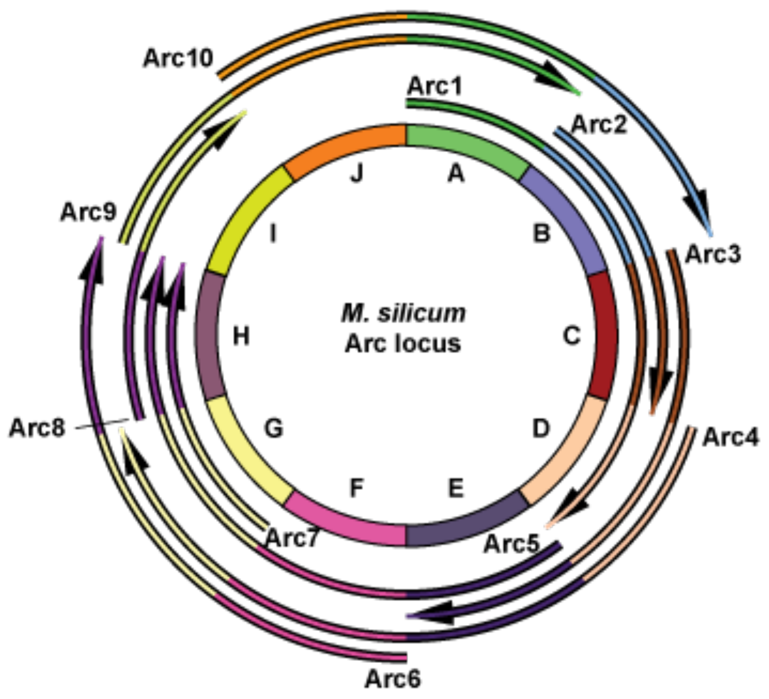
homework 09: the return of the ten Arcs

You fondly recall the circularly permuted structure of the *Arc* locus from w02, which expresses ten overlapping mRNA transcripts called Arc1 through Arc10. The *Arc* locus consists of ten segments (labelled A through J). Each of the ten Arc mRNA isoforms starts at a corresponding segment, and is 2-4 segments long.



The sand mouse *Arc* locus consists of ten segments A-J. Starting at each segment, it is transcribed into ten overlapping mRNA transcripts, Arc1-Arc10, each of which is 2-4 segments long.

These overlaps made it nontrivial to estimate the abundance of the individual Arc mRNA isoforms by RNA-seq. A recent paper by Lestrade et al. in the *Sand Mouse Journal* collected an RNA-seq dataset and made an attempt with their own program.

we'll use abstract "reads"

For the purposes of this exercise, we're going to abstract the "sequencing reads", as follows.

There are just 10 different read "sequences" a..j, corresponding to which segment A..J the read maps to. The length of each transcript isoform i, L_i, is 2-4, measured in segments; for example, Arc1 has length $L_1 = 4$, and it can produce four different reads a,b,c,d corresponding to its four segments A,B,C,D.

We'll also assume that there is no mismapping (no basecalling error), so when we observe a "read" a-j, we know it maps to the corresponding segment A-J.

With this abstraction (which saves us from working in detail with actual sequences and sequence mapping, and lets us work in just the essentials!), we are given read counts r_k for reads $k = (a, b, c \ldots j)$, and we're going to use expectation maximization to estimate the unknown abundances of the 10 Arc mRNA isoforms. Our notation and description will parallel that of Li and Dewey (2010). Our goal is to work through the main contribution of that paper, which is to be able to deduce unknown transcript abundances from observed read counts, even in the presence of multimapped and mismapped reads.

the structure of the Arc locus

Lestrade *et al.* provide a supplementary data file that contains information about the structure of the Arc locus (same as shown in the Figure, above right):

Isoform	Length	Sequence
Arc1	4	ABCD
Arc2	2	BC
Arc3	3	CDE
Arc4	4	DEFG
Arc5	4	EFGH

Arc6	3	FGH
Arc7	2	GH
Arc8	2	HI
Arc9	3	IJA
Arc10	3	JAB

So, for example, segment A appears in three different transcripts Arc1, Arc9, and Arc10, so even if our read sequences were perfectly accurate, we wouldn't know whether a read 'a' came from Arc1, Arc9, or Arc10. It came from one of them, proportional to their relative abundances, but their relative abundances are exactly what we're trying to estimate from the read counts!

The supplementary data file also provides the observed data: mapped read counts r_k from an RNA-seq experiment that collected a total of 1 million mapped reads:

read sequence	**counts**
a	111446
b	76074
c	63285
d	124405
e	126593
f	123914
g	125104
h	89216
i	69544
j	90419

Lestrade *et al.* provide the script they used to estimate the expression levels of each Arc isoform. Their analysis follows from the method called `rescue` in Li and Dewey (2010). First each read x is assigned to its cognate segment X, then read counts are assigned to possible isoforms uniformly: i.e. a read 'a' maps to segment A, and 1/3 count is given to Arc1, Arc9, and Arc10, the three isoforms that share segment A. This gives them estimates of nucleotide abundances ν_i for each transcript, which they convert to transcript abundances τ_i. (If multiplied by 10^6, the transcript abundances would be in TPM, but converting to TPM doesn't make sense for only 10 transcripts; let's just leave them as fractional transcript abundance.)

The τ estimates resulting from the Lestrade *et al.* analysis are:

isoform	$\hat{\tau}_i$
Arc1	0.096
Arc2	0.072
Arc3	0.108
Arc4	0.120
Arc5	0.106
Arc6	0.097
Arc7	0.083
Arc8	0.088
Arc9	0.120
Arc10	0.111

Lestrade's conclusion is that all 10 mRNA transcripts are expressed at about the same level, within about two-fold of each other.

Lestrade has a reputation as a well-meaning but somewhat bumbling investigator. You recognize that Lestrade's estimates could be quite poor, because his inference method is not sound. We need something like EM, the approach described by Li and Dewey (2010), to get this right.

further explication

We need to set up some more technical explication before we get to the exercise.

First, let's make a clear statement of the generative probability model we're specifiying here: an abstract and simplified version of the model in Li and Dewey (2010).

We have two random variables: the transcript isoform T, and a segment S that an observed read maps to.

To generate one observed read, we assume there are two steps.

- Choose a transcript isoform i with probability ν_i, according to the nucleotide abundances; i.e. $P(T) = \nu$.
- Given isoform i, we choose one of the segments $j = 0..L_i - 1$ of T_i uniformly -- i.e. for isoform T_i of length L_i, we sample its segments with probability $\frac{1}{L_i}$.
- Given that a read emanates from segment j, it will be correctly mapped to an observed read $k = j$. (I could introduce a basecall error model and a mismapping rate, to stick closer to the Li and Dewey model, but I've kept it simplified.)

Each of the N total reads is sampled independently one at a time by this process.

Therefore the joint probability is assumed to be factorized as:

$$P(S, T \mid \nu, L) = P(S \mid T, L) P(T \mid \nu)$$

Remember that the ν here is an mRNA abundance in *fraction of nucleotides*, not *fraction of transcripts*. The notation in Li and Dewey (2010) uses θ to make a distinction between θ and ν because they include a 'noise' isoform θ_0 in their generative model, to model getting unmapped reads that don't map back to the transcriptome at all. We don't have that step in this exercise.

Although the probability model is working with nucleotide abundances ν internally, reported expression levels (the output of the Lestrade analysis, for example) are transcript abundances. As usual, you'll need to keep nucleotide versus transcript abundance straight in your analysis. Recall that:

$$\nu_i = \frac{\tau_i L_i}{\sum_j \tau_j L_j}$$

$$\tau_i = \frac{\nu_i}{L_i} \left(\sum_j \frac{\nu_j}{L_j} \right)^{-1}$$

To do the expectation step of expectation maximization, you're going to need to obtain $P(T \mid S, \nu, L)$: i.e. if I show you one observed read, which isoform do you guess that it came from? You will assign each observed count proportional to your current estimate for the $P(T \mid S)$ distribution. After you do this for all the reads r_k, you have an expected read count c_i assigned to each isoform i.

To do the maximization step, you're just going to normalize your expected counts c_i to get a new maximum likelihood estimate for ν_i.

OK, now we should be ready, in principle. You may also need to refer both to Li and Dewey (2010) and to this week's lecture notes.

1. write a simulator as a positive control

Write a function that simulates N=1000000 observed read counts, following the model specified above, for any Arc locus structure (i.e. lengths L_i) and any transcript abundances τ. (You can assume that there are 10 segments and isoforms, though my version of this script will allow that to vary too.)

Use your function to generate a test data set for known model parameters τ, L that you've chosen.

2. calculate the log likelihood

Write a function that calculates the log likelihood: the log probability of the observed data (the observed read counts r_k) if the model and its parameters were known (i.e. τ, L), for a given locus structure of Arc. Explain the steps of your calculation.

Calculate and show the log likelihood of one of your generated test data sets, for your known parameter values.

Use Lestrade's approach (and his code, if you like) to estimate abundances of each Arc isoform in your test data set. Compare to your true τ. (Terrible, right?) Calculate and show the log likelihood given what Lestrade would estimate for the τ, compared to the log likelihood for the true τ in your positive control.

You should observe that the true τ parameter values give a much better log likelihood than the Lestrade et al. estimates, because the Lestrade et al. estimates are poor.

3. estimate isoform abundances by EM

Write a function that estimates unknown isoform abundances τ_i for each isoform Arc1..Arc10, given read counts r_k and the structure of the Arc locus including the lengths L_i, using expectation maximization.

Apply your function to the data in the Lestrade et al. supplementary data file. Show your estimated τ_i.

What are the most abundant two transcripts, and how much of the population do theyaccount for? What are the least abundant two transcripts? What do you think of Lestrade et al.'s conclusion that all 10 mRNA transcripts are expressed at about the same level? Explain why Lestrade's method's is worse.

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_09.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

- Again I'm emphasizing starting with "write a data simulation script". Many biological data analysis methods are conceptually straightforward but prone to error and confusion when you go to implement them. A strong strategy is to first think about what control experiments you're going to do to make sure that your implementation gets the right answer, if you happened to know the right answer. When we have a generative probability model in mind, as the basis of our analysis method, we can do a *very* strong positive control: we can sample synthetic data sets from our generative probabilistic model. If you can't get the right answer on ideal data (drawn exactly from the same model) you're not likely to get the right answer on real data.

- Had Lestrade et al. done a positive control like this, they would have realized that their analysis script couldn't solve this problem correctly. (Even if the name of their script hadn't already tipped them off.)

- You can't write the simulator without understanding the generative model. So by having you write the simulator first, I'm not only giving you a positive control, I'm also warming you up, making sure you understand the conceptual generative model in Li and Dewey (2010). If you're doing biological data analysis, you'll find that the exercise of writing a generator for synthetic positive and negative control datasets forces you to think in terms of generative probability modeling, and forces you to state your assumptions explicitly.

- If (when) you run into trouble implementing EM, you can use your generator to generate simpler cases to help you debug. One example is to set all the L_i to 1. Now each read is directly measuring the abundance of transcript isoform T_i. The naive analysis of Lestrade et al's script works fine in this case, and so should yours.

- When you calculate the log likelihood of the model given the data, note that you have to marginalize $P(S, T \mid \nu, L)$ over any "hidden variables" of the model, here the isoforms T, to get the likelihood $P(S \mid \nu, L)$.

- In iterative optimization algorithms like EM, it's a good idea to use your log likelihood calculation at each iteration, to track the log likelihood for each successive estimation of ν. EM is essentially a steepest-descents optimizer, so you expect to see your log likelihood improve at each step. On this particular problem, Li and Dewey (2010) proved in their supplementary material that there is only a single global optimum, so you expect the log likelihood of your final estimate ν to be at least equal to the log likelihood of then true ν -- this is an excellent condition to check. If you want some practice plotting data with `matplotlib` and friends, a plot of log likelihood versus iteration number is a good thing to be looking at.

- For another example where everything is known, that you can use as an additional control that your scripts are working, see this example data file. This gives you a different Arc structure and lengths L, known τ, and observed read counts r_k from my simulator. The results of running my EM script on those data are in this example output file, including the log likelihoods. (My answer doesn't include the multinomial coefficient term in the log likelihood calculation. It's a constant with respect to the abundance parameters, so it's not relevant to comparisons of different abundance estimates. It's common for people to drop the multinomial coefficient in problems like this, leaving a log likelihood that's only correct up to a missing constant.) You can use this example to test each of your three scripts: you can check that your simulator generates similar counts r_k given these parameters; you can check that you get the same log likelihood as I do given true τ. You can also check that your EM implementation gets as close to the true τ as I do. It will probably be most useful for checking that your generator is working, because once you have your generator, you can generate any number of examples like this for yourself.

- How many iterations of EM should you do? In my solution, I just do a lot (namely, 1000). It would be more sophisticated to test for convergence. You could check that the log likelihood has stopped improving; though sometimes you'll get in situations where your parameter estimates are changing while your log likelihood is only changing gradually, because you're skating along a plateau in the likelihood surface. Another convergence test is to check whether the estimated parameters ν are changing; but then you have to define what your (scalar) measure of change for the (vector) ν is going to be, and there's different ways you might do that. This is a bit of an art form, and it's ok not to worry about it for this exercise. Our iterations are not expensive, and you can just do a bunch of iterations and beat the problem into submission, like my solution is going to do.
