Destruction by Point Mutation

Isaac Sarver

1. Introduction

The question to be answered is: “Given that a point mutation has occurred in the genetic code for a protein, what is the probability that the protein is non-functional?” This is not a question of how long will be before the protein is non-functional or an organism is inviable. Nor does it pretend to cover all manner of mutation or a genome in general. Something that is not covered but can be found in a genome is point mutations to control genes and introns.

To proceed through the rest of the paper, some terms will be required for people not familiar with genetics. Transitions are point mutations that exchange purines for the other purine or pyrimidines for the other pyrimidine. Transversions are point mutations that exchange purines for pyrimidines or pyrimidines for purines. Purines are adenine and guanine. Pyrimidines are cytosine and thymine (6, 8, 9).

I will be counting the “stop” codons as an amino acids.

One of the reasons I did this project is because I have seen the genetic code table and the many redundancies in it. I've wondered how hard it would be to break a protein by using single point mutations. There are up to six ways to code for any given amino acid. While each codon can be changed into nine other codons by a single point mutation, only four amino acids out of 21 exceed this number of things they can change into(7). I also discovered that transitions are the most common of point mutations, despite the fact that there are two transition pairs and four transversion pairs(3). In all but four cases, transitions of the third nucleotide of any given codon doesn't change the protein in any manner(7).

2. Method

2a. Assumptions

Assumptions are what make this problem computationally reasonable. Proteins are made from chains of amino acids that curl up in energetically favorably configurations. Secondary structures come from the chain curling in such a way as to establish hydrogen bonds(11). The curling goes into producing the over all structure of the protein so as to give the protein functionality. I assume that if the secondary structures fall apart due to replaced amino acids, it is very likely that the higher order structures will also fall apart defeating the functionality of the protein.

I came across the the Chao-Fasman parameters. They assume that amino acids are associated with certain kinds of the secondary structures. The parameters were originally determined in 1978 when the sample size was in the region of ten's and computational power was short to come by(2). I found that others have since redone the analysis and found other values for the parameters. I used values that were computed in 2002 from a sample size of 1,077 proteins(3). Even so, the predictive power is estimated to be 40% accurate(2).

I had to figure out how likely each kind of point mutation are. I found a publication comparing human and chimpanzee genomes. Here I make either of two assumptions; which ever suits your fancy. One can assume that evolution is true and therefore humans and chimpanzees came from the same critter and so point mutations in the respective genomes are comparable. Or, one can assume that creation is true, and if something works don't fix it. Thus, the regions of the two genomes that are comparable, it is because they started out the same. In either case, the conclusion is that the two genomes are in fact comparable and the point mutations that are found are reasonable measures of the probability that a certain kind of mutation will happen(3).

For reasons unknown, the authors of the above publication combined transversions between guanine and thymine and between adenine and cytosine. I had to assume an equal distribution between the two because I didn't have any other way to split the two. While understandable, both transitions where also combined and so similarly divided(3).

The algorithm didn't determine that all mutations which introduced or removed “stop” codons didn't break the proteins. I am fairly certain that doing so breaks the proteins. I suspect that it didn't come that conclusion had something to do with the fact that break codons look exactly like proteins without any particular secondary structure. So I manually determined that all break codon mutations break the proteins.

I had initially started with a equal weighting of environments between isolation, changes at the ends of a protein, and changes in the middle of the protein. I then realized that isolated proteins of length 4 or 6 aren't likely to happen in nature. I also realized that proteins are mostly middle. I found that the average yeast protein is 466 amino acids long(10). I decided that moving proteins down the chain by three would bring the stretch into the inside of the protein as it takes three amino acids to get the algorithm to not segmentation fault on NULL pointers and to get it to do the entire length of protein. So I weighted ends at 3/466 and middle as the rest or 230/233.

2b. Algorithm

I built the algorithm from scratch myself. I did have pointers from Dr. Jeffery in the CS department on how to make the program work better. He showed me the optimizer options with g++ and gprof, which shows you where you are spending your time and so where you may be able to manually optimize. Helpful hints included doing as many things as possible on single loops, avoid address calculations, and avoid doing loops where possible.

The protein is an abstract data type that is loosely based on the linked list. Each amino acid is a node in the linked list with numbers describing each one's Chao-Fasman parameters and in what kind of secondary structure it was in. DNA is strictly a linked list. The feature of indices was ignored in one function to improve efficiency. To maintain them would require looping to keep something that wasn't needed at the location.

The algorithm in the computer required three amino acid spaces to either side of the protein. I took advantage of that buffer and placed generalized “bookend” in those spaces. I changed the characteristics of the bookends to change the environment that the stretch of protein was in to reflect an end, alpha helix, beta sheet, or nondescript or otherwise unspecified secondary structures.

While I could have used DNA to cycle through all possible single point mutations, I used the protein instead and then took advantage of the Total Probability Theorem. I also used the Total Probability Theorem to weight the prevalence of any given codon and the probability of any particular mutation occurring. I weighted the codons in accordance to the frequency of occurrence in the human genome.

2c. Execution

The first time I ran the program, I ran it with a tetrapeptide as four was a common number in the algorithm. I had a function to increment through the environment that I was having problems preventing segmentation faults in so I manually incremented the environment. Each iteration took around 21 seconds with 6,165,180 comparisons. Then I realized that the largest number in the algorithm was six so I ran it again with hexapeptides. Each iteration in that state takes between 1.5 and 2.5 hours depending on machine and work load. Each iteration has 2,718,844,380 comparisons.

3. Result

The result that I came up with after testing with tetrapeptides for the probability that given a point mutation has occurred is 44.11%. I thought that since parts of the algorithm call of examining stretches of six amino acids, that I should repeat with that many spots. The second time, using hexapeptides resulted in a probability of 51.65% for damage occurring.

4. Discussion

I wanted to test the genetic code of a full protein in a program that would randomly mutate the code. The results of that didn't agree well with the results above, due in part to the assumption about “stop” codon changes always cause damage. Since the program works similarly to the assumptions of

a geometric distribution, I can compare the two results. For some reason that I have yet to figure out, the longer the DNA, and thus the longer the protein, the harder it was to break it and get statistically significant results. So, I violated the assumption concerning no isolated proteins of length 4 and 6 and looked strictly at the data from isolation. The average mutations to break a tetrapeptide was 6.8 mutations. The average of the geometric distribution is 1/p. In this case, the result was .1154 probability of damage to a tetrapeptide given a point mutation. That means that the average would be 8.7 mutations. The variance of the geometric distribution is 66.42. But the variance of for this simulation was in the 10's of thousands. The hexapeptide significantly further away.

I also wanted to do a real protein and compare it with a randomly generated protein. I had selected and download the the protein for hemoglobin. I came across a Wikipedia article that told me that what I had was not going to realistically work. In eukaryotic cells, like those of humans, are riddled with introns. I don't now what the introns are so I can't remove them which makes the code worthless for me. I was curious to find if a real protein is more stable than a random protein.

Bibliography

1. Beavis, R and Fenyo, D. *Amino Acid Properties.* “Chou-Fasman Parameters.” http://prowl.rockefeller.edu/aainfo/chou.htm
2. Chou, P and Fasman, G. Ann. Rev. Biochem. “Empirical Prediction of Protein Conformation.” 1978. 47:251-76
3. Ebersberger, I et al. *Am. J. Hum. Genet.* “Genomewide Comparison of DNA Sequences between Humans
and Chimpanzees
” 2002. 70:1490-97 doi:10.1086/340787.
4. Kazusa DNA Research Institute. *Codon Usage Table*. “Homo sapian.” http://www.kazusa.or.jp/codon/cgi-bin/showcodon.cgi?species=9606
5. Konko, K and Ohuchi, S. *Peptide Science.* “Reconstructing Chou-Fasman Parameters.” 2002. 2001:257-60.
6. *Wikipedia.* “DNA.” http://en.wikipedia.org/w/index.php?title=DNA&oldid=397057366
7. *Wikipedia.* “DNA codon table.” http://en.wikipedia.org/w/index.php?title=DNA_codon_table&oldid=392069780
8. *Wikipedia.* “Intron.” http://en.wikipedia.org/w/index.php?title=Intron&oldid=395684115
9. *Wikipedia.* “Point mutation.” http://en.wikipedia.org/w/index.php?title=Point_mutation&oldid=395495079
10. *Wikipedia.* “Protein.” http://en.wikipedia.org/w/index.php?title=Protein&oldid=395988081
11. *Wikipedia.* “Protein Secondary Structure.” http://en.wikipedia.org/w/index.php?title=Protein_secondary_structure&oldid=394885130