homework 07: Wiggins' lost labels

You're still sorting out the mess that poor Wiggins left behind when he quit the lab. In another set of experiments he was doing, it looks like he had isolated a mutant sand mouse in a screen, and characterized the mutation's effect on overall gene expression in RNA-seq experiments. He did two sets of three replicates on wild type control mice, and one set of three replicates on his mutant sand mice. But unfortunately, he mislabeled the files and lost track of which was which. He's still not responding to your emails, so you're going to have to do some more detective work.

Wiggins data and scripts

You've found Wiggins' three data files, each of which contains mapped count data for three replicates each: w07-data.1, w07-data.2, and w07-data.3. Two of these files are from wild type samples, and one is from the mutant samples.

You've also found the R script he apparently used to do his analysis: analyze_W.r. His analysis pipeline used the well-regarded edgeR package for differential gene expression analysis. His R script takes an input file (called `mydata.tbl`, imaginatively) with 6 samples, three wild type and three mutant. He must've put two of the data files together to do a differential analysis of 3 wt vs. 3 mutant samples, but his merged input file has been lost.

His notebook says he identified 2107 differentially expressed genes significant at P<0.05.

Your task is to reproduce his work, figure out what

the missing labels on those files are, and see if you believe his result of 2107 significant differentially expressed genes.

install R, BioConductor, and edgeR

Differential gene expression analysis methods are almost invariably written in R and made available as BioConductor packages, including all methods considered to be reasonably current standards: edgeR, DESeq2, limma, EBSeq, and Sleuth. Of these, I chose `edgeR` semi-arbitrarily as our example to look deeper into.

This is an example of how it's useful to be able to do biological data analysis in both Python and in R. A goal of this homework is to install and use some R code, to see it in action, though without really learning it.

First, install R, if you don't have it on your system already. (Linux systems typically do. Type the command `R` to check.) Use one of the precompiled packages that the R team provides. For example, if you are on OS/X, you can download and install R-4.1.1.pkg.

Once you've done that, typing `R` will start the R command line environment; the command `q()` will quit, after it asks if you want to save your current R workspace:

```
% R

R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical C
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRA
You are welcome to redistribute it under certain conc
Type 'license()' or 'licence()' for distribution deta

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in public

Type 'demo()' for some demos, 'help()' for on-line he
'help.start()' for an HTML browser interface to help.
```

```
Type 'q()' to quit R.

> q()
Save workspace image? [y/n/c]: n
```

Start `R`. At the R command line, use BioConductor to install its `BiocManager` installer, then the `edger` package:

```
% R
> if (!requireNamespace("BiocManager", quietly = TRUE
> BiocManager::install("edgeR")
> q()
```

You can find more information at BioConductor's pages for the edgeR package, or for BioConductor itself.

You should be all set. If not, see the hints section for more information about installing R and edgeR.

but we're using Python

Part of the point of this homework is to learn a poor hacker's way of working in two different languages, such as Python and R: communicating through files. We can have our Python script write out an R script and its inputs as files, run the R script (for example using Jupyter Notebook to run `Rscript` as an external command) to produce output file(s), then parse the output datafile back into our Python flow.

Alternatively... we could just do our work in R! But Python's been plenty for us to tackle this semester, right?

1. write a python function to run an external edgeR analysis

Write a python function that takes the name of an input counts file as an argument (and any other arguments you need), and returns the results of an `edgeR` analysis: gene names, log fold changes, log CPM, P-values, and FDRs (and any other data you find you want to return).

You can use the poor hacker's style: have your python script write an R script to a file, run the R script externally with `Rscript` (see hints), and parse the resulting output file.

Use the same edgeR analysis steps that Wiggins' analyze_W.r script used.

2. reproduce Wiggins' data, assign the missing labels

There are three possible combinations of Wiggins' data files: (1,2), (1,3), or (2,3). Using your Python function for running his edgeR analysis, run all three analyses.

Which combination did he run to obtain his result of 2107 differentially expressed genes significant at P<0.05?

Which of the three files corresponds to the mutant sand mouse samples? Why?

3. Wiggins doesn't understand p-values

Do you agree with Wiggins' conclusion that 2107 genes are differentially expressed in that wt vs. mutant comparison? What did he fail to do?

Give a different conclusion of your own -- what is a more appropriate statistical cutoff, and how many genes are called differentially expressed at your threshold?

4. Wiggins missed something else too

Wiggins' analysis has a subtler problem. It's missing an important step in the edgeR analysis pipeline, and it just happens (!) that this example exercises exactly the problem that that edgeR step is designed to deal with. Find it and fix it -- you'll

need to add one step in the R analysis script -- and rerun the analysis. Now how many genes do you think are differentially expressed? What was the problem? (see hints)

turning in your work

Submit your Jupyter notebook page (a `.ipynb` file) to the course Canvas page under the Assignments tab. Please name your file `<LastName><FirstName>_<psetnumber>.ipynb`; for example, mine would be `EddySean_07.ipynb`.

Remember that we grade your psets as if they're lab reports. The quality of your text explanations of what you're doing (and why) are just as important as getting your code to work. We want to see you discuss both your analysis code, and your biological interpretations.

Hints

- You can install R in other ways, including with homebrew:

```
% brew install r
```

For other alternatives, see the R Project.

Make sure you have R installed and working before you install any R packages. You should be able to type `R` at the command line prompt and get into the R environment's prompt; `q()` quits.

- **Beware using `conda` to install R.** At the start of the course, we recommended installing anaconda to get a complete Python3 data science environment. In principle, you should also be able to use `conda install` to install R, BioConductor, and edgeR, using conda's `bioconda` package channel. However, one year I had all sorts of trouble with the R installation in anaconda. I ended up having

to completely delete anaconda and reinstall fresh. Now I'm terrified of ever trying `conda` again for this.

If you do try to install R and other R packages with anaconda and it works, let me know. If it doesn't, you can uninstall with:

```
% conda uninstall r-base
```

- The reason to give you Wiggins' R script is so you don't have to learn any R right now. You'll have to modify its steps in one obvious way, and for that you'll probably want to read parts of the edgeR user's guide.

- Although Wiggins' input file has been lost, in another place on his computer you do find a different mydata.tbl of six samples from a completely different experiment, in a different tissue (i.e. all the gene expression levels are different in this file - it has nothing to do with the pset data!). If you're having trouble getting `edgeR` to work, start by getting Wiggins' analyze_W.r script to work on this example mydata.tbl file. If you have that data file and Wiggins' R script in your current directory this should work for you:

```
Rscript analyze_W.r
```

and it will create an output file `myresult.out`.

This is also a good way to make sure that you have R and edgeR installed and working correctly.

- If you're on a machine with a unixy command line tools, you can use `join` to concatenate any pair of Wiggins' data files to create the input file you need for a differential gene expression analysis with `edgeR`. For example:

```
join -t $'\t' w07-data.1 w07-data.2 > merged.12
```

is an incantation that takes the two three-sample files w07-data.1 and w07-data.2 as input, and outputs one merged line for each corresponding line pair in the two files: the first field (the gene name, the *join field*), followed by the data columns of file1, followed by the data columns of file2. The `-t $'\t'` option says to output the merged data as tab-delimited data; the funny-looking `$'\t'` is a trick called ANSI quoting.

Of course if that's too weird looking for you, you can always whip up some Python to read the two input files and output the merged file you want.

- Part 4 may seem maddeningly vague, but I don't want to totally give the 'puzzle' away either; so here's some hints.
- Look at the log fold changes for the most significantly changed genes; this might give you an idea of what's going on.
- Consult the edgeR user's guide.
- Remember everything we've said all through the course about what RNA-seq experiments measure in an RNA population.
- You could do a differential expression analysis of all six wt samples against the three mutant samples, of course -- though I'm not asking for that in the pset, because the questions are semi-artificially arranged around analyses of three replicates each of wt and mutant.
