A New Front In Evolution: A Structural Landscape of Frameshift Mutations

Massachussetts Academy of Math and Science
Niam Shah

March 11, 2016

Table of Contents

Abstract

Currently, evolution is known to be caused by natural selection, gene flow, environmental factors, and mutations such as missense, nonsense, and duplications. It is unknown whether frameshift mutations contribute to evolution. These mutations are currently identified and studied by analyzing tissue samples collected from patients affected by pathogenic frameshift mutations. Therefore, many frameshift mutations, along with the evolutionary information they may contain, remain unidentified. An algorithm was developed to simulate all possible frameshift mutations that can occur in a human. This algorithm retrieved human mRNA sequences from the NCBI Nucleotide database, simulated frameshift mutations, and aligned the sequences to human proteins using BLAST. InterPro and SMART were used to create protein domain architectures for mutated, functional protein products. Three frameshift mutations were shown to conserve the gene's original protein. Five frameshift mutations were each shown to contribute to an evolutionary pathway. This work serves as an innovative approach to studying mutations at the protein domain level, and establishes a new frontier into evolutionary biology research.

Introduction

Bioinformatics has opened up new opportunities to analyze genetic material in a simulated environment at massive scale. One type of analysis is to simulate genetic mutations and model its effects. Currently, the primary method for identifying genetic diseases is to manually identify the mutations in samples collected from patients. Bioinformatics can be used to quickly and cheaply identify mutations.

Mutations can be studied in many different contexts, but most are studied in the context of disease because they are identified in affected individuals. However, bioinformatics can be used to study mutations from many different useful and relevant viewpoints, such as evolution. Moreover, because mutations can be completely and accurately modeled using bioinformatics, abnormalities and rare cases are often identified. These scenarios provide unique and interesting insights into the potential effects of a mutation.

Literature Review

Frameshift Mutations

A genetic disorder is a disease caused by an abnormality in an individual's DNA. These abnormalities can range from a single nucleotide mutation to a deletion or insertion of an entire chromosome (Genetic Disorders, 2014). Each year, approximately 5,860 infants do not survive due to genetic disorders and approximately 5.5% of the population has a genetic disorder (Collins, 2012). Of the affected population, 24% of cases are due to known frameshift mutations (Hu, 2012). Frameshift mutations are mutations caused by insertions or deletions of one or two nucleotides from a DNA sequence (Rosentiel, 2014). Because tRNA translates codons, groups of three mRNA nucleotides, to amino acids, frameshift mutations lead to a shift in the tRNA reading frame and thus a perturbed protein (Clancy & Brown, 2008; Rosentiel, 2014). These mutations generally occur in hot spots, repeated sequences of one or two nucleotides. This is due to a 'slip' of the DNA polymerase followed by the realignment of the DNA template and nascent strand during replication (Streisinger et al., 1966). However, frameshift mutations can also occur elsewhere in a DNA sequence. They lead to either an inactive protein or a protein with an altered structure and function (Rosentiel, 2014). Both of these cases are very dangerous and can result in many severe diseases such as Tay-Sachs disease, Crohn's disease, Cystic Fibrosis, and several types of cancer (D. Korkin, personal communication, December 4, 2015).

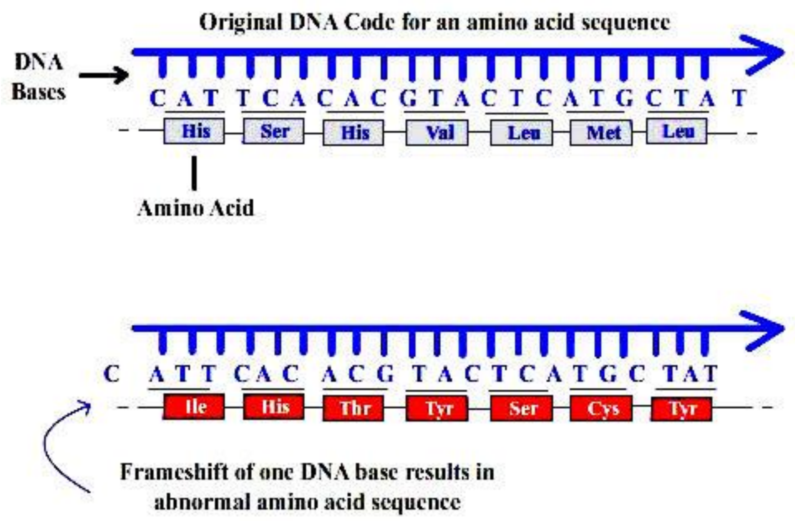

Original DNA Code for an amino acid sequence
DNA Bases
C A T T C A C A C G T A C T C A T G C T A T
His
Ser
His
Val
Leu
Met
Leu
Amino Acid
C A T T C A C A C G T A C T C A T G C T A T
Ile
His
Thr
Tyr
Ser
Cys
Tyr
Frameshift of one DNA base results in abnormal amino acid sequence

Figure 1. A shift in the reading frame by one base pair results in a dissimilar amino acid sequence and a perturbed protein (Environmental, 2012)

Current Identification of Frameshift Mutations

Researchers use many methods to identify frameshift mutations. For example, 2-aminopurine (AP), a fluorescent residue that can replace adenine in a DNA or RNA strand, can be used as a probe to reveal information about local conformations, structural and spatial arrangements of the nucleic acids (Johnson et al., 2004). Massively parallel sequencing is a process that simultaneously screens thousands of loci at once to locate pathogenic mutations (Tucker et al., 2009). However, these approaches are unreliable, costly, and less efficient than another method of mutation detection, called Polymerase Chain Reactions-Single Strand Conformation Polymorphism (PCR-SSCP) (Tucker et al., 2009; Hayashi, 1991).

This method uses polymerase chain reactions (PCR) to amplify gene segments by several orders of magnitudes. PCR-SSCP analyses differentiate between the nucleotide sequences of a large number of samples and are ideal for the diagnosis of cancer and other genetic diseases. First, PCR of genomic DNA or complementary DNA, cDNA, is performed using labeled substrates or reverse transcription PCR (RT-PCR). The PCR products are then denatured and run through a non-denaturing, poly-acrylamide gel electrophoresis. When the PCR product is denatured, it creates a unique folded structure because of intramolecular interactions between nucleotides. Thus, any difference in the sequence of a PCR product causes a unique folded structure. Gel electrophoresis separates samples based on their mobility due to size and shape allowing a single base pair mutation to be detected in a several hundred base pair fragment (Hayashi, 1991).

Although *Thermus aquaticus* (Taq) DNA polymerase can produce many additional mutations to a fragment, these mutations are negligible because there are approximately 10^4 molecules in each PCR reaction. PCR-SSCP only requires 5μl of DNA, which is much less than other techniques for detecting mutations, such as fluorescence of DNA fragments. However, since DNA-SSCP relies on each nucleotide of the DNA fragments, the temperature, concentration of ions, and solvents must be closely regulated to ensure that no other mutations occur (Hayashi, 1991).

All current approaches of identifying frameshift mutations require the condition of a patient to be known and a DNA sample of the patient (Tucker et al., 2009; Hayashi, 1991).

Frameshift Mutations Identified On Case-To-Case Basis

Several recent studies that identify frameshift mutations describe the process of diagnosing a disease and determining the location of the mutation in the genome.

One study describes the discovery of a frameshift mutation causing child onset schizophrenia (COS), a rare, yet severe case of schizophrenia. The UPF3B gene has been proven to cause syndromic and nonsyndromic X-linked mental retardation or autism and has been correlated with many neurological disorders such as autism, intellectual disability, and schizophrenia. Because a mother gave birth to two sons, one with autism and COS and the other with autism and ADHD, the UPF3B gene was hypothesized to have a mutation. To verify this, the UPF3B gene was analyzed using RT-PCR and PCR-SSCP. The sibling with autism and COS was found to have a four base pair deletion in exon seven of the UPF3B gene. The other sibling had the same mutation as well as an additional mutation. The four base pair frameshift mutation produces a premature stop codon resulting in a nonfunctional protein which causes autism and COS (Addington et al., 2011).

In another case related to obesity, two frameshift mutations in the MC4-R receptor gene were identified. The MC4-R receptor binds the α-MSH ligand in the hypothalamus. The α-MSH molecule is thought to transmit satiety signals, so inactivation of the MC4-R receptors is correlated with a dominant form of obesity. A PCR-SSCP analysis discovered a four base pair insertion and a four base pair deletion in the MC4-R coding region solely in obese patients. These mutations culminated in three premature stop codons in three different protein domains. This resulted in a nonfunctional MC4-R receptor leading to obesity (Hinney et al., 1999).

Researchers have also discovered a frameshift mutation causing Osteogenesis Imperfecta (OI). OI is a disease associated with osteoporosis and an increased risk of bone fractures. Most cases of OI result from non-frameshift mutations in procollagen genes, COL1A1 and COL1A2, which reduce the amount of procollagen 1, which is a precursor to structural proteins in the extracellular matrix. Homozygosity mapping was performed in the proband and his unaffected sister and revealed that the SP7/OSX gene was the most likely to contain the mutation. After analyzing the SP7/OSX coding region through a process very similar to PCR-SSCP, a single base pair deletion was detected. The OSX protein encoded by the mutated SP7/OSX gene avoids the nonsense-mediated mRNA decay machinery and is functional. The mutated protein's harmful function results in OI (Lapunzina et al., 2010).

Table 1. Documented disease-causing frameshift mutations

Condition/Disease	Mutated Gene	Insertion or Deletion (base pairs)	Functional or Nonfunctional Protein	Harmful or Beneficial
Obesity	MC4-R	Both (4) (4)	Nonfunctional	Harmful
Child Onset Schizophrenia	UPF3B	Deletion (4)	Nonfunctional	Harmful
Spinal Muscular Atrophies	Survival Motor Neuron	Deletion (5)	Nonfunctional	Harmful
Chron's Disease	NOD2	Insertion (1)	Nonfunctional	Harmful
Osteogenesis Imperfecta	SP7/OSX	Deletion (1)	Functional	Harmful
Non-Small Cell Lung Cancer	p53	Both (11) (4)	Nonfunctional	Harmful
Cystic Fibrosis	CFTR	Both (2) (1)	Nonfunctional	Harmful
Hypertrophic Cardiomyopathy	TNNC1	Insertion (1)	Unknown	Harmful
Gastric and Colorectal Cancer	SGOL1 and PDS5B	Insertion (1)	Nonfunctional	Harmful
Breast Cancer	CCR5	Deletion (32)	Functional	Harmful
Tay-Sachs Disease	HEXA	Both (1) (2)	Nonfunctional	Harmful

The Protein Domain-Centric Approach

Today, whole-genome DNA sequencing has enabled the evaluation of tissue samples from hundreds of patients to be screened for driver mutations, mutations that are involved with causing a certain disease. Most approaches to identifying driver mutations involve a gene-centric approach, identifying driver mutations that occur in a high-percentage of the patients' gene-ofinterest. However, this approach is limited to a small subset of genes and does not take into account mutations occurring very rarely (Nehrt et al., 2012).

Additionally, gene-centric approaches do not distinguish between mutations located on different sites of a gene, so information about the functional context of the mutation is disregarded. This approach can be unreliable because it has the potential to mischaracterize a mutation. (Zhong et al., 2009). Examples have been shown where mutations in the same protein, but in different protein domains, conserved and distinct functional and structural components of a protein, can lead to various disease phenotypes. Mutations can more accurately be studied on a protein-domain level because individual protein-domains add functional information to reveal the impact of the mutations (Nehrt et al., 2012).

In this study, tumor samples from 100 patients with colon adenocarcinoma were analyzed. Somatic frameshift mutations, stop-gain and stop-loss mutations, and single nucleotide variants that caused changes in the amino acid sequences were identified among the patients. Gene and domain mutational landscapes were created, where each gene represented an area on the xy-plane and the frequency of the gene's mutation is represented on the z-plane (Nehrt et al., 2012).

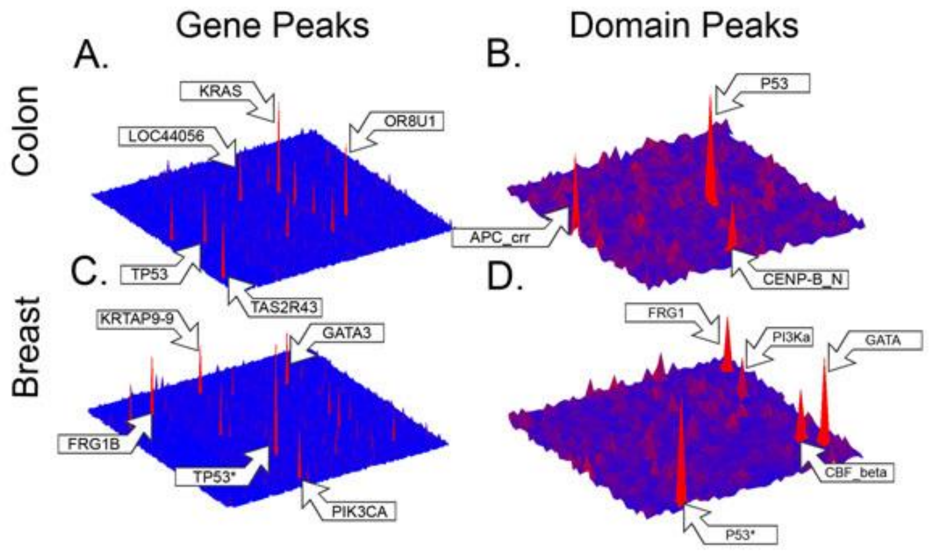


Figure 2. An example of a domain landscape mapping tumor samples from patients with colon adenocarcinoma or breast adenocarcinoma (Nehrt et al., 2012)

In the colon cancer study, some of the gene peaks, such as the TP53 and APC gene peaks resulted in the P53 and the APC_crr domain peaks, respectively. However, the CENP-B_N domain peak was caused by aggregations of 'gene hills,' such as the TIGD7 and JRKL gene hills. The domain peaks can therefore be a result of gene peaks or aggregations of gene hills, and these conclusions would not have been made using a gene-centric approach. Domain landscapes can also be compared to find domains that are responsible for multiple genes. For example, the KRAS, CELA1, SERTAD3 domains cause both colon and breast cancer (Nehrt et al., 2012).

Domain landscaping also reveals information about the modularity of a protein, the domains of a protein that change the function of it (Nehrt et al., 2012).

Homology Modeling

Homology modeling is a modeling process to determine structural information about an un-modeled protein based on the three-dimensional structure of a homologous protein, a protein with a high residue similarity and similar ancestry. This process is based around the idea that the structure of a protein is evolutionarily conserved better than its amino acid sequence. It is currently the most reliable and accurate method of obtaining structural information about a protein (Zvelebil & Baum, 2007).

Models can either be built around the sequences of a single homolog or the averages of multiple related homologs. If the sequence similarity between the target and template sequences is approximately 90%, then a single homolog will give the most accurate structure. However, if the sequence similarity between the target and template sequences is lower, approximately 45%, then the averages of homologous sequences will give the most accurate structure of the protein. Template sequences can be found using online programs such as BLAST or FASTA, which find the homologs in the Protein Data Bank. If no homologs are found, short sequences of homology can be used to construct an incomplete model. Alternatively, a threading method to find a possible template can be tried (Zvelebil & Baum, 2007).

The next step is accurate sequence alignment. This is the most important stage because even if one residue is misaligned, the target protein structure can be completely incorrect. To increase the accuracy of the model, charged residues are generally located on the surface of the protein and multiple homologous sequences should be used to model the core, even if only one homolog is used to model the protein. Online programs such as ClustalW can automate sequence alignment (Zvelebil & Baum, 2007).

Because the core is the most structurally conserved region of a protein, it is modeled first. This is done by aligning the x-y-z coordinates of each atom of the template's core and matching it to the aligned residues of the target molecule. The atoms are then joined together by peptide bonds. Regions with insertions or deletions should not be included in the core, but if they must, the residues should be realigned to produce a more favorable conformation. If multiple core structures have been made, the most favorable one should be chosen. Because insertions and deletions have been omitted, the core is a set of discontinuous chains. These chains are connected by loops, which contain insertions and deletions. Loops are involved in ligand recognition, ligand binding, and sometimes are components of the active site of proteins. Loop sequences are short fragments containing the insertion or deletion. They can be found on databases with high-resolution structures. Loops are annealed to the discontinuous chains and are modeled from anchor points, two or three residues on the ends of the chains (Zvelebil & Baum, 2007).

Side chains confer distinct structural and functional properties of a protein. If the side chain residues are aligned and conserved, the side chains of the template and target proteins have the same conformation. (Zvelebil & Baum, 2007).

Errors between the core and loops are resolved using energy minimization. Programs such as CHARMM alleviate bad bond angles and improve the interactions between atoms by altering local geometries. To assess the quality of the final structure, programs such as PROCHECK and MolProbity compare torsion angles, bond angles, bond lengths, and distributions of polar and hydrophobic residues against a standard (Zvelebil & Baum, 2007).

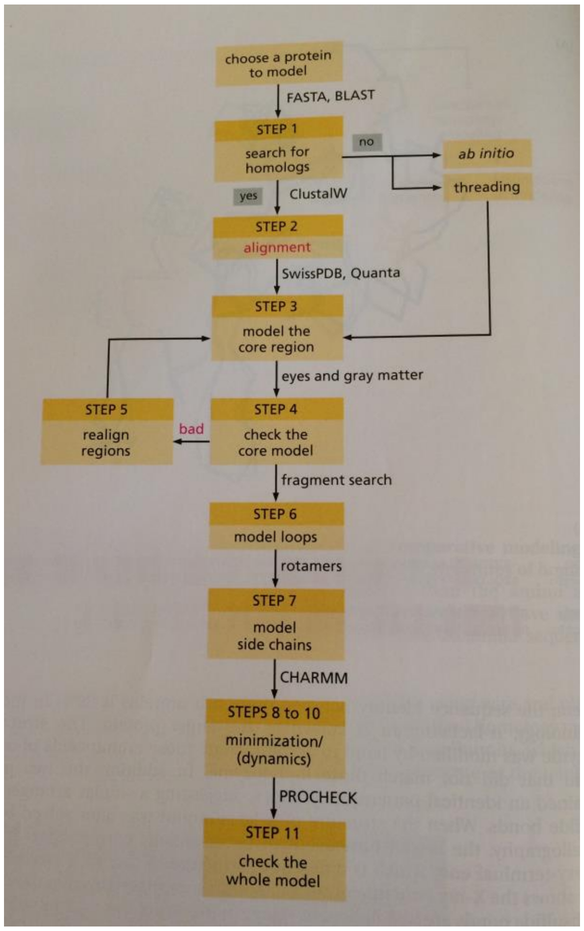


Figure 3. A flowchart showing the process of homology modeling (Zvelebil & Baum, 2007)

Although homology modeling can be done semi-automatically by using online tools at each step, fully automated programs such as COMPOSER and MODELLER exist. COMPOSER aligns the target sequence to the template sequence three residues at a time and identifies structurally conserved regions. Based on these regions, COMPOSER predicts the structure of the core and the loops and follows the process described above. MODELLER predicts protein structure by extracting spatial constraints and atom-atom distances from the template protein, combining this information with properties such as bond length and angle preferences, and applying the combined properties to the target protein (Zvelebil & Baum, 2007).

Sequence Similarity Search

A sequence similarity search is the process of taking a DNA or protein sequence and matching it to known similar sequences in a database, so that the function of the input DNA or protein sequence can be inferred from the function of the known protein. This is a complex process because the goal is to match protein domains. If one attempted to match proteins over their entire lengths, very few matches would be found. The most popular way to do this search is via a method called 'Basic Local Alignment Search Tool,' or BLAST (Madden, 2013). As its name implies, this tool works by trying to match domains over shorter lengths of the protein sequences. The BLAST algorithm can be broken down into three steps: setup, preliminary search, and traceback. In the setup phase, BLAST breaks the input sequence into short, fixed-length, sequences called "words" based on the search parameter, database, and scans for low-complexity or other repeated sequences in the input sequence (Madden, 2013). Next, in the search phase, each word is searched in the database, and every matching sequence is added to a gap-free extension. Then, the gap-free extensions that meet a certain score are used to seed a gapped extension. The score for gapped extensions is calculated, without taking into consideration any insertions or deletions. Finally, in the third phase called traceback, insertions and deletions for gapped extensions that achieve a certain score are calculated. The final output is a matched protein with a matching score that indicates how closely the proteins matched. BLAST also calculates an expect value, or e-value, that indicates how many matches at a given score would have happened by chance. The e-value effectively indicates the confidence level of the alignment match. Low e-values means very high confidence (Madden, 2013).

Protein Classification

In order to understand the function of a protein, it is necessary to identify its domains and functional sites. Tools such as InterPro match proteins to their component domains and functional sites. InterPro has a database of protein families, domains, and functional sites of known proteins (Alex et al., 2015). Given a new protein sequence, InterPro will match the features of the known proteins to the new protein, and then return the corresponding domain or functional site. The database consists of diagnostic signatures and the proteins that they significantly match. Some signatures are simple, matching sequences of amino acids, and other signatures are more complex, and consist of statistical models such as Hidden Markov models. InterPro is a collection of eleven such databases, from very high-level, structure-based, classifications to very specific sub-family classifications. Thus, InterPro is considered a one-stop shop for protein classification (Alex et al., 2015).

Protein Domain Architecture

Once the domains of a protein are known, the domain architecture must be created. Tools such as Simple Modular Architecture Research Tool, or SMART, perform this. SMART identifies and annotates genetically mobile domains across more than 1200 domains associated with signaling, extracellular, and chromatic associated proteins (Schultz et al., 2000). Domains are annotated with respect to phyletic distributions, functional class, tertiary structures, and functionally important residues. The annotations come from the SMART database, where they have been manually curated and annotated. SMART produces high-resolution schematics of protein architectures that can be saved to file and included in other documents. SMART also produces a rich set of annotations that allow the researcher to easily navigate to many other sources of information about the domain or functional site (Schultz et al., 2000).

Python

Python is a high-level, object-oriented programming language that supports modules and packages (“Welcome To Python.org,” 1991). One such package is BioPython, which is a set of freely available tools for biological computation (BioPython, 2013). Python also has a large standard library that covers areas of string processing, making string manipulation very straightforward and simple (“Welcome To Python.org,” 1991). This is very useful because DNA, mRNA, and amino acids are all strings that will be processed.

Engineering Plan

Engineering Problem

Currently, frameshift mutations are identified on a case-by-case basis in patients with diseases caused by these mutations. Thus, many frameshift mutations remain unidentified, and have rarely been studied from an evolutionary perspective.

Engineering Goal

The goal of this project was to engineer a computer simulation to generate a structural landscape of all possible frameshift mutations that can occur in Homo sapiens to reveal evolutionary information about mutated, functional protein products.

Methodology

The computer program will be written in Python in the PyDev Integrated Development Environment (IDE). The first objective of the program will be to simulate the frameshift mutation in each gene of a *Homo sapien*. The gene sequences will be taken from the NCBI Nucleotide database. To do this, the reading frame will be shifted one or two nucleotides forwards. Because frameshift mutations are caused by a shift in the reading frame due to an insertion or deletion of nucleotides not divisible by three, a shift in the reading frame without the addition of nucleotides will also simulate the same frameshift mutation. Next, translation will be simulated. The resulting amino acid sequence will then be aligned to a protein using BLAST, and undergo further analysis using InterPro. SMART will be used to create protein domain architectures of functional, mutated protein products.

If successful, this research will correctly identify many frameshift mutations, and do so in a significantly quicker and more efficient manner. Additionally, information about the evolutionary consequences of frameshift mutations and novel proteins may be revealed. This project will be completed at home and at the Massachusetts Academy of Math and Science at WPI.

Methodology

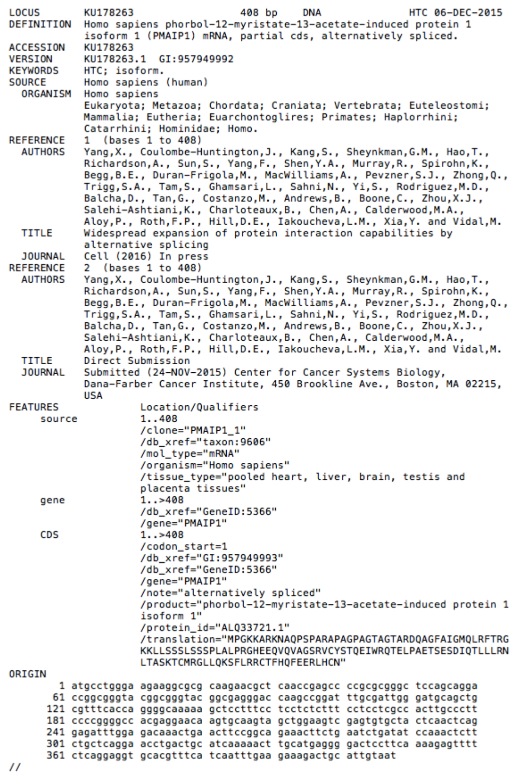
The purpose of this program was to simulate all possible frameshift mutations that can occur in a human. In order to complete the simulation, the NCBI Nucleotide database, BLAST, and InterPro had to be accessed. The simulation was written in Python mainly because of the ability to use BioPython, a set of tools useful for accessing databases and other online tools. The NCBI Nucleotide database was accessed, and all mRNA records with coding DNA sequences were retrieved. One-frame and two-frame frameshift mutations were simulated on each CDS, the mRNA sequences were translated, and the mutated and original amino acid sequences were sent to BLAST. Sequences that had results from BLAST were analyzed using InterPro. Finally, sequences that had results from InterPro were manually analyzed using SMART. A detailed description of each method and program can be seen below.

The accessNCBI Method

Table 2. Objects used in the accessNCBI method

accessNCBI Objects		
Object	**Package**	**Object Description**
Entrez	Bio	Specified a search query for NCBI databases and retrieved a record’s Unique Identification Number (UID)
SeqIO	Bio	Read NCBI database records and write them to files
os	*	Allowed use of operating system dependent functionality

The accessNCBI method was used to retrieve all human mRNA records from the NCBI Nucleotide database. The NCBI Nucleotide database was searched for the mRNA records using the Entrez object. Each mRNA record’s UID was returned and the Entrez object accessed the records using the returned UIDs. Through an iteration structure, the SeqIO and os objects read each GenBank formatted record and wrote it to a file, named after the record’s accession number, located on a local directory named ‘mRNARecords.’ The SeqIO object was chosen because it can later be used to read and obtain information from files storing mRNA records.



```
LOCUS       KU178263                 408 bp    DNA              HTC 06-DEC-2015
DEFINITION  Homo sapiens phorbol-12-myristate-13-acetate-induced protein 1
            isoform 1 (PMAIP1) mRNA, partial cds, alternatively spliced.
ACCESSION   KU178263
VERSION     KU178263.1  GI:957949992
KEYWORDS    HTC; isoform.
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 408)
  AUTHORS   Yang,X., Coulombe-Huntington,J., Kang,S., Sheynkman,G.M., Hao,T.,
            Richardson,A., Sun,S., Yang,F., Shen,Y.A., Murray,R., Spirohn,K.,
            Begg,B.E., Duran-Frigola,M., MacWilliams,A., Pevzner,S.J., Zhong,Q.,
            Trigg,S.A., Tam,S., Ghamsari,L., Sahni,N., Yi,S., Rodriguez,M.D.,
            Balcha,D., Tan,G., Costanzo,M., Andrews,B., Boone,C., Zhou,X.J.,
            Salehi-Ashtiani,K., Charloteaux,B., Chen,A., Calderwood,M.A.,
            Aloy,P., Roth,F.P., Hill,D.E., Iakoucheva,L.M., Xia,Y. and Vidal,M.
  TITLE     Widespread expansion of protein interaction capabilities by
            alternative splicing
  JOURNAL   Cell (2016) In press
REFERENCE   2  (bases 1 to 408)
  AUTHORS   Yang,X., Coulombe-Huntington,J., Kang,S., Sheynkman,G.M., Hao,T.,
            Richardson,A., Sun,S., Yang,F., Shen,Y.A., Murray,R., Spirohn,K.,
            Begg,B.E., Duran-Frigola,M., MacWilliams,A., Pevzner,S.J., Zhong,Q.,
            Trigg,S.A., Tam,S., Ghamsari,L., Sahni,N., Yi,S., Rodriguez,M.D.,
            Balcha,D., Tan,G., Costanzo,M., Andrews,B., Boone,C., Zhou,X.J.,
            Salehi-Ashtiani,K., Charloteaux,B., Chen,A., Calderwood,M.A.,
            Aloy,P., Roth,F.P., Hill,D.E., Iakoucheva,L.M., Xia,Y. and Vidal,M.
  TITLE     Direct Submission
  JOURNAL   Submitted (24-NOV-2015) Center for Cancer Systems Biology,
            Dana-Farber Cancer Institute, 450 Brookline Ave., Boston, MA 02215,
            USA
FEATURES             Location/Qualifiers
     source          1..408
                     /clone="PMAIP1_1"
                     /db_xref="taxon:9606"
                     /mol_type="mRNA"
                     /organism="Homo sapiens"
                     /tissue_type="pooled heart, liver, brain, testis and
                     placenta tissues"
     gene            1..>408
                     /db_xref="GeneID:5366"
                     /gene="PMAIP1"
     CDS             1..>408
                     /codon_start=1
                     /db_xref="GI:957949993"
                     /db_xref="GeneID:5366"
                     /gene="PMAIP1"
                     /note="alternatively spliced"
                     /product="phorbol-12-myristate-13-acetate-induced protein 1
                     isoform 1"
                     /protein_id="ALQ33721.1"
                     /translation="MPGKKARKNAQPSPARAPAGPAGTAGTARDQAGFAIGMQLRFTRG
                     KKLLSSSLSSSPLALPRGHEEQVQVAGSRVCYSTQEIWRQTELPAETSESDIQTLLLRN
                     LTASKTCMRGLLQKSFLRRCTFHQFEERLHCN"
ORIGIN
        1 atgcctggga agaaggcgcg caagaacgct caaccgagcc ccgcgcgggc tccagcagga
       61 ccggcgggta cggcgggtac ggcgagggac caagccggat ttgcgattgg gatgcagctg
      121 cgtttcacca ggggcaaaaa gctcctttcc tcctctcttt cctcctcgcc acttgccctt
      181 ccccggggcc acgaggaaca agtgcaagta gctggaagtc gagtgtgcta ctcaactcag
      241 gagatttgga gacaaactga acttccggca gaaacttctg aatctgatat ccaaactctt
      301 ctgctcagga acctgactgc atcaaaaact tgcatgaggg gactccttca aaagagtttt
      361 ctcaggaggt gcacgtttca tcaatttgaa gaaagactgc attgtaat
//
```

Figure 4. An example of a record, from the NCBI Nucleotide Database, written to a file

The callBLAST Method

Table 3. Objects used in the callBLAST method

callBLASTObjects		
Object	**Package**	**Object Description**
NCBIWWW	Bio.Blast	Specified a BLAST query
NCBIXML	Bio.Blast	Parsed BLAST ouptut
SeqIO	Bio	Read NCBI database records and write them to files
generic_rna	Bio.Alphabet	Contained functions that pertained to mRNA sequences
Seq	Bio.Seq	Managed information about a sequence
os	*	Allowed use of operating system dependent functionality

The callBLAST method was used to access BLAST and send queries of original and frameshift mutated sequenes to it. The os object retrieved each file that was written by the accessNCBI method. The SeqIO object read each file's record, and the Seq and generic_rna objects extracted the CDS (Coding DNA Sequence) from the record and saved it as a string. Subtracting one and two nucleotides to the beginning of the mRNA CDS substring simulated one-frame and two-frame frameshift mutations, respectively. The NCBIWWW object accessed the Protein BLAST (Basic Local Alignment Search Tool) algorithm and both mutated sequences were inputted into BLAST against the Non-redundant Protein Sequences Database. Only human proteins were returned and the Expect Value was set to 0.0001 in order to increase the speed of the BLAST processing time and ensure very accurate results (D. Korkin, personal communication, January 18, 2016). The BLAST results were returned in XML (EXtensible Markup Language)-format and written to a file, named after the record's accession number and number of frames shifted. These files were located on a local directory named 'BLASTResults.' Mutations that led to an immediate stop codon resulted in empty amino acid sequences. The files for these sequences were created, but contained no content. If a BLAST Query disconnected from the BLAST servers, 'Failed' was written to the file. This program was run again for the failed query sequences.

Running BLAST Using Distributed Computing

Table 4. Objects used in the CreateBLASTDirs and SetupPC programs

CreateBLASTDirs and SetupPC Objects		
Object	**Package**	**Object Description**
os	*	Allowed use of operating system dependent functionality
shutil	*	Provided functions for moving and transferring files

The CreateBLASTDirs and SetupPC programs were written to transfer necessary files to multiple computers and run the callBLAST method on each computer. In order to increase the speed of BLASTing all mutated sequences, nine computers, which ran ten terminal windows each, were used. Based on the total number of terminal windows (in this study, there were 90 terminal windows), the CreateBLASTDirs program had the os object create one directory for each terminal window. The shutil module was used to divide the total number of mRNA records to each terminal window's directory. These directories were named from 'mRNARecords1' ranging to 'mRNARecords90.'

The SetupPC program used the os object to create ten directories on each computer. These directories were named from ‘FS1’ ranging to ‘FS10.’ The shutil module copied the BLAST and Simulation programs, as well as ten mRNA record directories to a directory named ‘FS.’ Then, the BLAST and Simulation programs and one mRNA record directory were copied to each of the ten ‘FS1’-‘FS10’ directories. A directory named ‘BLASTResults’ was also created in each of these ten directories using the os object.

A command line script named DoAll was used to open ten terminal windows and run the Simulation program. Before a new terminal window was opened, there was a ten second interval to prevent ten BLAST queries from being sent instantaneously to the BLAST servers.

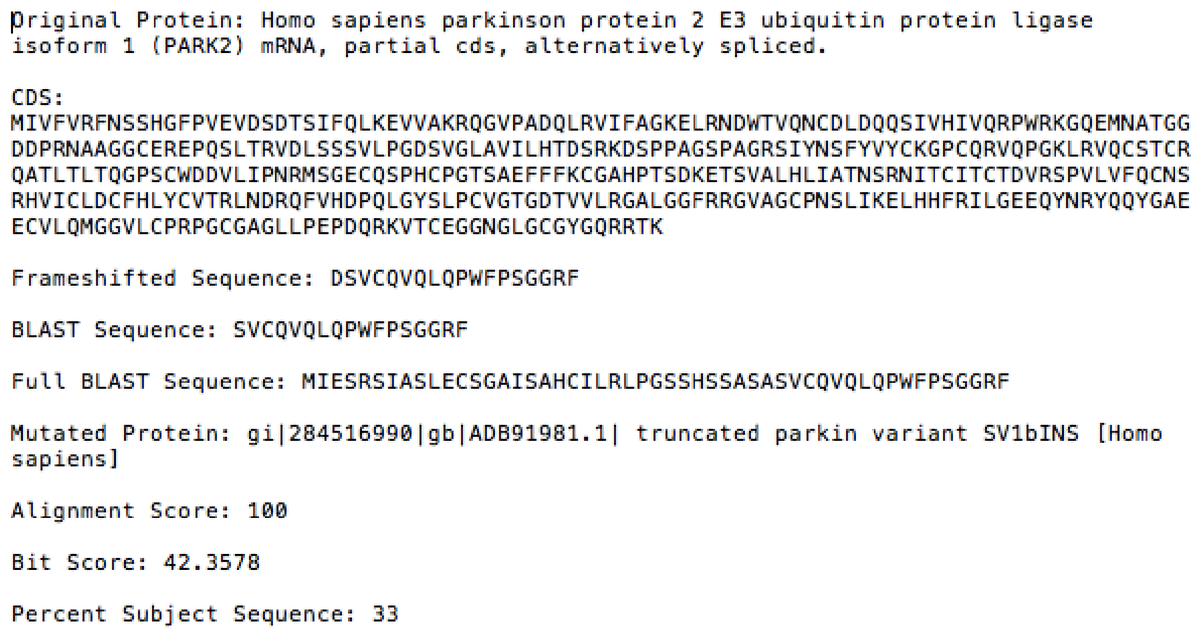
The readBLAST Program

Table 5. The objects used in the readBLAST program

readBLAST Objects		
Object	**Package**	**Object Description**
Entrez	Bio	Specify a search query for NCBI databases
NCBIXML	Bio.Blast	Parsed BLAST ouptut
SeqIO	Bio	Read NCBI database records and write them to files
generic_rna	Bio.Alphabet	Contained functions that pertained to mRNA sequences
Seq	Bio.Seq	Managed information about a sequence
os	*	Allowed use of operating system dependent functionality
shutil	*	Provided functions for moving and transferring files

The readBLAST program was used to parse the BLAST output and write important information to separate files. The NCBIXML object was used to read each file from the BLASTResults directory. Files that contained BLAST results were copied to the ‘FunctionalProteins’ directory using the shutil object. Each file’s corresponding mRNA record using the SeqIO object and the CDS and record title were accessed using the Seq object. The NCBIXML object read each file from the ‘FunctionalProteins’ directory and stored the best BLAST result, based on the bit score. The bit score is a logarithmic-based score provided by

BLAST that factors in the alignment similarity score and the percent subject sequence between the queried sequence and the resulting BLAST sequence. For the result with the highest bit score, the alignment score, the bit score, the percent subject sequence score, and the resulting BLAST sequence were stored in variables. The Entrez and SeqIO objects accessed the BLAST sequence's protein record on the NCBI Protein database and the entire BLAST sequence was stored. The os object wrote the original mRNA record title, the translated CDS, the mutated amino acid sequence, the returned BLAST amino acid sequence, the entire BLAST amino acid sequence, the returned BLAST protein name, the alignment score, the bit score, and the percent subject sequence score to a file located in a local directory named 'ProteinResults.'



```
Original Protein: Homo sapiens parkinson protein 2 E3 ubiquitin protein ligase
isoform 1 (PARK2) mRNA, partial cds, alternatively spliced.

CDS:
MIVFVRFNSSHGFPVEVDSDTSIFQLKEVVAKRQGVPADQLRVIFAGKELRNDWTVQNCDLDQQSIVHIVQRPWRKGQEMNATGG
DDPRNAAGGCEREPQSLTRVDLSSSVLPGDSVGLAVILHTDSRKDSPPAGSPAGRSIYNSFYVYCKGPCQRVQPGKLRVQCSTCR
QATLTLTQGPSCWDDVLIPNRMSGECQSPHCPGTSAEFFFKCGAHPTSDKETSVALHLIATNSRNITCITCTDVRSPVLVFQCNS
RHVICLDCFHLYCVTRLNDRQFVHDPQLGYSLPCVGTGDTVVLRGALGGFRRGVAGCPNSLIKELHHFRILGEEQYNRYQQYGAE
ECVLQMGGVLCPRPGCGAGLLPEPDQRKVTCEGGNGLGCGYGQRRTK

Frameshifted Sequence: DSVCQVQLQPWFPSGGRF

BLAST Sequence: SVCQVQLQPWFPSGGRF

Full BLAST Sequence: MIESRSIASLECSGAISAHCILRLPGSSHSSASASVCQVQLQPWFPSGGRF

Mutated Protein: gi|284516990|gb|ADB91981.1| truncated parkin variant SV1bINS [Homo
sapiens]

Alignment Score: 100

Bit Score: 42.3578

Percent Subject Sequence: 33
```

Figure 5. An example of a file containing the BLAST results as described above

The InterPro Program

Table 6. The objects used in the InterPro Program

InterPro Objects		
Object	**Package**	**Object Description**
os	*	Allowed use of operating system dependent functionality
time	*	Provided time-related functions

The InterPro program accessed and sent sequence queries of the original CDS and the frameshift mutated amino acid sequence to InterPro. A text file of each CDS and frameshift mutated sequence was added to the 'InterProInput' directory. The os object accessed each file from this directory, and added it to the sequence key of a params dictionary. The InterPro REST API was downloaded and the serviceRun and getResult methods were used (Binns et al., 2014). The serviceRun method accessed InterPro and submitted a sequence query of the sequence stored in the params dictionary. It returned a jobid that was used to call the getResult method. The getResult method wrote the serviceRun results to an XML file stored in the 'InterProOutput' directory. The files were named after the mRNA accession number and whether it was a CDS sequence or a frameshifted sequence. If a query failed, the time object was used to force the program to sleep for three minutes, so that if the InterPro servers were down, all subsequent queries would not also fail.

The InterProAnalysis Program

Table 7. The objects used in the InterProAnalysis Program

InterProAnalysis Objects		
Object	**Package**	**Object Description**
os	*	Allowed use of operating system dependent functionality
etree	lxml	Implemented the extended ElementTree API for XML

The InterProAnalysis program parsed the XML files from the InterPro program and identified domains returned for frameshifted amino acid sequences. lxml, a python library containing tools to parse XML was downloaded. A getDomains method was created to find all protein domains in the XML file. The etree object accessed each XML file and the root's children that had a type of 'DOMAIN' were returned.

The os object iterated through each CDS XML file and found its protein domains using the getDomains method. The frameshifted XML file was also accessed and its protein domains were found using the getDomains method. If the getDomains method returned a result for both the CDS and frameshifted XML files, each files' protein domains were printed out.

SMART

The Simple Modular Architecture Research Tool (SMART) was used to create protein domain architectures from the mutated, functional protein products returned by InterPro. SMART was accessed manually because there were very few sequences that needed to be analyzed.

Prototype Process Flow

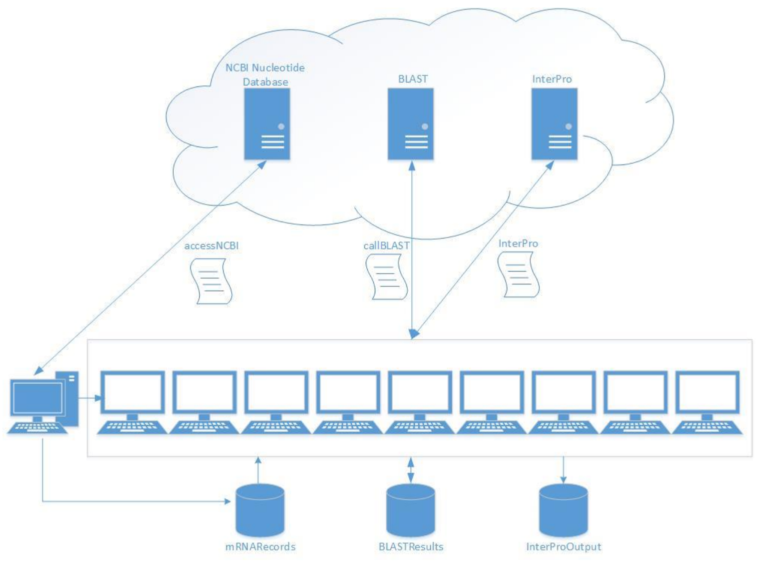


Figure 6. The diagram above shows the prototype process flow

Results

Testing

In order to test the prototype, the effects of ten frameshift mutations from the Database of Single Nucleotide Polymorphisms (dbSNP) were compared to the effects of the simulation's frameshift mutations in the same gene. dbSNP contains mutations that have been discovered by scientists in subjects' genomes, so dbSNP can be used to determine the accuracy of the simulation. Records from dbSNP were chosen if the mutation occurred toward the beginning of the mRNA sequence because mutations in the simulation were simulated at the beginning of the mRNA sequence.

Table 6. Testing using the dbSNP database. The reference SNP, mRNA accession number, number of frames shifted, and resulting protein products are shown below.

Testing Using The dbSNP Database				
Reference SNP (rs)	**mRNA Accession Number**	**Number of Frames Shifted**	**dbSNP Mutation Protein Product**	**Simulation Protein Product**
rs3050458	NM_001242480.1	1	Nonfunctional	Nonfunctional
rs74212399	NM_038958.1	1	Nonfunctional	Nonfunctional
rs36144466	NM_038958.1	1	Nonfunctional	Nonfunctional
rs387906218	NM_006642.3	2	Nonfunctional	Nonfunctional
rs397515335	NM_006671.3	2	Nonfunctional	Nonfunctional
rs587777847	NM_006642.3	1	Nonfunctional	Nonfunctional
rs587777846	NM_001005741.2	2	Nonfunctional	Nonfunctional
rs397515336	NM_006642.3	1	Nonfunctional	Nonfunctional
rs397518434	NM_001005741.2	2	Nonfunctional	Nonfunctional
rs398123529	NM_000157.3	2	Nonfunctional	Nonfunctional

Because all ten dbSNP frameshift mutations matched the corresponding ten simulation frameshift mutations, the simulation was considered accurate and successful. Further analysis of the simulated, mutated amino acid sequences was able to proceed.

Criteria Matrix

The program's functions and abilities were measured using a criteria matrix. In Table 7., there are six different criteria, and the program was given a score ranging from zero to three for each criterion. A score of zero signifies that the function is not present in the program. A score of one signifies that the function is present in the program. A score of two signifies that the program is able to recover from an error that may occur while running. A score of three signifies that the program is able to restart, and maintains its current state.

Table 7. Criteria matrix. The program was given a score of zero ranging to three for each criterion.

Criteria Matrix	
Criteria	**Prototype Score**
Mutation Simulation	3
Variability of Mutation Position	0
Access NCBI Nucleotide Database	1
Access BLAST	3
Access InterPro	3
Run Programs Using Distributed Computing	3
Total	**13**
Maximum	**18**

In the remainder of this section, data is categorized and analyzed. The simulation simulated frameshift mutations on 24,797 genes. Each gene produced two mutated sequences, so a total of 49,596 amino acid sequences were analyzed using BLAST. These results yielded nonfunctional proteins, no proteins, and potentially functional proteins. Potentially functional proteins were analyzed using InterPro, and were then concluded to be either functional or indeterminable. The protein domain architectures for functional proteins were created using SMART, of which eight are shown below.

Table 8. The number of BLAST results is classified by the bit score, a logarithmic scaled version of the raw alignment score.

Score of BLAST Results	
Bit Score (S')	**Number of BLAST Results**
0	0
20	118
40	1979
60	587
80	248
100	162
120	78
140	113
160	101
180	66
200	85
220	35
240	27
260	14
280	25
300	27
320	10
340	7
360	5
380	3
400	5
420	5
440	2
460	4
480	2
500	1
520	1
540	2
560	0
580	2
600	6
620	2
640	1
660	1
680	0
700	1
720	0
740	0
760	2
780	1

In Figure 7., the bit score frequency is plotted. The values and frequency of the bit scores can be seen in Table 8. (above).

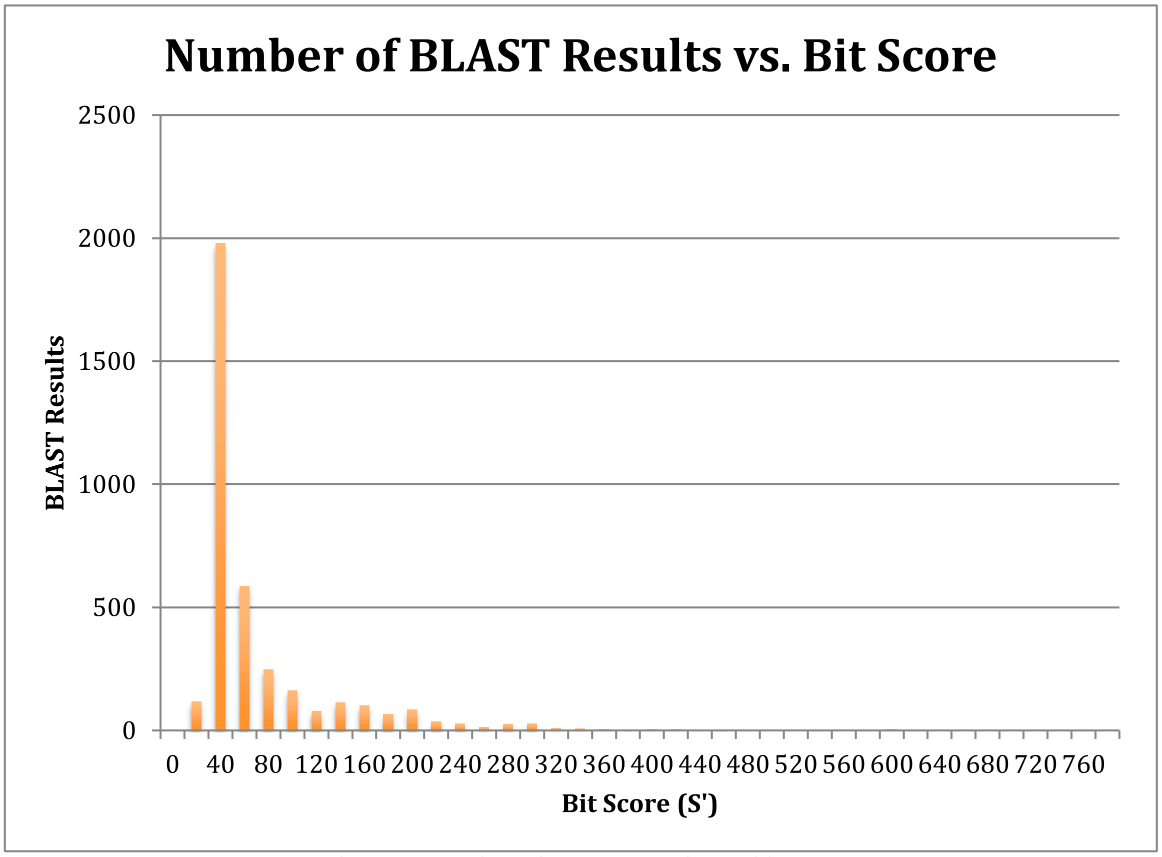


Figure 7. Number of BLAST results vs. bit score

After the BLAST analysis as completed, each frameshift mutated protein product was classified as nonfunctional, no protein, or potentially functional. Potentially functional proteins were categorized as nonfunctional or functional after the InterPro analysis.

Table 9. Breakdown of frameshift mutation protein products. Each protein was determined to be nonfunctional, no protein, or functional after the BLAST and InterPro analysis.

	Frameshift Mutated Proteins	
	Number of Proteins	**Percent of Total Proteins (%)**
Nonfunctional (No BLAST Results)	40765	82.5
No Protein	5097	10.7
Indeterminable (After InterPro Analysis)	3711	6.8
Functional (After InterPro Analysis)	23	0.04
Total	49596	100

Figure 8. shows the classified frameshift mutated protein products. The data below was obtained from Table 9.

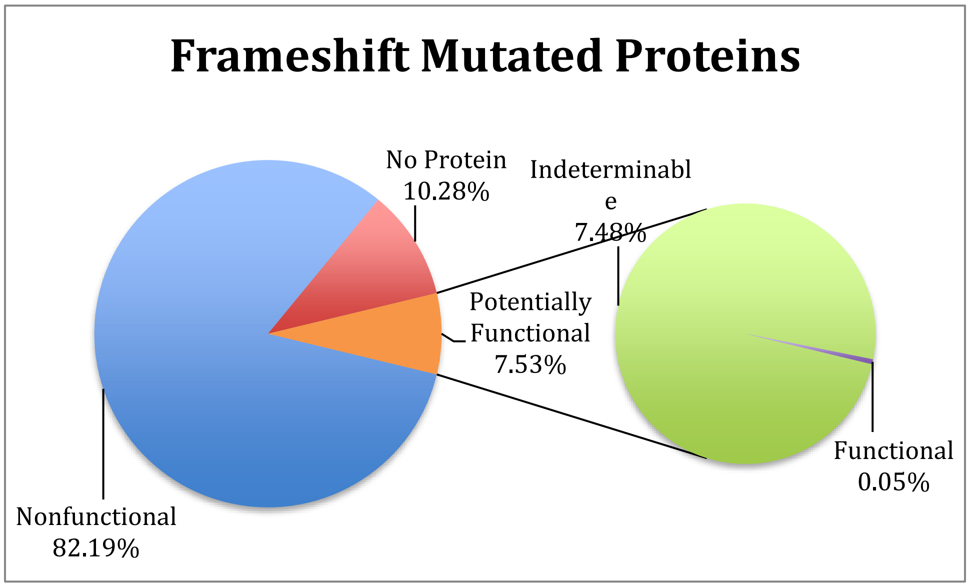


Figure 8. Frameshift mutation protein products. Approximately 0.05% of frameshift mutated protein products were functional.

The protein domains recognized by InterPro were used to create a protein domain landscape. Figure 9. shows the total number of protein domains that compose the mutated protein products and the non-mutated proteins.

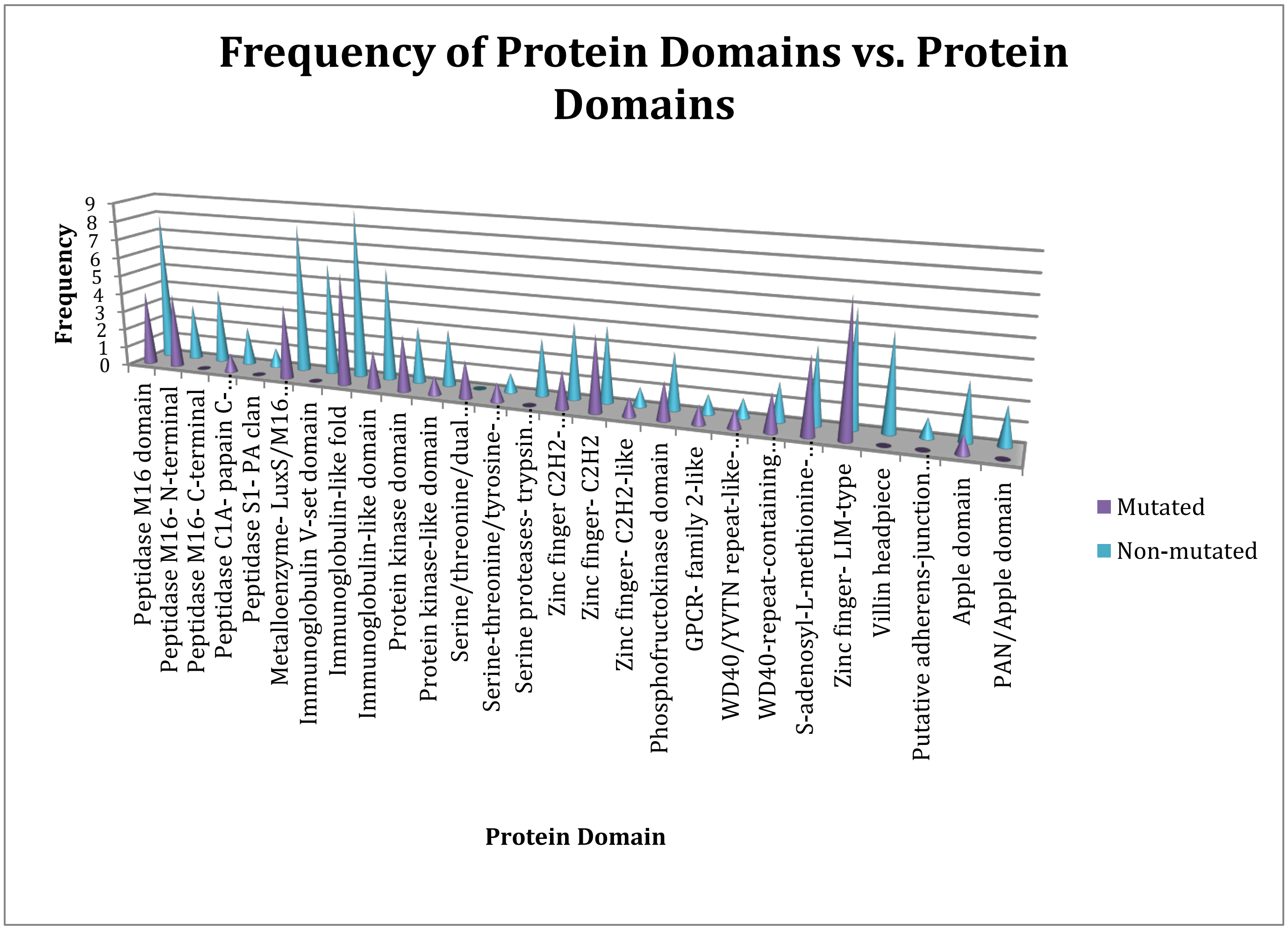
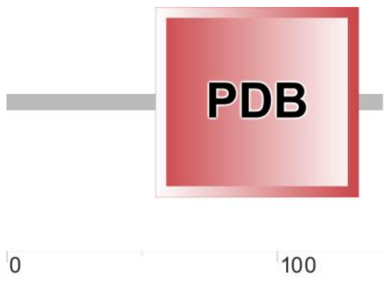
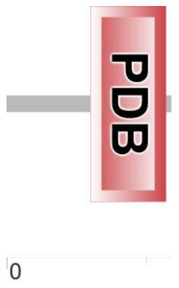


Figure 9. Frequency of protein domains vs. protein domains. The number of protein domains in the 23 frameshift-mutated genes' functional protein domains were counted and plotted. The same was done for the 23 corresponding non-mutated genes' proteins.

AB977775.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





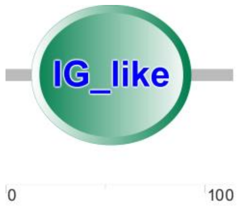
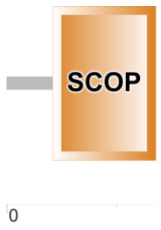
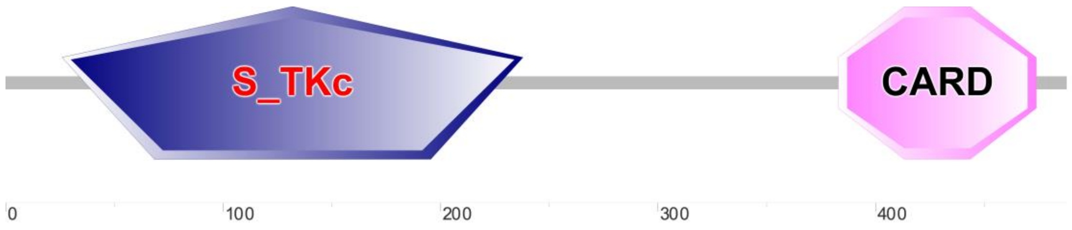


Figure 10. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the AB977775.1 mRNA record.

KU178438.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





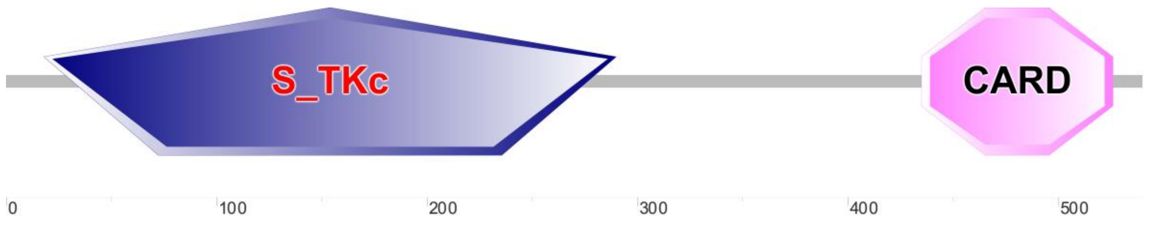
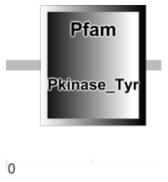
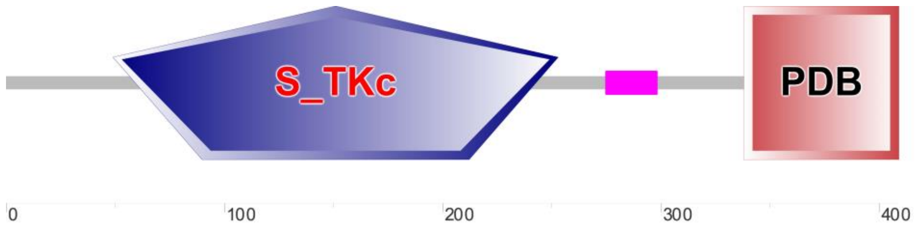


Figure 11. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the KU178438.1 mRNA record.

KU178529.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





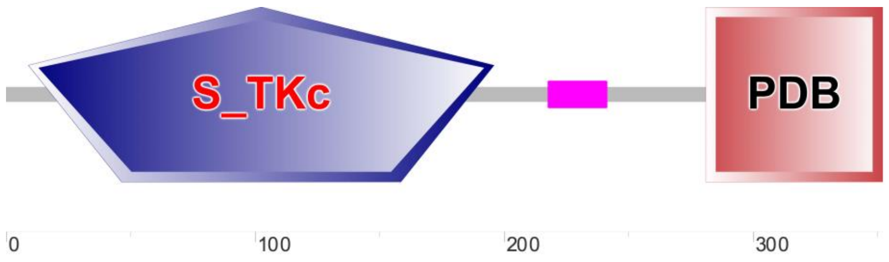
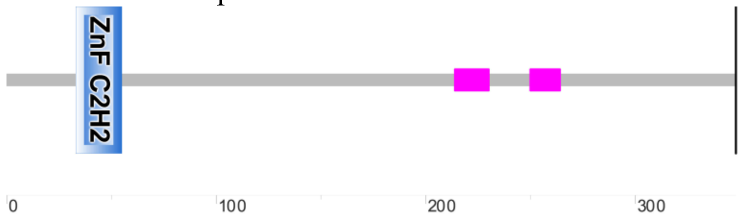
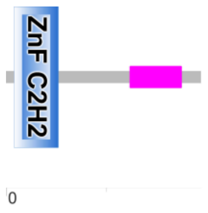


Figure 12. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the KU178529.1 mRNA record.

NM_001136509.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





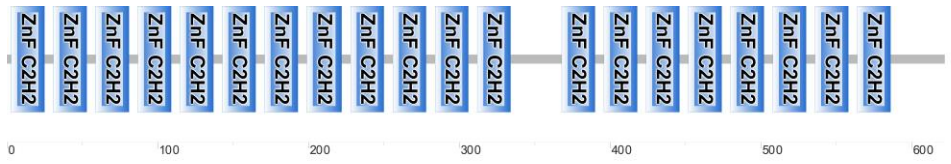
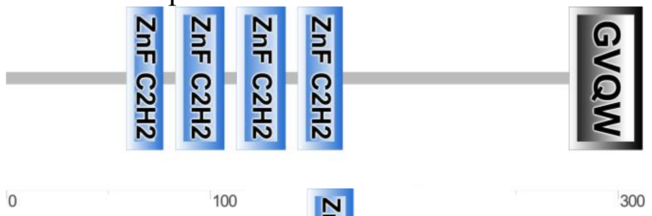
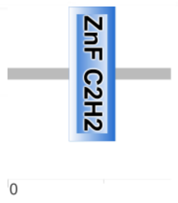


Figure 13. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001136509.1 mRNA record.

NM_001242928.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





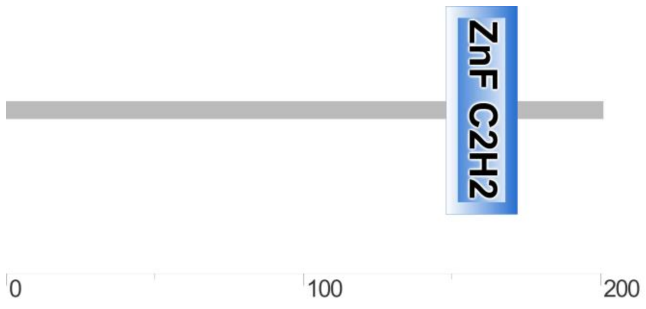
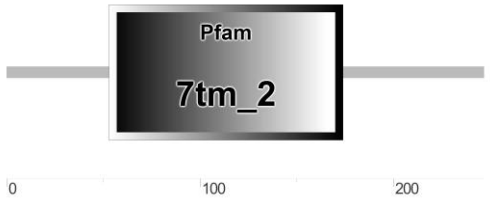
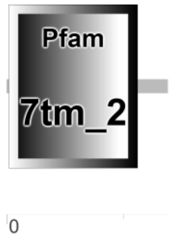


Figure 14. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001242928.1 mRNA record.

NM_001251883.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





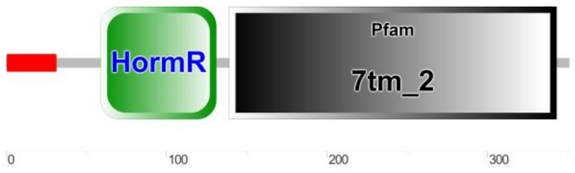
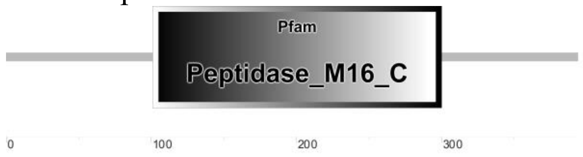
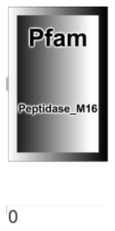


Figure 15. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001251883.1 mRNA record.

NM_001282944.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:





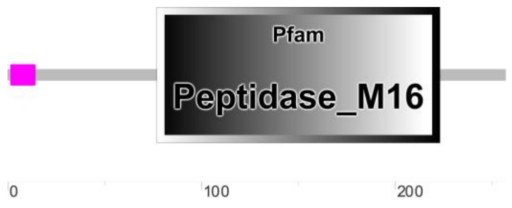
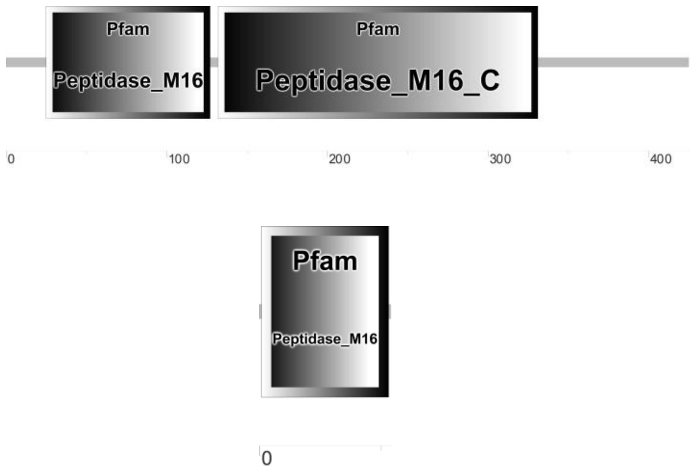


Figure 16. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001282944.1 mRNA record.

NM_001282946.1 Structural Protein Domain Architecture Analysis

The structural protein domain architectures of the CDS, frameshift mutated, and BLAST protein products are shown below in respective order:



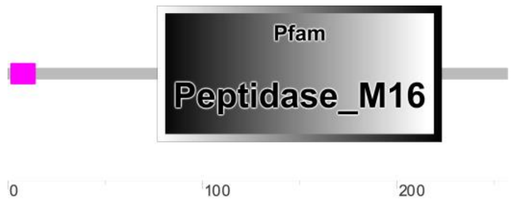


Figure 17. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001282946.1 mRNA record.

Data Analysis and Discussion

The data gathered from this study achieved the engineering goal. The structural landscape created by simulating frameshift mutations provided evolutionary information about frameshift mutated amino acid sequences that form functional proteins. The majority of frameshift mutated mRNA sequences encountered a premature stop codon that terminated the mRNA sequence. This resulted in 10.28% of the mutated sequences having no amino acid sequence at all and contributed to the 82.19% of the mutated sequences that coded for nonfunctional proteins. Of the remaining sequences, Figure 7. shows that the majority of potentially functional frameshift mutated amino acid sequences had a low bit score, suggesting a poor sequence alignment to known human proteins. In this study, 7.48% out of 7.53% (99.3%) of potentially functional proteins were classified as indeterminable because the protein domains of these proteins, if any exist, are currently unknown.

An important trend in Figure 9., the protein domain landscape, is that the mutated and non-mutated proteins have similar protein domain frequencies. This indicates that the protein domains were generally conserved through the frameshift mutation, and because protein domains are the structural and functional units of a protein, the mutated protein product likely had the same or a similar structure and function to the original protein. The structural protein domain landscapes showed two important results.

The first result was that for some proteins, the original protein's structure and function is conserved through frameshift mutations. In cases AB977775.1, NM_001136509.1, and NM_001251883.1, the original protein's structure and function was conserved through a frameshift mutation because the structural domain architecture of the original protein and the mutated protein product was identical.

Table 10. Conserved protein descriptions. The name and a description of each protein conserved through a frameshift mutation.

Conserved Protein Descriptions	
mRNA Accession Number	Protein Name
AB977775.1	T cell receptor beta chain V-D-J-region
NM_001136509.1	Zinc finger protein 843
NM_001251883.1	Vasoactive intestinal peptide receptor 1

The second result was that for other proteins, the mutated protein product evolved from the original protein through a frameshift mutation. In cases KU178438.1, KU178529.1, NM_001242928.1, NM_001282944.1, and NM_001282946.1, the original protein has evolved into the frameshift mutated protein product because the mutated protein product structural domain architecture is different than the original protein structural domain architecture, but identical to the BLAST protein structural domain architecture. Therefore, the frameshift mutated protein product had the same structure and function as the BLAST protein, which in all cases was a variation of the original protein. This indicated that the BLAST protein likely evolved from the original protein. In these five cases, an evolutionary pathway, via a frameshift mutation, from the original protein to the BLAST protein was shown.

Table 11. Original and mutated protein product descriptions. The name and a description of the original and mutated (evolved) proteins

Original and Mutated Protein Product Descriptions		
mRNA Accession Number	**Original Protein Name**	**Mutated Protein Name**
KU178438.1	Receptor-interacting serine-threonine kinase 2 (Signaling complex in immune system)	GIG30 (Regulation of apoptosis)
KU178529.1	Serine/threonine kinase 25 isoform 2 (Polarization of the Golgi Apparatus)	Serine/threonine-protein kinase 25 isoform 4 (Pro-apoptotic kinase)
NM_001242928.1	Zinc finger protein 410	Unnamed protein product (Moderately similar to Zinc finger protein 410)
NM_001282944.1	Peptidase (mitochondrial processing) alpha	Unnamed protein product (Highly similar to PMPA)
NM_001282946.1	Peptidase (mitochondrial processing) alpha	Unnamed protein product (Highly similar to PMPA)

Conclusions

The results of this simulation suggest that frameshift mutations can produce functional proteins that are the same as or likely evolved from the original protein. This is shown through the three frameshift mutated protein products that were conserved and the five frameshift mutated protein products that demonstrated possible evolutionary pathways. Although most scientists believe that natural selection, gene flow, random fluctuations in genetic composition due to environmental effects, and mutations such as missense, nonsense, and duplications are the only means of evolution, this study shows that frameshift mutations also have the potential to lead to evolution. The results gathered warrant further investigation and research into the evolutionary effects and potential of frameshift mutations.

Limitations and Assumptions

There were several limitations and assumptions made in this study. First, it was assumed that all human genes were present on the NCBI Nucleotide Database. Additionally, it was assumed that BLAST, InterPro, and SMART were updated when needed and functioning correctly. A major limitation was the time that the researcher had to complete this project. If more time were allotted, the researcher would have further explored the evolutionary potential of frameshift mutations, as well as the diseases they may cause. Computational resources were another limitation because the researcher only had access to nine computers. Access to more computers would have sped up processes such as calling BLAST and InterPro.

Applications and Future Experiments

There are many future experiments and applications to this project. First, the five mutated, functional protein products that were likely to have evolved from their original proteins could be traced back until humans had only the original protein. This would provide context as to when a frameshift mutation may have occurred to produce a mutated protein product. The other fifteen mutated, functional protein products each serve as case studies because the mutated protein product domain architecture is different than the original and BLAST protein domain architectures.

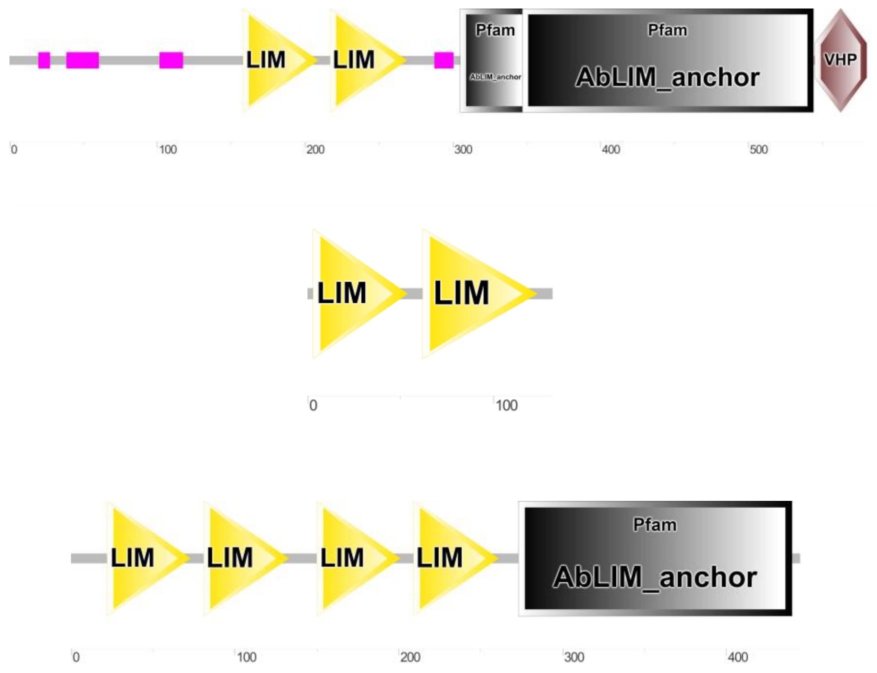


Figure 18. The structural domain architectures for the CDS, frameshift mutated, and BLAST protein products of the NM_001301027.1 mRNA record.

In the example above, the frameshift mutated protein product is functional, but does not correspond to a known human protein. For such proteins, homology modeling could be used to determine their structure and function. Another future experiment could be to run the simulation, but BLAST each mutated amino acid sequence against a database containing all organisms' proteins. This could provide more accurate ‘hits’ and additional functional proteins would likely be identified. One last future extension of this study is to change the location of the mutation from the beginning of the mRNA sequence to other parts, such as the middle or close to the end. Changing the position of the mutation would also change how much of the sequence is affected by the frameshift mutation and would produce a new set of unique results. The findings from this study also have applications to the real world. The identified mutated, functional protein products can help evolutionary biologists better understand how humans evolve and where humans come from. Running this simulation on the genes of other organisms could allow humans to implement a directed evolution approach in order to manipulate the behavior and genetic traits of that organism.

Literature Cited

Addington, A. M., Gauthier, J., Piton, A., Hamdan, F. F., Raymond, A., Gogtay, N., ... & Rouleau, G. A. (2011). A novel frameshift mutation in UPF3B identified in brothers affected with childhood onset schizophrenia and autism spectrum disorders. *Molecular psychiatry*, *16*(3), 238.

Biopython. (2013, June 1). Retrieved November 27, 2015, from http://biopython.org/wiki/Main_Page

Clancy, S. & Brown, W. (2008) Translation: DNA to mRNA to Protein. Nature Education 1(1):101

Collins, D. (2012, April 1). Prevalence of Genetic Conditions / Birth Defects. Retrieved November 22, 2015, from http://www.kumc.edu/GEC/prof/prevalnc.html

Environmental Biodetection Products Incorporated - EBPI-KITS.COM / EBPI.CA. (2011, July 1). Retrieved November 22, 2015, from http://www.ebpi-kits.com/TA98 Frame Shift Mutation.html

Genetic Disorders: MedlinePlus. (2014, May 13). Retrieved November 22, 2015, from https://www.nlm.nih.gov/medlineplus/geneticdisorders.html

Hayashi, K. (1991). PCR-SSCP: a simple and sensitive method for detection of mutations in the genomic DNA. *PCR methods and applications*, (1), 34-8.

Hinney, A., Schmidt, A., Nottebom, K., Heibult, O., Becker, I., Ziegler, A., ... & Hebebrand, J. (1999). Several mutations in the melanocortin-4 receptor gene including a nonsense and a frameshift mutation associated with dominantly inherited obesity in humans. *The Journal of Clinical Endocrinology & Metabolism*, *84*(4), 1483-1486.

Hu, J., & Ng, P. C. (2012). Predicting the effects of frameshifting indels. *Genome Biol*, *13*(2), R9.

Lapunzina, P., Aglan, M., Temtamy, S., Caparrós-Martín, J. A., Valencia, M., Letón, R., ... & Ruiz-Perez, V. L. (2010). Identification of a frameshift mutation in Osterix in a patient with recessive osteogenesis imperfecta. *The American Journal of Human Genetics*, *87*(1), 110-114.

Madden, T. (2013). The BLAST sequence analysis tool.

Nehrt, N. L., Peterson, T. A., Park, D., & Kann, M. G. (2012). Domain landscapes of somatic mutations in cancer. *BMC genomics*, *13*(Suppl 4), S9.

Rosentiel, S. (2014, February 11). *Frameshifts, deletions, and dynamic mutations*. Lecture presented at Molecular Genetics in Brandeis University, Waltham.

Streisinger, G., Okada, Y., Emrich, J., Newton, J., Tsugita, A., Terzaghi, E., & Inouye, M. (1966, January). Frameshift mutations and the genetic code. In *Cold Spring Harbor Symposia on Quantitative Biology* (Vol. 31, pp. 77-84). Cold Spring Harbor Laboratory Press.

Welcome to Python.org. (1991, August 1). Retrieved November 27, 2015, from https://www.python.org/doc/essays/blurb/

Zhong, Q., Simonis, N., Li, Q. R., Charloteaux, B., Heuze, F., Klitgord, N., ... & Vidal, M. (2009). Edgetic perturbation models of human inherited disorders. *Molecular systems biology*, *5*(1), 321.

Zvelebil, M., & Baum, J. (2007). *Understanding bioinformatics*. Garland Science.

Appendix 1.A

accessNCBI

```
def accessNCBI ():
   import os
   from Bio import Entrez, SeqIO
   Entrez.email = "nshah2@wpi.edu"
   start=0
   moreRecords = True
   save_path = "./mRNARecords"
   while moreRecords:
      handle = Entrez.esearch(db="nucleotide", term="Human[Organism] AND mRNA[Filter]",
usehistory="y", retmax="20", retstart=str(start))
      record = Entrez.read(handle)
      idList = record['IdList']
      for id in idList:
         net_handle = Entrez.efetch(db="nucleotide", id=str(id), rettype="gb", retmode="text")
         record = SeqIO.read(net_handle, "genbank")
         for seqFeature in record.features:
            if seqFeature.type == 'CDS':
               filename = os.path.join(save_path, str(record.id)+".txt")
               out_handle = open(filename, "w")
               SeqIO.write(record,out_handle, "genbank")
               out_handle.close()
               break
         net_handle.close()
      start+=20
      if (len(idList)<20):
         moreRecords=False
   handle.close()
```

Appendix 1.B

callBLAST

```
def callBLAST ():
    from Bio.Blast import NCBIWWW
    from Bio.Blast import NCBIXML
    from Bio import SeqIO
    from Bio.Alphabet import generic_rna
    import os
    from Bio.Seq import Seq
    import shutil
    save_path = "./BLASTResults"
    for file in os.listdir("./mRNARecords"):
        out_handle = open("./mRNARecords/"+file, "r")
        record = SeqIO.read(out_handle, "genbank")
        for seqFeature in record.features:
            if seqFeature.type == 'CDS':
                for i in range(1,3):
                    cds=Seq(str(record.seq[seqFeature.location.start+i:]), generic_rna)
                    BLASTSeq=cds.translate(to_stop=True)
                    BLASTFile = os.path.join(save_path, str(record.id)+"-"+str(i)+"-frame.txt")
                    if  not (os.path.isfile(BLASTFile)):
                        BLAST_handle = open(BLASTFile, "w")
                        if (len(BLASTSeq)!=0):
                            try:
                                result_handle = NCBIWWW.qblast(program="blastp", database="nr",
sequence=BLASTSeq, entrez_query="txid9606[ORGN]",
matrix_name='BLOSUM62',word_size='2',expect='0.0001',gapcosts='11
1',composition_based_statistics='no adjustment')
                                blast_record = NCBIXML.read(result_handle)
                                result_handle.seek(0)
                                BLAST_handle.write(result_handle.read())
                            except:
                                BLAST_handle.write("Failed")
                        BLAST_handle.close()
        out_handle.close()

    '''
    E_VALUE_THRESH = 0.0001
                    for alignment in blast_record.alignments:
                        for hsp in alignment.hsps:
                            if (hsp.expect < E_VALUE_THRESH):
                                    print('****Alignment****')
                                    name = str(alignment.title)
                                    if (name.index('>')!=-1):
```

```
                                nameStop = name.index('>')
                                name = name[:nameStop-1]
                            SeqIO.write(str(('sequence:', name)),in_handle, "genbank")
                            SeqIO.write(str(('length:', alignment.length)),in_handle, "genbank")
                            SeqIO.write(str(('score:', 100 * hsp.identities /
alignment.length)),in_handle, "genbank")
                            SeqIO.write(str(('e value:', hsp.expect)),in_handle, "genbank")
                            SeqIO.write("",in_handle, "genbank")

                SeqIO.write("",in_handle, "genbank")


    print "Starting BLAST"
    result_handle = NCBIWWW.qblast(program="blastp", database="nr", sequence=BLASTSeq,
entrez_query="txid9606[ORGN]",
matrix_name='BLOSUM62',word_size='2',expect='10',gapcosts='11
1',composition_based_statistics='no adjustment')
    print "Finish BLAST"
    blast_record = NCBIXML.read(result_handle)
    print "Reading BLAST"
    E_VALUE_THRESH = 0.04
    for alignment in blast_record.alignments:
        for hsp in alignment.hsps:
            if (hsp.expect < E_VALUE_THRESH) and (len(BLASTSeq)==alignment.length):
                print('****Alignment****')
                name = str(alignment.title)
                if (name.index('>')!=-1):
                    nameStop = name.index('>')
                    name = name[:nameStop-1]
                print('sequence:', name)
                print('length:', alignment.length)
                print('score:', 100 * hsp.identities / alignment.length)
                print('e value:', hsp.expect)
                print""
        break
    '''
```

Appendix 1.C

readBLAST

```
import os
import shutil
from Bio.Blast import NCBIXML
from Bio import SeqIO, Entrez
from Bio.Alphabet import generic_rna
from Bio.Seq import Seq

for file in os.listdir("./BLASTResults"):
    alreadyIsProtein=os.path.join("FunctionalProteins",file)
    if not (os.path.isfile(alreadyIsProtein)):
        if (str(file)!=".DS_Store" and os.stat("./BLASTResults/"+file).st_size != 0):
            out_handle = open("./BLASTResults/"+file, "r")
            if not(out_handle.readline()=="Failed"):
                out_handle.seek(0)
                blast_record = NCBIXML.read(out_handle)
                if blast_record.alignments:
                    shutil.copy("./BLASTResults/"+file,"./FunctionalProteins")

Entrez.email = "nshah2@wpi.edu"
save_path="./proteinResults"
for file in os.listdir("./FunctionalProteins"):
    alreadyIsProtein1=os.path.join("proteinResults",file)
    if not (os.path.isfile(alreadyIsProtein1)):
        counter = 0
        mRNAFile=file[:file.index('-')]
        frameShift = int(file[file.index('-')+1])
        mRNA_handle = open("./mRNARecords/"+mRNAFile+".txt", "r")
        record = SeqIO.read(mRNA_handle, "genbank")
        for seqFeature in record.features:
            if seqFeature.type == 'CDS':
                cds=Seq(str(record.seq[seqFeature.location.start:]), generic_rna)
                cds=cds.translate(to_stop=True)
                Mutatedcds=Seq(str(record.seq[seqFeature.location.start+frameShift:]), generic_rna)
                MutatedSeq=Mutatedcds.translate(to_stop=True)
                counter=counter+1
            if counter>1:
                print file
                print"multiple CDS"
        mRNA_handle.close()
        if (str(file)!=".DS_Store" and os.stat("./FunctionalProteins/"+file).st_size != 0):
            out_handle = open("./FunctionalProteins/"+file, "r")
            blast_record = NCBIXML.read(out_handle)
            #print "NEW SEQUENCE"
            #print ""
            bestBit = 0
            for alignment in blast_record.alignments:
                for hsp in alignment.hsps:
                    if hsp.bits>bestBit:
                       bestBit=hsp.bits
                       bestAlignment=alignment
                       bestHsp=hsp
            proteinResult = os.path.join(save_path, file)
            protein_handle = open(proteinResult, "w")
            protein_handle.write("Original Protein: " + str(record.description)+"\n\n")
            protein_handle.write("CDS: " + str(cds)+"\n\n")
            protein_handle.write("Frameshifted Sequence: " + str(MutatedSeq)+"\n\n")
            protein_handle.write("BLAST Sequence: " + str(bestHsp.sbjct)+"\n\n")
            protName = str(bestAlignment.title)
            protID = protName[protName.index("|")+1:]
            protID=protID[:protID.index("|")]
            net_handle = Entrez.efetch(db="protein", id=protID, rettype="gb", retmode="text")
            protRecord = SeqIO.read(net_handle, "genbank")
            net_handle.close()
            protein_handle.write("Full BLAST Sequence: " + str(protRecord.seq)+"\n\n")
            protein_handle.write("Mutated Protein: " + protName+"\n\n")
            protein_handle.write("Alignment Score: " +
str(100*bestHsp.identities/(bestHsp.sbjct_end-bestHsp.sbjct_start+1))+"\n\n")
            protein_handle.write("Bit Score: " + str(bestHsp.bits)+"\n\n")
            protein_handle.write("Percent Subject Sequence: " + str(100*(bestHsp.query_end-
bestHsp.query_start+1)/bestAlignment.length)+"\n\n")
            '''
            print('****Alignment****')
            print('sequence:', bestAlignment.title)
            print('length:', bestAlignment.length)
            print('alignment score:', 100*bestHsp.identities/(bestHsp.sbjct_end-
bestHsp.sbjct_start+1))
            print('bit score:', bestHsp.bits)
            print('percent subject sequence: ', 100*(bestHsp.query_end-
bestHsp.query_start+1)/bestAlignment.length)
            print('e value:', bestHsp.expect)
            print str(file)
            print""
            '''
            protein_handle.close()
            out_handle.close()
```

Acknowledgements

The author would like to thank many people for constantly supporting him throughout this project. Dr. Dmitry Korkin, a professor at WPI, was very helpful with developing the project idea and refining the project methodology. Mrs. Angela Taricco consistently provided insight about improving the project, and was very generous when allowing the researcher to use her computational resources. Ms. Maria Borowski helped the author write this research paper. The author would also like to thank his parents for their unwavering support.