Before generating random numbers, make sure to set a random seed so your results will be reproducible.

Problem 1

Normalization under a simple generative model

Here is a generative model for the number of RNA molecules observed under a compositional framework,

$$X_j \sim \text{Multinomial}(N_j, p_j)$$

- j indexes the sample (e.g. each mouse or human where the sample was created from).
- N_j an integer. The total number of molecules sampled from the pool.
- X_j a vector in $\mathbb{R}^d$. The number of molecules for d genes in sample j. In this example there are 500 genes. In your typical human example you would expect about 12,000 genes to be expressed.
- p_j a vector in $\mathbb{R}^d$. The *true* relative abundance of d genes.

You can think of it this way: you have a bag of infinite RNA under the *true and unobserved* proportion p_j. You decide to pull out N_j RNA molecules from your bag. The number X_{jg} is the number of molecules you pulled out that are from gene g.

Note, that when a normalization procedure is working, you would expect most points to approximately sit on the $x = y$ line, so use that information where relevant in this homework.

(a) Read `p.tsv` as the vector p_j. Make 3 simulations of X_j with $N_j = \{15000, 30000, 150000\}$ where $j = (1, 2, 3)$. Your final result will be a $d \times 3$ matrix with columns summing to N_j. Make a scatterplot of the samples with 15000 molecules (X_1) versus the sample with 150000 molecules (X_3). We refer to these as the *raw counts*.

(b) Write a function that implements the DESeq-style normalization where the input is a $d \times N_{\text{samples}}$ matrix ($N_{\text{samples}} = 3$ in (a)). First, define X_{jg} as an integer representing the number of molecules of gene g in sample j. Then, for each gene within each sample, we have the value $\hat{s}_{jg}$:

$$\hat{s}_{jg} = \frac{X_{jg}}{\left(\prod_{k=1}^{N_{\text{samples}}} X_{kg}\right)^{1/N_{\text{samples}}}}.$$

Note, this function is not defined when any of the numbers in the denominator are zero. Filter the data accordingly.

Then, the *sample specific* normalization factor is:

$$\hat{s}_j = \text{median}_g(\hat{s}_{jg}).$$

Keep track of the entire distribution of $\hat{s}_{jg}$ along with the median for each sample. You will need this later.

(c) Make histograms of the $\hat{s}_{jg}$ values for samples $j = \{1, 3\}$ (I'm asking for two histograms). How do they compare? Additionally, making scatterplots of the values might help illuminate some things. Finally, report $\hat{s}_j$ and interpret it.

(d) Normalize values by dividing the observed data by the normalization factor as follows:

$$Y_j = \frac{X_j}{\hat{s}_j}.$$

Note that $\hat{s}_j$ is a scalar and X_j is a vector. Now make a scatterplot of Y_1 versus Y_3. How does it compare to your result from (a), the scatterplot of X_1 versus X_3.

(e) Set $N_j = \{1e6, 1e6, 1e6\}$ and redraw three samples from the generative model. Estimate $\hat{s}_{jg}$ and $\hat{s}_j$ using this matrix. How do your results compare to your result from (c)? Is this intuitive?

(f) Load `q.tsv` as the vector p_j for this next part. Simulate $X_j^{(q)}$ using $N_j^{(q)} = \{1e6, 1e6, 1e6\}$. Again, how dos your $\hat{s}_{jg}$ and $\hat{s}_j$ results compare between the samples and compare to the samples in (c) and (e).

(g) Finally, make a $d \times 6$ matrix by concatenating the matrix from (e) and (f). Run your normalization procedure again to produce $\hat{s}_{jg}$ and $\hat{s}_j$. Compare your result here to the results in (c), (e), (f). What happened and why? If you are totally confused, something you can do is compare the distribution of p and q. Finally, compute the normalized values (Y_j) and make a scatter plot of the first sample and the final sample ($j = (1, 6)$). To be clear, this is plotting one sample from the abundance distribution p and the other from q.

Supplementary details

This stuff isn't important unless you are interested. If you are, read (and rock) on.

p and q are probability distributions (obviously) that I made up. However, they have some interesting properties. The first distribution, p, is drawn as:

$$p \sim \text{Dirichlet}\left(\mathbf{1}_d\right).$$

Dirichlet is one of my faves. It's a distribution *on distributions*. Whoaaa. The mean value for each individual entry of p is $1/d$ and the variance is something... ugly. Look it up. Interestingly, if you were to sample from p then from X_j infinitely many times, you should get that a mean gene expression over all your samples of N_j/d (assuming the same N_j).

Next, q is derived from p using this procedure that seems crazy, but isn't so crazy if you look carefully:

```
n_different = round(n_genes * 0.4)
logfc = abs(rnorm(n_different, 1.2, 0.5))
```

```
logfc = logfc * sample(c(-1, 1), length(logfc),
  replace = TRUE, prob = c(0.1, 0.9))
# make a few really gross
logfc[1:5] = 4

is_different = 1:n_different

q_j = p_j
q_j[is_different] = exp(log(p_j[is_different]) + log(2^logfc))
q_j = q_j / sum(q_j)
```

Cool, huh?!