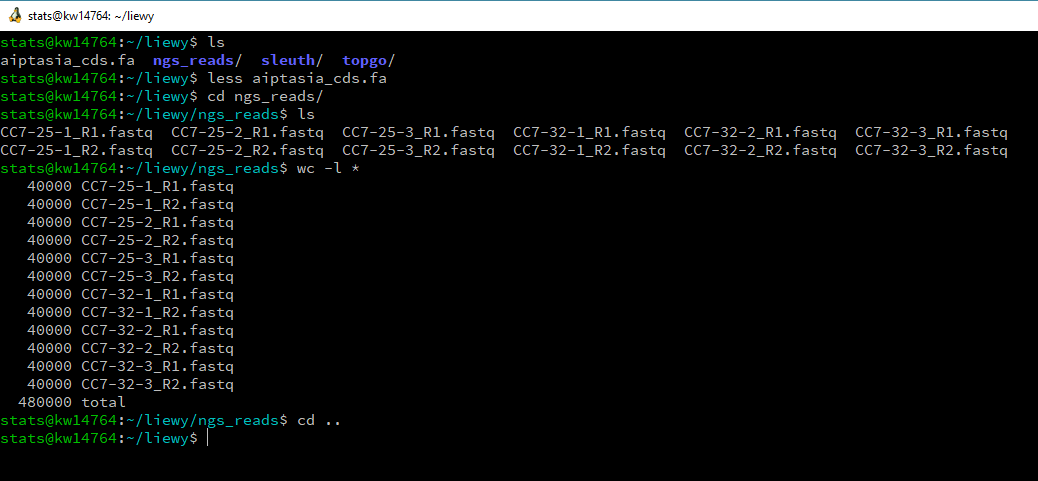


Each file has 40,000 lines, i.e. 10,000 reads (as each NGS read occupies 4 lines in the file).

(The real files have ~10,000,000 reads, but it’s huge, so I cut out a small portion of the real thing.)

OK, that’s all, let’s go back to the previous folder.

**cd ..**



1. **Trimming reads**

Most older differential expression packages (DESeq2, edgeR, baySeq, ...) requires you to trim adapter sequences from reads.

Nowadays, new packages allow you skip this step. Fortunately, we’re using one (kallisto) that allows us to skip this, so skip this we will!

I left this section in just in case your colleagues/bioinformatician/PI prefer the older packages. If you need to trim adapters, look into TrimGalore or trimmomatic.

1. **Running kallisto**

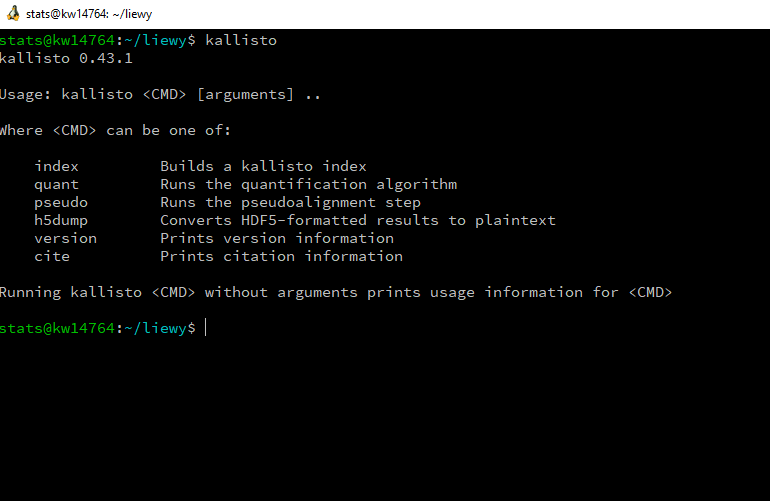
“kallisto” is the package that calculates relative frequencies of the transcripts. The unit of measurement is “tpm”: **t**ranscripts **p**er **m**illion sequenced transcripts.

(Just a quick illustration: if 20 sequenced reads out of 2 million reads maps to Gene X, the gene has a tpm value of 10, i.e. 10 reads per 1 million reads.)

**5a: how do I see what kallisto can do?**

There’s an online manual (<https://pachterlab.github.io/kallisto/manual>), or... type kallisto with no arguments. Well-written programs usually give you hints on how to run it.

**kallisto**

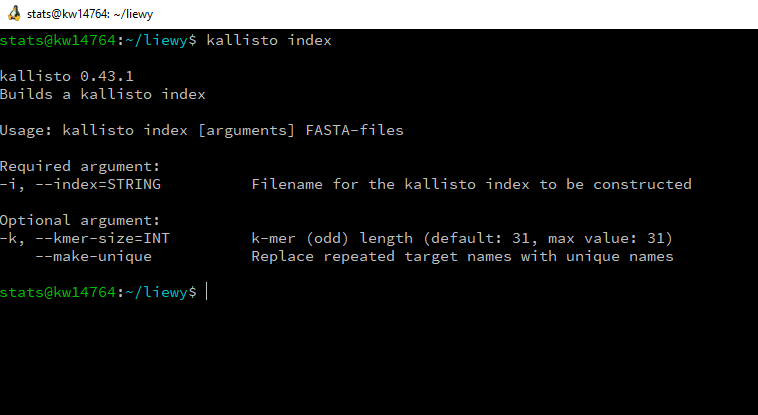


Note the line “Running kallisto <CMD> without arguments prints usage information for <CMD>”. This means that help’s always at hand!

**5b: run kallisto index**

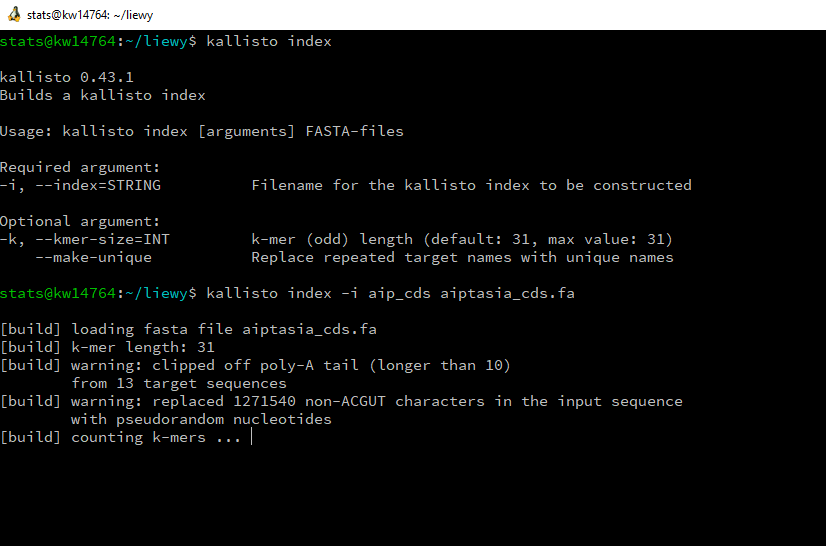
To get more info on how to run kallisto index, run it with no arguments.

**kallisto index**



A-ha. Run

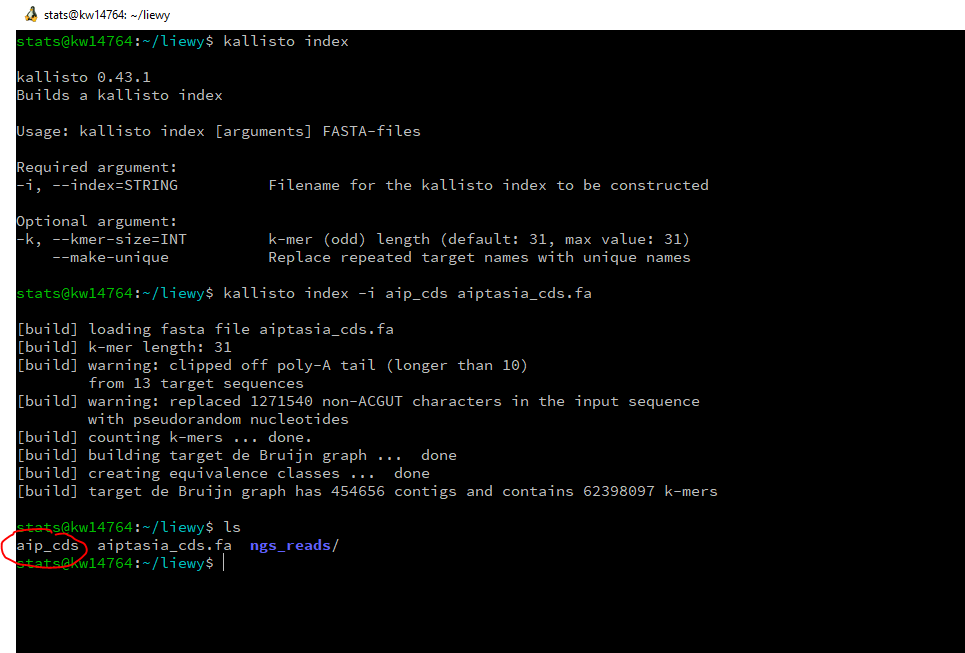
**kallisto index -i aip\_cds aiptasia\_cds.fa**



Let this run for a bit, about 5 mins or so. Toilet break!

When it finishes, you should see a new file called “aip\_cds”.

**ls**



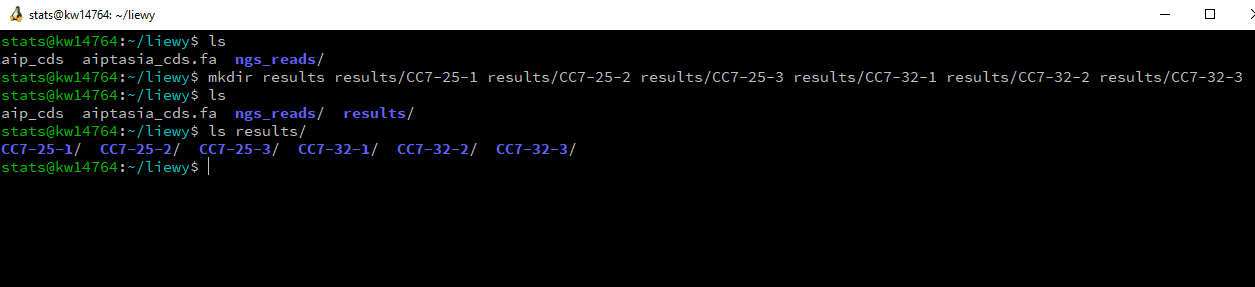
Done!

**5c: Create empty folders to contain results**

To create these folders, there’s a hardworking (but easier to understand) way and a lazy (but harder to understand) way. Both ways lead to Rome.

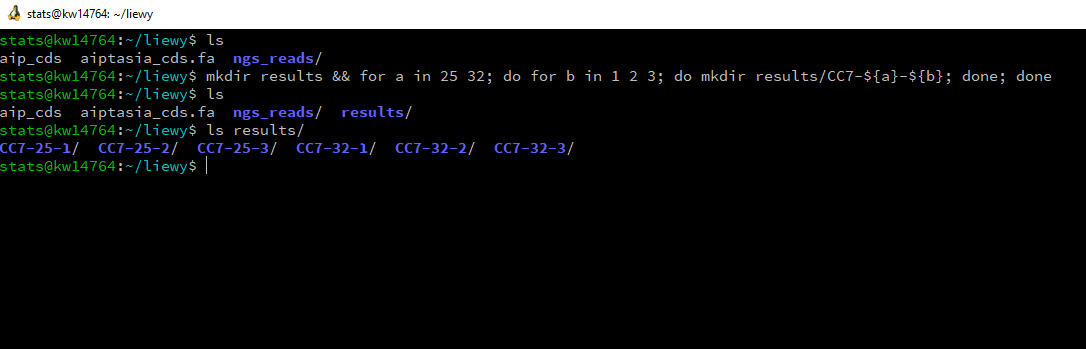
EITHER

**mkdir results results/CC7-25-1 results/CC7-25-2 results/CC7-25-3 results/CC7-32-1 results/CC7-32-2 results/CC7-32-3**



OR

**mkdir results && for a in 25 32; do for b in 1 2 3; do mkdir results/CC7-${a}-${b}; done; done**



**5d: calculate TPMs via kallisto quant**

Same as previous, hardworking vs. lazy.

EITHER

Run six commands, one after another

**kallisto quant -i aip\_cds -o results/CC7-25-1 --bias --rf-stranded -b 100 ngs\_reads/CC7-25-1\_R1.fastq ngs\_reads/CC7-25-1\_R2.fastq**

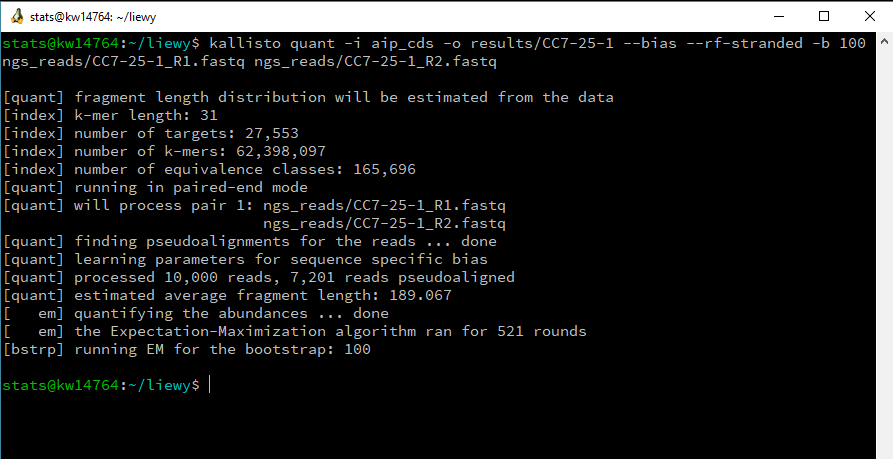
**kallisto quant -i aip\_cds -o results/CC7-25-2 --bias --rf-stranded -b 100 ngs\_reads/CC7-25-2\_R1.fastq ngs\_reads/CC7-25-2\_R2.fastq**

**kallisto quant -i aip\_cds -o results/CC7-25-3 --bias --rf-stranded -b 100 ngs\_reads/CC7-25-3\_R1.fastq ngs\_reads/CC7-25-3\_R2.fastq**

**kallisto quant -i aip\_cds -o results/CC7-32-1 --bias --rf-stranded -b 100 ngs\_reads/CC7-32-1\_R1.fastq ngs\_reads/CC7-32-1\_R2.fastq**

**kallisto quant -i aip\_cds -o results/CC7-32-2 --bias --rf-stranded -b 100 ngs\_reads/CC7-32-2\_R1.fastq ngs\_reads/CC7-32-2\_R2.fastq**

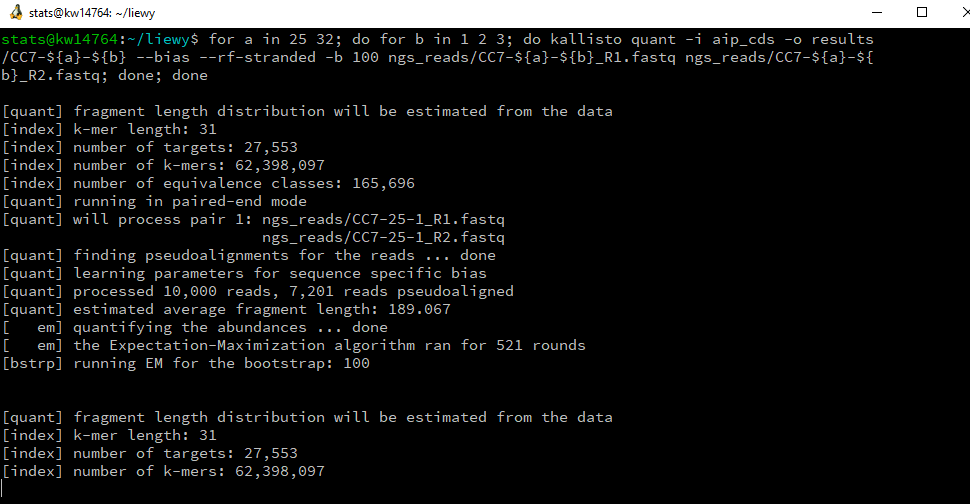
**kallisto quant -i aip\_cds -o results/CC7-32-3 --bias --rf-stranded -b 100 ngs\_reads/CC7-32-3\_R1.fastq ngs\_reads/CC7-32-3\_R2.fastq**



OR

Run one very complex command (it’s a loop that runs 6 commands in succession)

**for a in 25 32; do for b in 1 2 3; do kallisto quant -i aip\_cds -o results/CC7-${a}-${b} --bias --rf-stranded -b 100 ngs\_reads/CC7-${a}-${b}\_R1.fastq ngs\_reads/CC7-${a}-${b}\_R2.fastq; done; done**



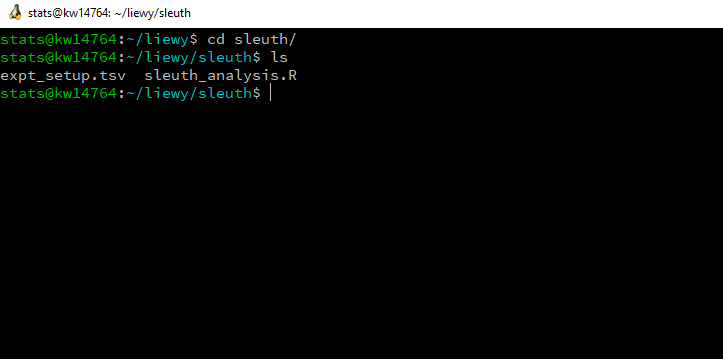
Done!

1. **Running sleuth**

Go into the sleuth folder, and look around.

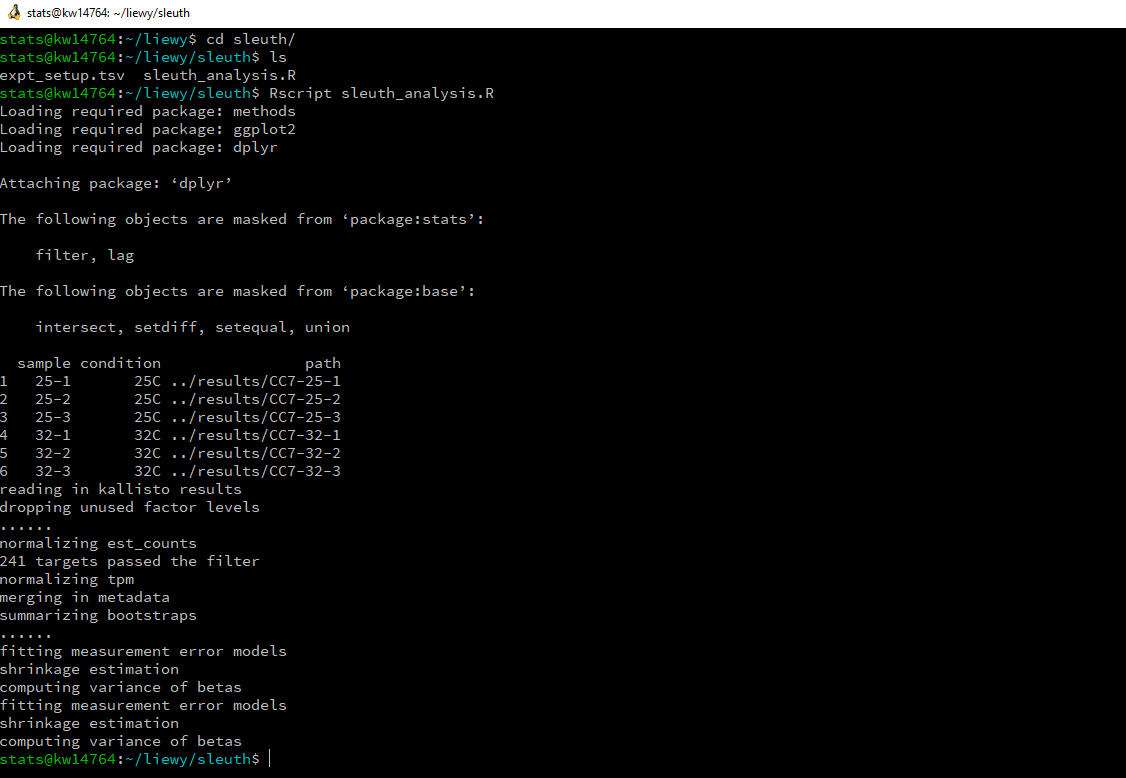
**cd sleuth**

**ls**



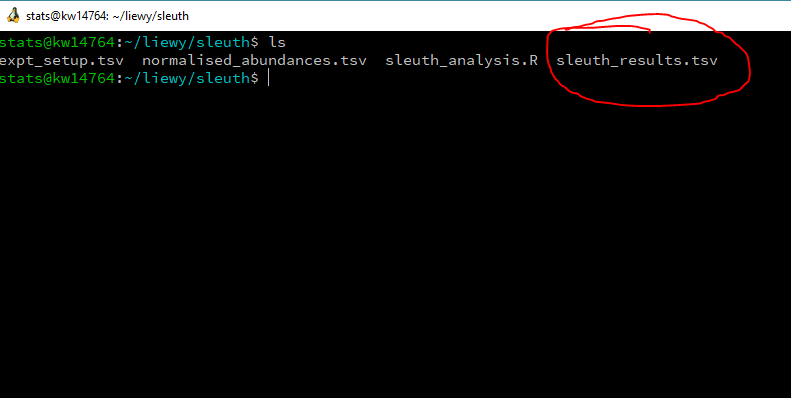
I have provided you the code to run the sleuth analysis—if you’re curious, feel free to read it (**less sleuth\_analysis.R**)—but let’s just go ahead and run the R script with...

**Rscript sleuth\_analysis.R**



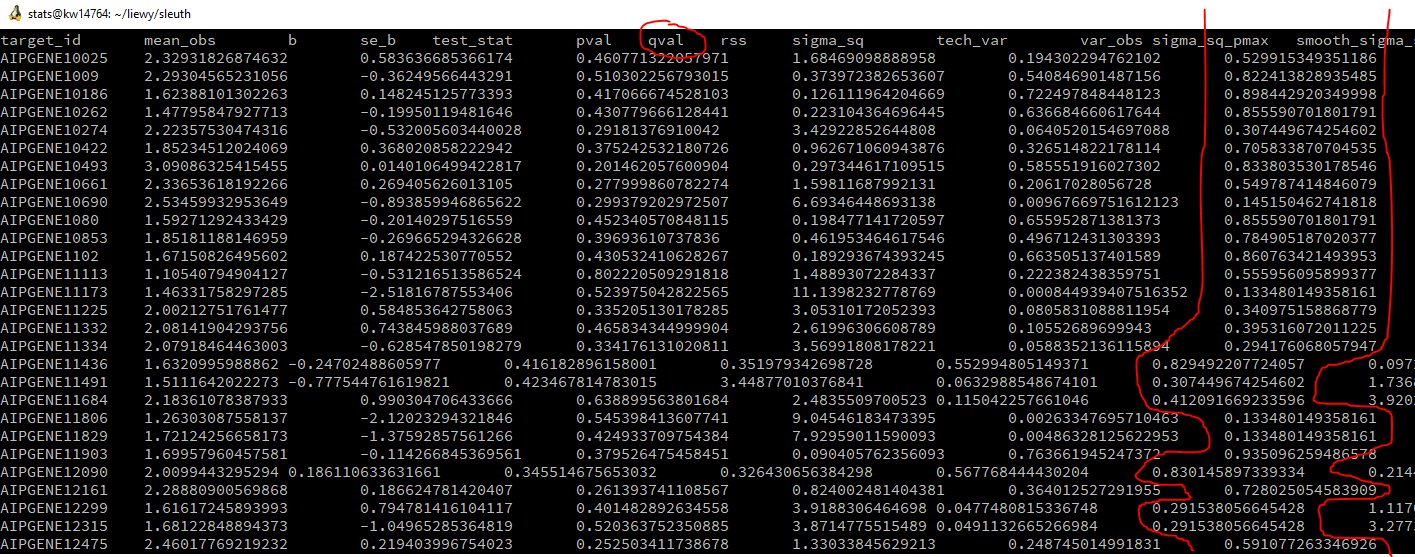
The R script produces two files. We’re interested in one of them.

**ls**



Let’s have a look at the file

**less sleuth\_results.tsv**



“qval” (i.e. post-Benjamini-Hochberg multiple testing corrected “pval”) is the column that tells me whether a gene is significantly expressed or not.

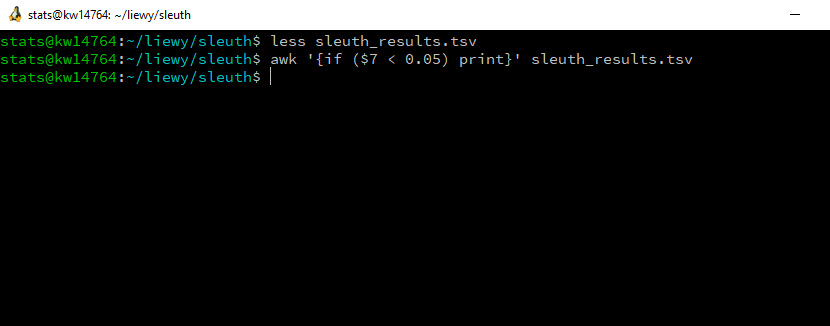
If qval < 0.05, the gene is significantly differentially expressed at 32 C.

If qval > 0.05, the gene is not differentially expressed.

Press **q** to get out of less.

How do we which genes are differentially expressed, from the command line?

**awk '{if ($7 < 0.05) print}' sleuth\_results.tsv**

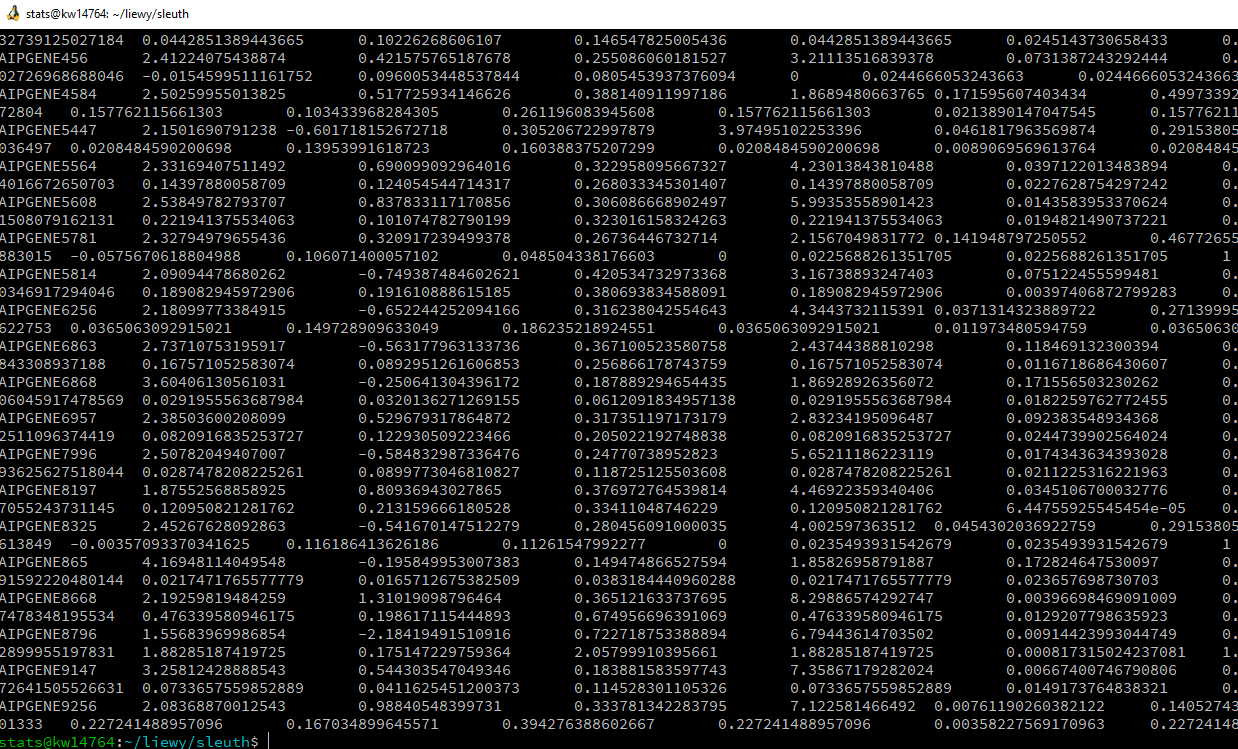


... uh, no results? What.

Unfortunately, because we only used a tiny tiny fraction of the NGS data produced by the experiment, the pipeline failed to find any genes that were differentially expressed. The underlying reason is because very few genes had had reads mapping to it. Differential expression works best when you have low (but detectable) expression in the three replicates of one condition and high expression in the other three replicates.

Doesn’t matter for now—we’ll proceed by defining genes that have p < 0.5 (ha) as significantly differentially expressed.

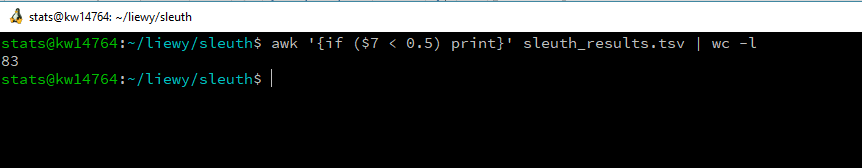
**awk '{if ($7 < 0.5) print}' sleuth\_results.tsv**



Hallelujah, there IS something produced.

To find out how many genes that are “differentially expressed”,

**awk '{if ($7 < 0.5) print}' sleuth\_results.tsv | wc -l**

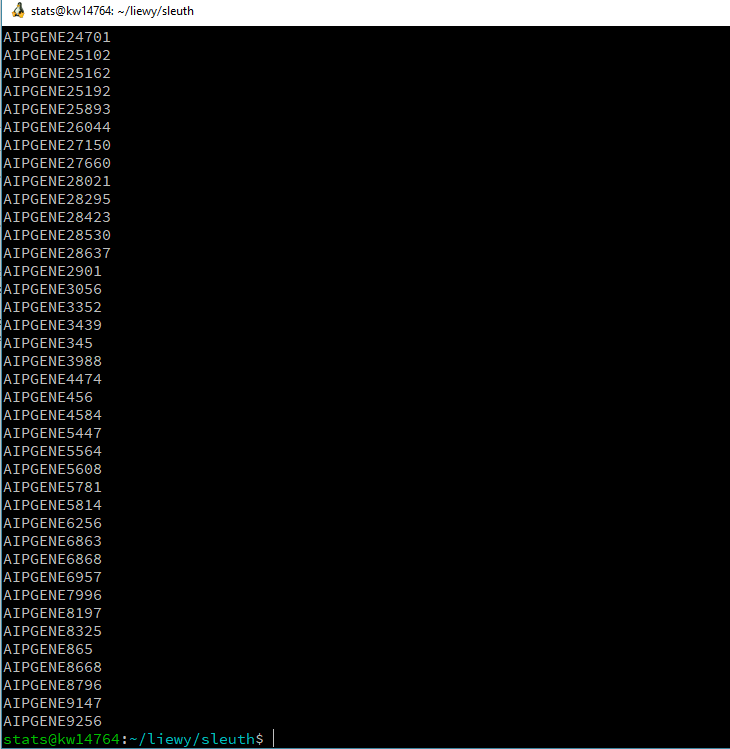


(I hope you remember what “|” was—I covered it in my morning session. Recall also the “%<%” thingy that Nate presented in day 1. This is piping—you produce some text output in the first command, which is then piped into a line-counter, “word count dash line”.)

We have 83 genes differentially expressed! Yay!

To see which genes they are, we change the command after the pipe.

**awk '{if ($7 < 0.5) print}' sleuth\_results.tsv | cut -f 1**

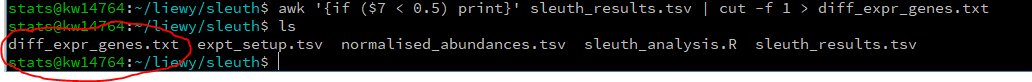


The cut command “cuts” out the first (-f 1) column of the table. Let’s save these bunch of genes into a file!

**awk '{if ($7 < 0.5) print}' sleuth\_results.tsv | cut -f 1 > diff\_expr\_genes.txt**

And let’s verify that the file exists.

**ls**



Let’s get back to the previous folder

**cd ..**

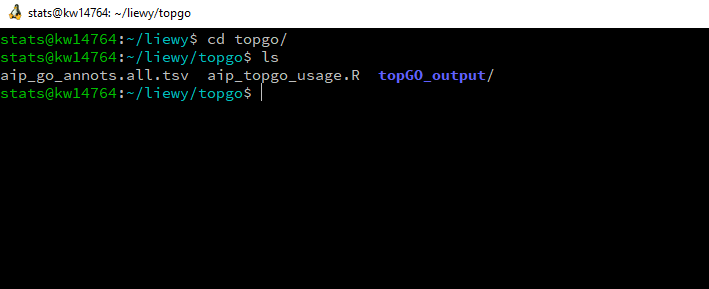
Done!

1. Finale: Running a GO term analysis to find enriched GO terms

Enter the topgo folder, and look around.

**cd topgo**

**ls**

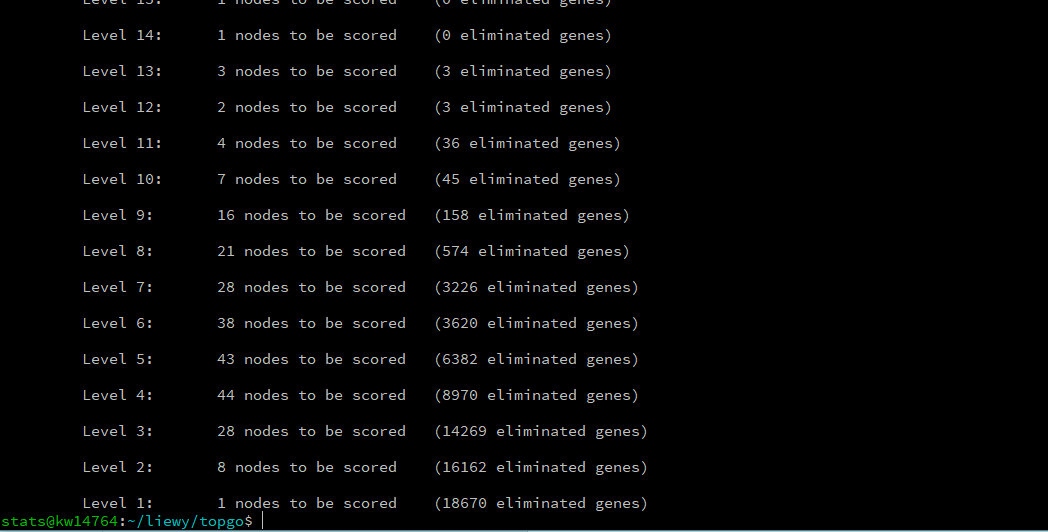


Again, I have made life easier for you. I have modified the R script needed to do this section as “aip\_topgo\_usage.R”.

So... run it :)

**Rscript aip\_topgo\_usage.R**

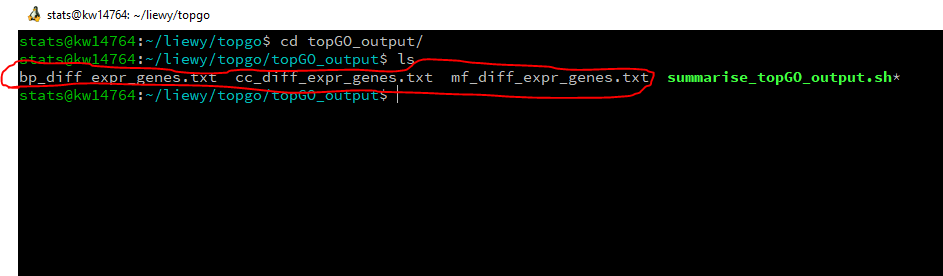
This process takes a while, so toilet break #2!



You should see this as the script ends. The script produces a few files in the folder topGO\_output.

**cd topGO\_output**

**ls**



The circled files were produced from the R script. Feel free to look at them using **less**. If you do, remember to press **q** to quit.

To process these files, run the shell script in the same folder.

EITHER

**source summarise\_topGO\_output.sh**

OR (the lazier way)

**./summarise\_topGO\_output.sh**

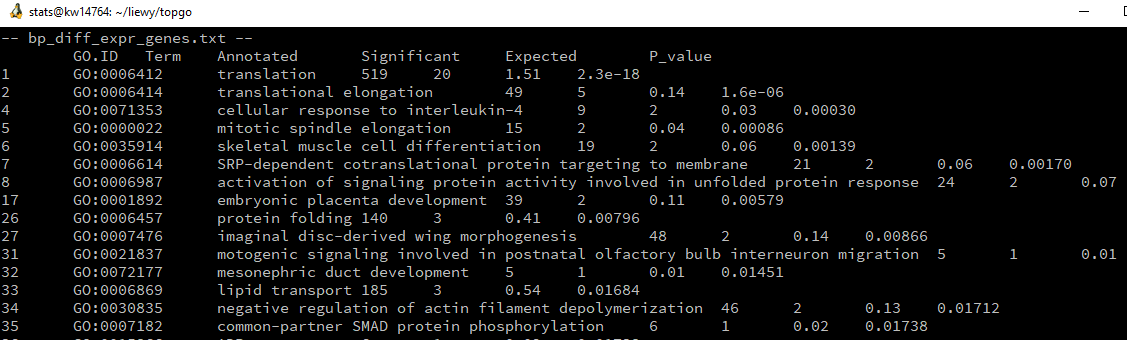
The script produces an additional file.

**ls**



Let’s check out the summary file.

**less summary\_diff\_expr\_genes.txt**



Thus, we can conclude that heat-stressed genes tend to be translation-related / translation-elongation-related. It could perhaps be that under heat stress, there is an increased expression of chaperone genes, which in turn, aid in the correct expression and folding of proteins.

1. **Conclusion**

This is basically how I’d carry out a transcriptomics analysis.

The analysis results in a list of GO terms that describe what sorts of genes tend to be differentially expressed. Use this list to guide you in designing future experiments. Some people use this list and basically write it up as a paper, a practice which I generally dislike, because of the absence of experimental proof.

Please DO NOT trust the results of today’s practical—remember, we used p < 0.5 to decide whether a gene was differentially expressed, just to squeeze out something for the later steps. If you publish results with p < 0.5, you deserve all the scorn you get from your reviewers! :p

I hope you enjoyed the ride! If you want more info about the GO term analysis, check out

<https://github.com/lyijin/topGO_pipeline>

I have not written up the kallisto/sleuth bits, but if I ever do, it’ll be on my github.