Assignment 4

Gene Expression
Overview

- **Due:** by end of 2/18/2022
- **Start early, this assignment may take some time**

Look for TODO comments if you forget what things we want you to do or comment
Tips and Tricks: FUNCTIONS ARE YOUR FRIENDS

● Functions allow you to re-run the same code over and over again
  ○ Computer scientists are lazy and you should be too with your code!
  ○ Why copy paste if you can just call a function?
● You can provide functions with arguments to make them versatile
  ○ If only one small thing (like the data) is changing, use function arguments to run the same code on different objects
● When debugging functions help in determining which part of your code is breaking

As a general rule: Functions are really good at one thing, break up your steps into functions

TIP: You can write more functions than we specifically ask for if it is helpful to you

● Some simple tutorial for function in Python: [https://www.w3schools.com/python/python_functions.asp](https://www.w3schools.com/python/python_functions.asp)
An example of a function used in this assignment

```python
# Library size
# Calculate library size of each sample (e.g. sum of RNA-seq counts)
def library_sizes(dictionary, list_of_samples):
    # Get the total number of samples
    num_samples = len(list_of_samples)
    # Initialize N, a list to hold the total counts from each sample
    N = []
    # For loop to iterate over each sample
    for i in range(num_samples):
        # Append a new float zero value for each sample (goes to index i)
        N.append(0.0)
        # For loop to iterate over each value in our dictionary
        for v in dictionary.values():
            # Get the count from the i index of this gene and add it to the total for sample i
            N[i] += v[i]
    # Return the list containing each sample's library size
    return(N)
```

dictionary: the count matrix structured as {gene:[list of counts by sample]}
Tips and Tricks:

1 is different than 1.0

Int and float are different and sometimes incompatible
Premise

- Have 20 subjects who are at a high risk for developing type 2 diabetes
  - You put them on a year-long exercise regimen
- You have muscle biopsies from each subject before beginning the exercise regimen and after one year (2 time points)
- You want to characterize their skeletal muscle transcriptomes and hope to better understand the genetic underpinnings of insulin signaling and insulin sensitivity after exercise in human skeletal muscle.
Important Files

- raw_counts.txt
  - RNA-Seq count data from the muscle biopsies before and after the exercise regime for the 20 individuals
  - Look at the data to get a rough idea (How many columns? Rows? What do those columns/rows mean?)
  - Do not copy this file to your folder to save space

- gene_expression.py
  - Python script that you will need to complete
  - Read through this script, see what is completed, what do you need to do

- README.txt
  - README to answer the questions in the assignment
RNA-Seq count data from the muscle biopsies before and after the exercise regime for the 20 individuals.

Look at the data to get a rough idea (How many columns? Rows? What does those columns/rows mean?)
Part 0 - Setting up for success

- Make script exit if the number of input parameters is incorrect
- Fill in the code for the `translate_dictionary` function
  - Translate a dictionary (as input) from a dictionary of counts by sample for genes to a dictionary of counts of genes for each sample
  - `{gene:[list of counts by sample]} to {sample:[list of counts by gene]}`.
- Comment the `upper_quartile_norm` function to explain what each line does
- Create and comment a function `fishers_linear_discriminant`
  - Remember that functions should be able to work using different data sets, so make sure that your function would work with data from a different experiment with different numbers of before and after samples.
Part 1- Data filtering

1. Remove genes that have zero counts in all samples
   a. Create a dictionary with genes that pass your filter
   b. TIP: create a new dictionary for each filtering step
   c. **DO NOT ALTER THE ORIGINAL DATA FILE**

2. Calculate the counts per million (cpm) of each gene left in your data
   a. Use the `counts_per_million` function to your data that passed the filter

3. Create a dictionary of genes that pass your second filter
   a. Remove genes that have 20 or more samples have cpm < 1
   b. **NOTE:** this dictionary should be a dictionary of raw counts, not cpm. You are just doing the filtering based on cpm.
Part 2 - Data visualization

1. Plot the library sizes (total counts) for each sample using the genes that passed your filters in Part 1
2. Use the matplotlib library in python to plot things
3. Make sure your plot has labeled axes!
4. Save this plot as library_size.png
Graphing in Python

https://matplotlib.org/gallery/index.html

https://matplotlib.org/api/pyplot_summary.html

Or others if you are more comfortable with:

Pandas
etc...

Ask Google about detailed questions : )
Part 3 - Data normalization

1. Use the `upper_quartile_norm` function to normalize the data you have from Part 1

2. Plot the normalized library sizes (total counts)
   a. TIP: Look at your code from Part 2 to make this easier!
   b. (optional) TIP: Make a function to plot your data
Part 4 - Data exploration

1. Use Fisher’s Linear Discriminant (FLD) to identify genes that are differentially expressed between the Before and After groups
   a. Use the FLD function you wrote in Part 0
   b. Output the genes with the ten highest FLD values (include gene name and FLD values)

NOTE: when calculating FLD for each gene, remember to split the expression values into group 1 (Before) and group 2 (After)
What to turn in

- Edited script: `gene_expression.py`
- Output files
  - `library_size.png`
  - `library_size_normalized.png`
  - `mean_expression.png`
- Your `README.txt` with the answers to the questions and the commands you used to answer the questions
- Extra credit only: `dendrogram.png`

**REMEMBER TO COMMENT YOUR CODE**

*(for accurate grading and for your future self!)*
Suggestions

1. Submit assignment on time
2. Discuss questions with classmates (but no copying)
3. Google is your best friend!
4. Post questions on Piazza