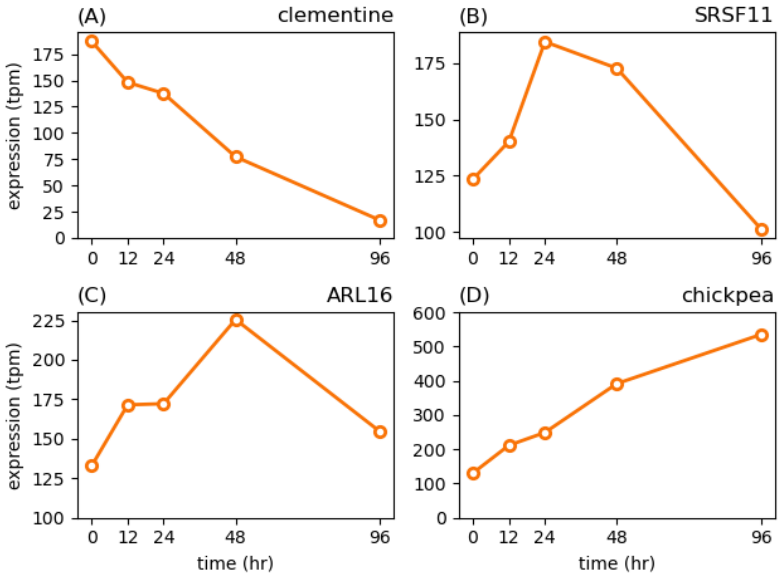
pset 01: the case of the dead sand mouse

You've just joined the Holmes lab. A senior postdoc in the group, Moriarty, recently published a paper that described what he calls the "thanatotranscriptome" of the sand mouse. Moriarty *et al.* says that a large number of genes in the prefrontal cortex of the sand mouse are differentially upregulated *after the mouse dies*.

the experiment

Moriarty's methods section says that he instantly kills a sand mouse at time t=0. At t=0, 12, 24, 48, and 96 hours, he dissects out the prefrontal cortex, prepared poly-A+ mRNA, and does RNA-seq. An example of the results is shown in Figure 1 from Moriarty *et al.*: *clementine* goes down after death as you might expect from being dead... but *SRSF11* comes up at 24hrs then goes down, *ARL16* peaks at 48h, and *chickpea* is still coming up at 96h after death:



In his discussion, Moriarty says that these data

provide evidence for an ancient program of cortical gene expression that causes the sand mouse's life to flash before its eyes (very slowly).

Well, maybe. But you suspect an artifact of some kind. You decide to look into the result. After carefully reading the paper, it looks like there aren't any obvious problems with the RNA-seq experiment itself. Each data point is an average over several dead sand mice, and the variation seems small, so it's not that this is just experimental noise. The dissections seem carefully done, so it's not that he mistakenly collected anatomically different chunks of brain (which could easily give gene expression differences). He also did some controls to make sure that the relative proportions of different cell types in the dissected tissue hadn't changed much (a common problem in differential RNA-seq analysis of tissues composed of mixtures of cell types: you measure a population average, so if cell type composition changes, it can look like gene expression changes.) Everything suggests to you that the expression level (TPM) calls are reproducible and robust, so if there's a problem, it's in his interpretation of the expression levels.

the data

The expression data for the paper are available in his Supplementary Data Table 1: Moriarty_SuppTable1.

It happens that you've also just been reading another paper from Irene Adler's lab that seems relevant. Adler *et al.* systematically measured mRNA synthesis rates (in mRNA/hr) and mRNA halflife (in hr), also in the prefrontal cortex, for every gene in the sand mouse. Those data are available in their Supplementary Data Table 2: Adler_SuppTable2.

Download both data files, which you'll need for the exercises below.

1. check that the gene names match

In principle, the gene names in the two data files ought to match, but (sigh) you know that all kinds of things can go wrong in practice: people use different names for the same gene, spell them with different capitalization or punctuation or added spaces, and so on.

Write a Python script to compare the gene names in the two data files. Output the names that appear in Moriarty_SuppTable1 but not Adler_SuppTable2, if any. Use plain Python for this pset. Don't use Pandas, if you already happen to know it. One of the points of the pset is to be able to ingest crappily-formatted data files, by writing your own parsing code.

You're especially suspicious of the Moriarty *et al.* data table because he mentions in his paper that he exported the supplementary methods tables from Microsoft Excel, and you've run across people on Twitter discussing not just one but two terrifying papers about how Excel has systematically corrupted the supplementary data files in many published articles.

If there's a difference - why?

2. explore the data

Write basic Python code to:

- output the five genes with the highest mRNA synthesis rate. (i.e. in Adler_SuppTable2)
- output the five genes with the longest mRNA halflife. (i.e. in Adler_SuppTable2)
- output the five genes that have the highest ratio of expression at t=96 hours post-mortem vs. t=0 (i.e. in Moriarty_SuppTable1)

(You might want to explore the data in other ways too, on your own, as you're doing the next part.)

3. figure out what happened

This is partly an exercise in manipulating data files line-by-line in Python, but it's also designed to give you a Very Large Clue as to what happened in the dead sand mouse experiment.

Write a Python script that merges the two data files, line by line, merging them on gene name. That is, for each line in file 1 for gene X, find the corresponding line for gene X in file 2; we're going to write a single output file with one line per gene. The genes are in different orders in the files, so this merge isn't *entirely* trivial. For any gene name X that isn't found in both files (*cough cough* Excel corruption *cough cough*) just skip it. For each gene name that is found in both files, output one whitespace-delimited, column-justified data line consisting of 7 fields per line:

- gene name
- Four expression ratios relative to t=0: i.e. tpm[12h]/tpm[0], tpm[24h]/tpm[0], tpm[48h]/tpm[0], tpm[96h]/tpm[0], by processing the TPM data in Moriarty_SuppTable1
- RNA synthesis rate (in mRNA/hr) and mRNA decay halflife (in hr) from Adler_SuppTable2

Save your merged dataset to a file.

Merging data sets, even crudely like this, is often useful as a step towards exploring data and looking for problems, outliers, and correlations.

Explore the data, however you want, for example by looking at the genes with the highest expression ratio t=96/t=0. What do you think is the real explanation for what happened in the dead sand mouse experiment?

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_01.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.

---

notes

- Figure 1 is an example of a data presentation practice that can be deceptive if you're not careful about it: the y-axes are different in all the subplots, and they don't all start at zero. This risks visually exaggerating small relative differences. For example, the differential expression of *SRSF11* isn't as impressive as the graph may make it appear; it goes from 123 TPM at t=0 to 185 TPM at t=24h, only a 1.5x expression difference. Edward Tufte, a guru of data visualization, has written about this and other common problems in graphs.

- A somewhat related classic is the IgNobel prize-winning fMRI study of a dead Atlantic salmon. (Containing a deadpan Methods section: "The salmon was shown a series of photographs depicting human individuals in social situations. The salmon was asked to determine what emotion the individual in the photo must have been experiencing.")

---

hints

- The python script that Moriarty *et al.* used to make Figure 1 is available in his Supplementary Methods: figure1.py. You

could use it as a starting point for parsing the Moriarty_SuppTable1 file. It also gives you an advance sneak peek at using matplotlib to plot data.

- Here's essentially the same script in a Jupyter Notebook page, which you can download and open in Jupyter Notebook, and you can compare/contrast a Notebook page to a command line script.

- The first hundred sand mouse gene names in Moriarty_SuppTable1 are fictitious (I chose vegetable and fruit names) but the other 19,931 are the names of all the protein-coding human gene names in the GRCh38 human genome annotation. If you want, you can get that annotation from here for a GRCh38 human genome assembly as a gzip'ed GTF (gene transfer format) file. It's fun to play with. Here's a starting point: the python script I used to pull the gene names out. For a fabulous set of tools for manipulating genome annotation coordinates, look for bedtools.

- **Super useful**: Though we want you to use basic Python for your pset answers, be aware that you can do a bunch of this week's work even more easily at a unix-y command line. This is often true for simple data manipulations, especially on simply formatted files that can be dealt with by line or whitespace-delimited field. For example:

```
# Remove comment lines from a data file:
% grep -v "^#" Adler_SuppTable2

# Sort a data file on the numbers in field #3, ig
% sort -n -k3 Adler_SuppTable2

# Combine the two: ignore comments and sort the r
% grep -v "^#" Adler_SuppTable2 | sort -n -k3

# Sort non-comment lines in alphabetical order
% grep -v "^#" Adler_SuppTable2 | sort

# Look at the first 10 lines of a data file:
% head -10 Adler_SuppTable2
```

```
# ...or the last 10:
% tail -10 Adler_SuppTable2

# Permute the order of the lines of a file:
% shuf Adler_SuppTable2

# Take a random subset of 10 lines from a file:
% shuf Adler_SuppTable2 | head -10

# Get a sorted list of names from each file into
# look for names unique to file1 or file2. (i.e.
% grep -v "^#" Moriarty_SuppTable1   | awk '{prin
% grep -v "^#" Adler_SuppTable2 | awk '{print $1}
% comm -3 foo baz

# Output five genes with highest mRNA synthesis r
% grep -v "^#" Adler_SuppTable2 | sort -n -r -k2

# Output five genes with highest mRNA halflife (s
% grep -v "^#" Adler_SuppTable2 | sort -nr -k3 |

# Output the highest five tpm[t=96]/tpm[t=0] rati
% grep -v "^#" Moriarty_SuppTable1   | awk '{prin
```

Examples of frequently used UNIX power tools on the command line:

- **head**, **tail**: take first (last) lines of a file
- **sort**: sort a file on any field
- **shuf**: shuffle lines of a file
- **grep**: select for (or against) lines that match a pattern
- **sed**: apply search/replace pattern to each line
- **awk**: ancient, powerful lore; an original scripting language.

On OS/X, `shuf` may not be installed for you, and other programs may exist but be somewhat braindead. Install GNU coreutils. For example, `brew install coreutils`, if you use Homebrew as a package manager.

To get brief help on using any of these, use `man`: i.e., `man sort`.
