UCSC BME 205 Fall 2013

Bioinformatics: Models and Algorithms
Homework 6: null models

(Last Update: 15:45 PDT 1 November 2013)

Does the ORF code for a protein?
Due 8 Nov 2013

This exercise is mainly one in exploring appropriate null models for deciding whether to accept or reject a hypothesis. It is based on the lecture given in class Better than Chance: the importance of null models.

Jonathan Trent had found an open reading frame (ORF) that was 388 codons long on the reverse strand of the gene of a protein that he studies (thermosome subunit alpha, THSA_SULSH), and he was wondering whether or not it coded for a protein as well. The putative gene product of the ORF is tentatively named "oops". [Kagawa HK, Osipiuk J, Maltsev N, Overbeek R, Quaite-Randall E, Joachimiak A, Trent JD. doi:10.1006/jmbi.1995.0585 The 60 kDa heat shock proteins in the hyperthermophilic archaeon *Sulfolobus shibatae*. *Journal of Molecular Biology* 1995 Nov 10;253(5):712-25.]

There are several approaches possible for gathering evidence for or against the hypothesis that oops is a protein. These include such diverse methods as looking for evidence that homologous proteins exist in other species (particularly as stand-alone genes—not just reverse strands of other thermosomes), looking for codon bias in the ORF, looking for promoters in the appropriate places, doing structure predictions for the putative protein, and so forth. For this assignment we will look at only one question: how unexpected is the 388-codon-long ORF?

We also need to be explicit about what **exactly** we mean by an ORF. In some past years I've been a bit sloppy about this, and different interpretations have been used. This year, let us use the definition that an ORF of length N is a sequence of 3N bases that starts with ATG and does not have any stop codons in frame. It does not need to have a stop codon after it (if, for example, we reach the end of the sequence we are testing).

Since this is a Python-programming exercise, we will write a short program that generates DNA sequences using various null hypotheses, and estimate the p-value of finding that long an ORF "by chance" on the opposite strand of the thermosome gene.

We will need species-specific training data for our various null models. Unfortunately, *Sulfolobus shibatae* has not been fully sequenced, but the very closely related species *Sulfolobus solfataricus P2* has been sequenced. Furthermore, using tblastn to search for homologs of oops reveals that *S. solfataricus* has a 95% identical ORF (with no stop codons) that is at least as long opposite one of its thermosome genes, so we can do the analysis in *S. solfataricus*. The genome is available on the web S.solfataricus.fa.gz and on the SoE machines as /soe/karplus/.html/bme205/f05/S.solfataricus.fa.gz

A big part of the point of this exercise is to point out the importance of choosing the right null model. The hypothesis we are testing is that the ORF is a protein-coding gene—that the absence of stop codons has been selected for through evolution by the need to make the full-length protein product. Our null hypothesis is that no such selection is being done on the ORF.

Here are four null models corresponding to different versions of the null hypothesis:

1. The DNA is undergoing no selection at all, or selection only for G+C content.

   The 560*3 bases of the sequence are chosen at random (using a 0-order Markov chain) from the distribution of bases in the genome of the species. This model is the most commonly used null model for determining how unlikely an ORF is. Since the frequency of G is the same as C, and the frequency of A is the same as T, this model has only one parameter—the G+C frequency in the genome. You could look up the G+C frequency on sites like http://www.fut.es/~debb/HGT/ssol.d/welcome.html, but why should we trust someone else's measurement? You have a k-mer counting program, so check the genome yourself. (When I did, I got a slightly different GC content than the site above, since their counts were only for coding regions!)

   For this model, there is no point to doing simulations, as one can explicitly compute the probability of A, T, G, followed by 387 codons that are not TAG, TGA, or TAA for any series of 388 codons. How many such windows are there on the double-stranded genome? What is the expected number of windows that are ORFs? Note that we don't have to worry that overlapping ORFs could throw off our calculations, since expected values are additive, even if the random variables are dependent. In my hands, the expected number is about 7E–08.

2. The DNA is undergoing selection, but this selection is not specific to being protein coding.

   For this model, we can get a little fancier, looking into the frequency of all triple bases in the 3-strand genome. Compute $P(ATG)*(1–P(TAG)–P(TGA)–P(TAA))^{387}$ based on 3-mer counts (rather than just 1-mer counts), and get the E-value (the number of expected ORFs of that size in the whole double-stranded genome). We still need no simulation, as simple counting and calculation suffice. In my hands, the expected number is about 5E–08.

3. The DNA is on the opposite strand of a 560-codon-long protein-coding gene for S. solfataricus.

   The string of 1680 bases is constructed by taking the reverse complement of a string of an ORF of 560 codons (ATG then 559 non-STOP codons) drawn from the distribution of codons for the species. The codon preference table is easily found on the web—you can google `Sulfolobus solfataricus codon` to find this and other codon-usage tables. For a newly sequenced genome, you would first have to identify a number of genes (probably by homology to other species) and count the codons used in those genes.

   This model will require simulation, since the calculation of probabilities is not trivial. Generate 10,000 (or more) 560-codon ORFs, and find the largest ORF on the opposite strand in each one. This is a generative model that constructs a string of 560*3=1680 bases. The open reading frame could appear in any of the 3 reading frames of this string.

   Output a histogram of lengths for further analysis (you'll probably need to extrapolate to get a reasonable E-value: see below). Note that the probability of the longest ORF being 388 or longer needs to be multiplied by the number of protein genes (about 2994 proteins in *S. solfataricus*). I get an E-value of about 0.048, though I did not observe any ORFs longer than 369 in a sample of 50,000.

   This is a fairly crude analysis, since it assumes that all protein-coding genes are exactly 560 codons long. A more sophisticated analysis taking into account the distribution of gene lengths seems unnecessary, especially since we are ignoring the possibility of an ORF that starts outside the reverse-complement of a gene.

4. The DNA is on the opposite strand of THSA_SULSH.

   The string of bases is is constructed by taking the reverse complement of a series of codons that codes for THSA_SULSH, using the codon biases of S. solfataricus. That is, where the sequence of THSA_SULSH has an L, we randomly select a leucine-encoding codon, using the frequencies of the codons from the codon bias table. You should be able to find the sequence for THSA_SULSH in Entrez or Swissprot with no difficulty.

   Again we'll want a big sample of ORFs and a histogram of lengths of the longest ORFs on the opposite strand. We may need a much larger sample for this model, as we are unlikely to be able to fit a simple distribution to the histogram. I get an E-value of about 0.002, both by direct count (of 100,000 random gene sequences for THSA_SULSH) and by fitting distributions to the CDF.

The program

Write a program named reverseORF that accepts (at least) the following options:

--protein filename
: Read the protein sequence for the forward strand (in FASTA format) from the specified file (needed for the last model)

--codon filename
: Read the codon bias table from the specified file. (Use a standard format so that you can download tables from the Web.) If a codon table is specified, but no protein is specified, the program should use the random-codon model.

--num_sequences number
: The default number of sequences generated in the sample should be 10 000.

For each sequence generated, scan the three possible reading frames and determine the longest open reading frame (sequence of codons that starts with ATG and does not contain a stop codon) in the sequence. Don't make the mistake of generating all the sequences at once, then analyzing them all., as this seriously limits how big a sample you can make. Each sequence should be generated, analyzed, and discarded.

Output a histogram indicating how often each length occurred as the longest ORF. The output should have four tab-delimited columns: the length, the number of occurrences of that length as the longest ORF, the probability of that length (the second column divided by the number of samples tried), and the estimated probability of seeing that long an ORF or longer (P-value). The histogram gives a good estimate when the counts are fairly high, but for the tail of the distribution we are interested in, the estimates may be quite poor. Increasing the number of samples helps a little, but takes a long time. On my old machine, 40,000 samples took 6 minutes.

Note: a p-value of 0 makes no sense—if you never observe a long ORF in N samples, then all you can estimate directly is p-value<1/N, and even that is not very reliable. To get more accurate P-values you may need to extrapolate the histogram, by fitting some family of distributions. I tried Gumbel and lognormal, and the lognormal looks like a much better fit, particularly when matching the cumulative distribution. I don't know what family is the **right** one for longest-run statistics, but articles like "Success run statistics defined on an urn model" by Frosso S. Makri, Andreas N. Philippou, and Zaharias M. Psillakis [Adv. in Appl. Probab. Volume 39, Number 4 (2007), 991–1019. doi:10.1239/aap/1198177236] may be helpful to those who want to learn more.

I ran gnuplot from null.gnuplot with data from hist1. You should, of course, run with your own data.

Provide plots (using gnuplot or similar tool) of the histograms for the two different null models, using at least 10,000 samples for each null model (more samples will give you a better estimate, if you have the time). Interpret the results: is the 388-long ORF unlikely to have arisen by chance? Is the evidence for "oops" strong enough to justify doing lab work to look for evidence of transcription and translation of the sequence? Remember to switch from P-values to E-values before making any conclusions, as each of the null models has a different number of opportunities for forming ORFs.

You should turn in a PDF file which includes your plots and your interpretations of them. Personally, I prefer to generate such PDF files using LaTeX and the graphics package, with gnuplot to generate the plots as eps or pdf files, but you are free to use other methods for generating the PDF files.

Bonus Points

Which frame the longest ORF appears in may not be uniformly distributed among the three frames. Modify your program to get separate histograms for the longest ORF in all three frames and the longest ORF in each of the three frames. Which frame produces the longest ORFs in models that assume you are opposite a protein-coding gene? Why?

This assignment looked only at the probability of this long an ORF opposite this particular gene. A more "bioinformatic" approach would look at all the genes of the genome and figure out the E-value (expected number of ORFs this long on reverse strands of genes) for the whole genome. Does the distribution of observed ORFs deviate significantly from the expected distribution using the null model(s)? Is the ORF that Jonathan Trent found an outlier, or is it pretty typical of ORFs on reverse strands of genes?

Design other computational tests for whether the oops sequence is a gene. You may need to get the DNA sequence for the ORF, which can be found in Entrez (I found it searching the protein data base for "trent sulfolobus"). The *S. solfataricus* genome is also available on the http://archaea.ucsc.edu genome browser, with some pre-computed tracks that may be useful.

SoE home

Kevin Karplus's home page

Biomolecular Engineering Department

BME 205 home page

UCSC Bioinformatics research

Questions about page content should be directed to Kevin Karplus
Biomolecular Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
USA
karplus@soe.ucsc.edu
1-831-459-4250

318 Physical Sciences Building