Fall 2012

A Database-Driven Website of β-Thalassemia Mutations

Vanessa Gaeke
San Jose State University

A Database-Driven Website of β-Thalassemia Mutations

A Writing Project

Presented to

The Faculty of the Department of Computer Science
And the Department of Biological Science
San José State University

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

by
Vanessa Gaeke
Advisor: Professor Sami Khuri
December 2012

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Sami Khuri Date
Department of Computer Science

Dr. Rachael French Date
Department of Biological Science

Dr. Teng Moh Date
Department of Computer Science

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research Date

Table of Contents

Acknowledgement

This project is the result of the efforts of many individuals. I would like to express my deep appreciation and gratitude to Dr. Khuri for his patient guidance and mentorship. This project would not have become reality if it weren't for Dr. Khuri's encouragement. I am grateful to have had the opportunity to work with him. I would also like to recognize and express my gratitude to Mrs. Natasha Khuri, who had offered, in addition to generous support in her time, many great ideas to my project that have made this project exciting and challenging.

I would like to thank Dr. French for her clear-cut explanation that greatly improved my understanding of the many aspects in genetics, and for her invaluable comments and suggestions to my thesis. I am fortunate to have learned an immense amount from Dr. French. I am grateful to Dr. Moh for his time and kind advice on my project. Lastly, I would like to recognize and thank Ms. Belinda Giardine at Pennsylvania State University, who tirelessly provided me with raw data files of mutation data, without which this project would be multiple folds more laborious.

Abstract

β-thalassemia is a genetic disease caused by mutations of the hemoglobin gene β (HBB). Over two hundred β-thalassemia mutations are known to date. While the pathological consequence of each β-thalassemia mutation has been well documented, the effect of a mutation at the molecular level of its DNA sequence is usually not illustrated for beginners. This paper describes the cause of β-thalassemia and a database-driven website containing mutant HBB sequences altered by β-thalassemia mutations. β-thalassemia mutations and their relevant data are first collected from the HbVar Database of Pennsylvania State University and loaded into a PostgreSQL database. Wild type HBB sequences affected by each mutation are then simulated and stored in the same database. At last a Perl script queries this database to dynamically create a web page with a list of mutant HBB sequences for view.

Chapter 1: Overview

β-thalassemia is a genetic disease common among Mediterranean and Asian people (Vichinsky, MacKlin, Waye, Lorey, & Olivieri, 2005). It is an autosomal recessive disorder characterized by a reduction or absence of the β subunit of hemoglobin, causing anemia of various degrees of severity. There are estimated to be at least 60,000 newborns severely affected with β-thalassemia each year (Higgs, Engel, & Stamatoyannopoulos, Thalassemia, 2011). However, there is no cure at present. β-thalassemia patients require regular blood transfusions and iron chelation therapy (Makhoul, 2005).

There are approximately two hundred documented mutations that cause β-thalassemia (Agouti, 2008). Mutations can occur within the hemoglobin β gene (HBB) as well as its regulatory elements. They impair β-globin synthesis in many ways. Some cause abnormal mRNA splicing, some result in nonfunctional mRNA, still others lead to abnormal post-transcriptional modification of mRNA (Forget, Martin, Higgs, Nagel, & Bunn, 2000). Many of these mutations are single base changes, but their effects are disastrous. The severity of the disease varies depending on the combination of an individual's mutant alleles.

Visualizing β-thalassemia mutations' effect on HBB at the nucleic acid level can be challenging. Many seemingly similar mutations do not actually have the same effect on the β-globin gene. A website would be a great medium to demonstrate how HBB mutations modify the HBB sequence. It can highlight base changes, cryptic splice sites, and premature stop codons. It also can use font color, size, and style to differentiate segments of the gene affected by each mutation. In addition, a live website can be accessed anywhere with internet access.

Unfortunately, whereas there are excellent hemoglobinopathy databases such as the Globin Gene Server (http://globin.bx.psu.edu/) at Pennsylvania State Hershey Medical Center, which documents thoroughly β-thalassemia mutations, there does not exist a website that illustrates the effect of each β-thalassemia mutation on the entire HBB sequence at the nucleic acid level.

As a result, this thesis aims to accomplish two goals. First, it explains human hemoglobin, the role of β-globin in hemoglobin, and the mutations that adversely impact β-globin production and lead to β-thalassemia. Second, it chronicles the development of a database-driven web site that demonstrates the molecular change in the human hemoglobin β gene sequence caused by over two hundred β-thalassemia mutations.

This website is designed for students as well as educators and researchers. It focuses on the organization as well as the presentation of data, allowing users quick and easy access to well-documented information on any β-thalassemia mutation. This project is unique in its kind because the existing hemoglobinopathy databases, such as the Globin Gene Server developed at Pennsylvania State Hershey Medical Center (http://globin.bx.psu.edu/hbvar/menu.html), are primarily designed for field specialists.

The rest of the thesis can be read as two parts. The first part consists of chapters 2, 3, and 4. Chapter 2 studies the human hemoglobin as the first step toward understanding β-thalassemia. It discusses in detail the hemoglobin gene clusters, including their anatomical structure and erythropoietic regulation of genes in them. Chapter 3 focuses on the hemoglobin β gene mutations which lead to β-thalassemia. It explains in depth its molecular makeup, gene expression, and covers a number of illustrative mutations that lead to β-thalassemia. Chapter 4 discusses the β-thalassemia syndrome, its phenotypes, and its changing epidemiology in North America.

The second part of this thesis chronicles the implementation of a website that allows users to query into a database consisting of more than two hundred documented β-thalassemia mutations, view the sequence of any of these mutant hemoglobin β alleles, and learn the effect it has on β-globin synthesis. Chapter 5 provides an introduction to the relational database theory, discusses the PostgreSQL database, and covers the schema created for this project. Chapter 6 provides an introduction to the PERL scripting language, covering two of the modules used for this project, and cascade style sheets (CSS), a language used to enhance the presentation of web pages.

The next chapter discusses hemoglobin in depth.

Chapter 2: Hemoglobin

2.1: Introduction

Hemoglobin is an essential protein found in red blood cells. It carries oxygen from the lungs to the rest of the body for energy synthesis, and collects the resultant carbon dioxide back to the lungs to be dispensed through exhalation (An Overview of Hemoglobin, 2002).

Hemoglobin is composed of four subunits, each of which contains a heme group that carries an oxygen molecule. These four subunits change according to the stage of development (Forget & Hardison, 2009). However, these four subunits are always comprised of two proteins encoded by the α-globin gene cluster on chromosome 16, and two proteins encoded by the β-globin gene cluster on chromosome 11. For instance, fetal hemoglobin is comprised of two α-globins made by the α-globin genes on the α-globin gene cluster, and two γ-globins made by the γ-globin gene in the β-globin gene cluster. Adult hemoglobin, on the other hand, is composed mainly of two α-globins from the α-globin gene cluster, and two β-globins from the β-globin gene cluster (Forget & Hardison, 2009).

The genes on the α-globin gene cluster and the β-globin gene cluster in mammals that produce the subunits for hemoglobin are arranged in the order of their expression throughout development. The appropriate genes on each cluster are expressed coordinately between the two loci on chromosome 11 and chromosome 16, resulting in balanced production of α-globins and β-globins needed for the synthesis of normal hemoglobin. The mechanisms that accomplish this task still elude our understanding (Forget & Hardison, 2009).

It is not a coincidence that vertebrates employ such a coordinated and balanced mechanism for hemoglobin synthesis. The α-globin gene cluster and the β-globin gene cluster actually evolved, through duplication, from a single ancestral hemoglobin gene (Forget & Hardison, 2009). Their separation onto different chromosomes allowed them to diverge into different genomic contexts, yet both gene clusters have retained their well-orchestrated regulation of gene expression for normal hemoglobin synthesis (Forget & Hardison, 2009).

Section 2.2 covers the structure of the α-globin gene cluster. Section 2.3 covers the β-globin gene cluster. Section 2.4 and Section 2.5 explain the structure and regulation of genes in these gene clusters.

2.2: α-globin Gene Cluster

The α-globin gene cluster is located on the short arm of chromosome 16. It is comprised of two ζ-globin genes followed by two α-globin genes. Embryonic hemoglobin is made up of two ζ-globins produced by the two ζ-globin genes on the α-globin gene cluster, and two ε-globins produced by the two β-globin genes on the β-globin gene clusters. Adult hemoglobin is made up of two α-globins produced by the two α -globin genes on the α-globin gene cluster, and two β -globins produced by the two β-globin genes on the β-globin gene cluster. The genes on the α-globin gene cluster are in the same order as their expression throughout development on the short arm of chromosome 16 (Figure 2.1).

Section 2.3 covers the structure of the β-globin gene cluster.

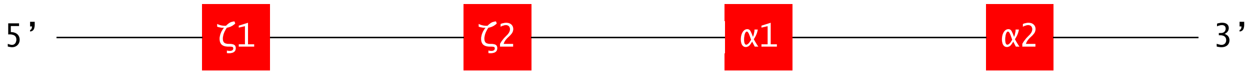


Figure 2.1: α-globin gene cluster

2.3: β-globin Gene Cluster

Like the α-globin gene cluster, the genes on the β-globin gene cluster are organized in the same order as their expression in development. The ε-globin gene is expressed at embryonic stage; γ-globin genes are expressed at fetal stage; both δ-globin genes and β-globin genes are expressed after birth. Embryonic hemoglobin has two ζ-globins produced by the two ζ-globin genes in the α-globin gene cluster, and two ε-globins produced by the single ε-globin genes in the β-globin gene cluster. Fetal hemoglobin has two α-globins produced by the two α-globin genes in the α-globin gene cluster, and two γ-globins produced by the two γ-globin genes in the β-globin gene cluster. 97% of the adult hemoglobin is comprised of two β-globins produced by the β-gene cluster, and two α-globins produced by the α-gene cluster. The remaining 3% is comprised of two δ-globins produced by the β-gene cluster, and two α-globins produced by the α-gene cluster on the short arm of chromosome 11 (Figure 2.2).

Section 2.4 explains the structure of genes in the α- and β-gene clusters.

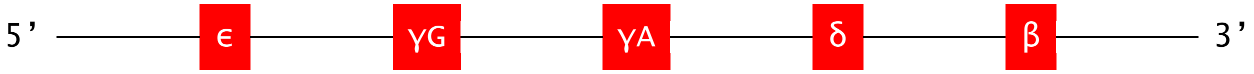


Figure 2.2: β-globin gene cluster

2.4: Structure of Globin Genes

Human DNA, or deoxyribonucleic acids, is composed of four different nucleic acids called nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). Genes are segments of DNA that encode protein. In the case of the human hemoglobin gene β (HBB), it encodes the β-globin protein that is used as building blocks for hemoglobin in red blood cells. Therefore, HBB is a sequence of combination of these four bases: A, C, G, and T.

All the globin genes have a similar structure. They all have three exons and two intervening sequences called introns sandwiched between the three exons (Figure 2.3).

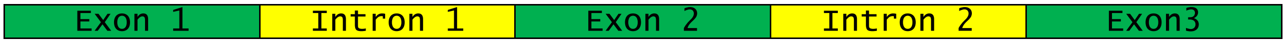


Figure 2.3: Globin gene structure

Exons contain the coding sequence of a gene that actually encodes protein -- ribosome translates a coding sequence from mature mRNA in the cytoplasm into amino acids that build a globin polypeptide (St. Clair, 2009). A coding sequence is a sequence embedded within the exons of a gene that begins with ATG and ends with either TAG, TGA, or TAA. In the case of the hemoglobin β gene, its coding sequence starts at the 51st base of exon 1 with the start codon ATG, encompasses the entire exon 2, and terminates at the 130th base of exon 3 with a stop codon TAA (Figure 2.4). Segments at the beginning of the first exon and towards the end of the last exon not part of the coding sequence are called the un-translated regions or UTR (St. Clair, 2009).

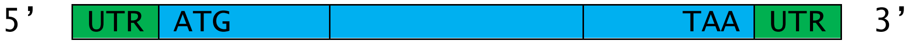


Figure 2.4: HBB's coding sequence and UTR with respect to its three exons

An intron begins with a GT dinucleotide and ends with an AG dinucleotide (Figure 2.5). This is called the GT-AG rule that ensures proper splicing of introns (Forget & Hardison, 2009). Although an intron does not encode proteins, it serves other important functional purposes. For example, the existence of introns makes alternative splicing possible. Alternative splicing is a process which removes introns as well as some exons from the precursor mRNA. It creates a wider range of exon combinations in mature mRNA, allowing the same sequence to encode more than one gene and thus make more than one protein. Introns also house regulatory elements like enhancers, and transcription factor binding sites (Forget & Hardison, 2009).

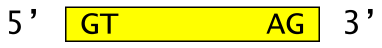


Figure 2.5: Consensus sequence of an Intron

Section 2.5 discusses the regulation of genes in the α- and β-gene clusters.

2.5: Regulation of Globin Genes

Both the α-globin gene cluster and the β-globin gene cluster are regulated by promoter and enhancer sites on the DNA of their respective chromosomes (Forget & Hardison, 2009). A promoter region is a DNA sequence that interacts with RNA polymerase II and its accessory factor to determine the transcription start site of a gene and initiate transcription. An enhancer region is a DNA sequence that increases the activity of promoter region and drives tissue-specific and developmentally stage-specific gene expressions (St. Clair, 2009). Promoter sites are usually found upstream of the gene's coding sequence whereas enhancer sites can be located at a considerable distance from the gene, on either side of the gene, or inside of the gene (St. Clair, 2009).

There are two most highly conserved patterns called motifs in a globin gene promoter region: the TATA box and the CAT box (Figure 2.6). The TATA box has a pattern of ATAAA at approximately 25 - 30 base pairs upstream of the transcription start site. The CAT box has a pattern of CCAAT at approximately 40 - 250 base pairs upstream of the transcription start site. As with the promoter region, there are also motifs in an enhancer region. One of the most highly conserved motifs in a globin gene enhancer region is the GATA-binding site, which has a sequence of GATA preceded by A or T and succeeded by A or G. Mutations in the promoter or enhancer region can lead to β-thalassemia as well as mutations in the HBB gene. For example, there is an enhancer element of a GATA-binding site at less than 1000 bases downstream of the β-globin gene whose function is perhaps related to the delay in expression of the β-globin gene until birth (Forget & Hardison, 2009).

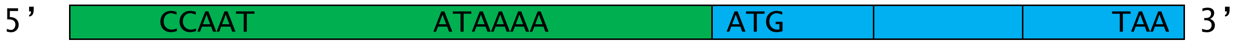


Figure 2.6: HBB’s promoter elements with respect to its coding sequence

Chapter 3 discusses the human hemoglobin β gene (HBB). It covers its anatomical structure, gene expression, and mutations with examples.

Chapter 3: Hemoglobin β Gene (HBB)

3.1: Introduction

The complete DNA sequence of the human hemoglobin β gene (HBB) consists of 742 nucleotides. HBB has three exons and two introns sandwiched between the three exons (Figure 2.3). There are 626 nucleotides in the coding sequence and 480 nucleotides in the intervening sequences. Exon one is 142 nucleotides long, exon two 223 nucleotides long, and exon three 261 nucleotides long (Figure 3.1). The introns consist of 130 and 850 nucleotides respectively.

HBB is immediately preceded by its promoter region (Figure 2.6). There are 5 main motifs in HBB's promoter region: BRE at -37 (GGGCTGG), TATA box at -31 (CATAAAAG), Inr at +1 (TTACATT), MTE at +27 (ACAACTG), and DPE at +32 (AGCAA) (Forget & Hardison, 2009). HBB's promoter region is immediately preceded by its proximal enhancer region, which has two main motifs: the CAT BOX of the pattern CCAAT, and the CACC box of the pattern CACCC.

Every three nucleotides in the coding sequence form a group called codon that maps to one of the twenty amino acids. As a result, the length of a coding sequence should always be a multiple of three. The first codon of a coding sequence, also called the start codon, is ATG, which forms methionine. The last codon, also called the stop codon, is either TAA, TAG, or TGA, none of which translates to an amino acid. Nucleotides in introns do not form codons since they do not encode protein. HBB's coding sequence encodes 147 amino acids.

The start codon is different from the start of the first exon. Likewise, the stop codon is not the end of the last exon. An un-translated region (UTR) is the portion of an exon that is not part of the gene's coding sequence. For instance, the portion of the first exon preceded by the start codon is called the 5' UTR, and the portion of the last exon succeeding a stop codon is called the 3' UTR. All other exons are entirely part of the coding sequence (Figure 2.4).

Section 3.2 discusses gene expression, an important concept for understanding mutations. Section 3.3 studies several examples of HBB mutations.

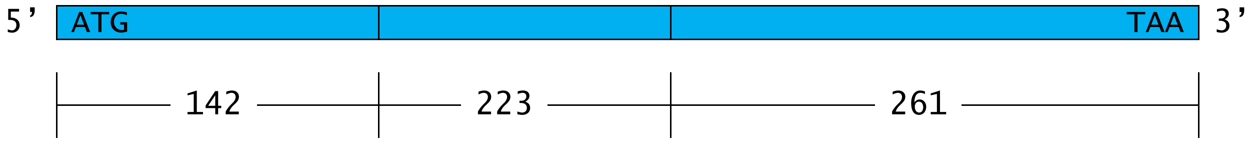


Figure 3.1: length of HBB's coding sequence

3.2: HBB Expression

Gene expression is the process by which information from a gene is used in the synthesis of a functional gene product, protein in most cases (St. Clair, 2009). Gene expression is the mechanism that determines which genes are expressed or not expressed in any given cell in an organism when all cells in an organism share the same DNA. In each cell type, a specific set of genes is activated, and the proteins they encode give the cell the characteristics of a muscle cell, bone cell, nerve cell, etc (St. Clair, 2009).

The level of expression of the HBB gene determines the amount of β-globins a cell is capable of producing. Under normal conditions, HBB produces a sufficient amount of β-globins to match the amount of α-globins to form healthy hemoglobin as discussed in chapter 2. When HBB is mutated, this results in β-thalassemia due to an imbalance between the α- and β-globin chains.

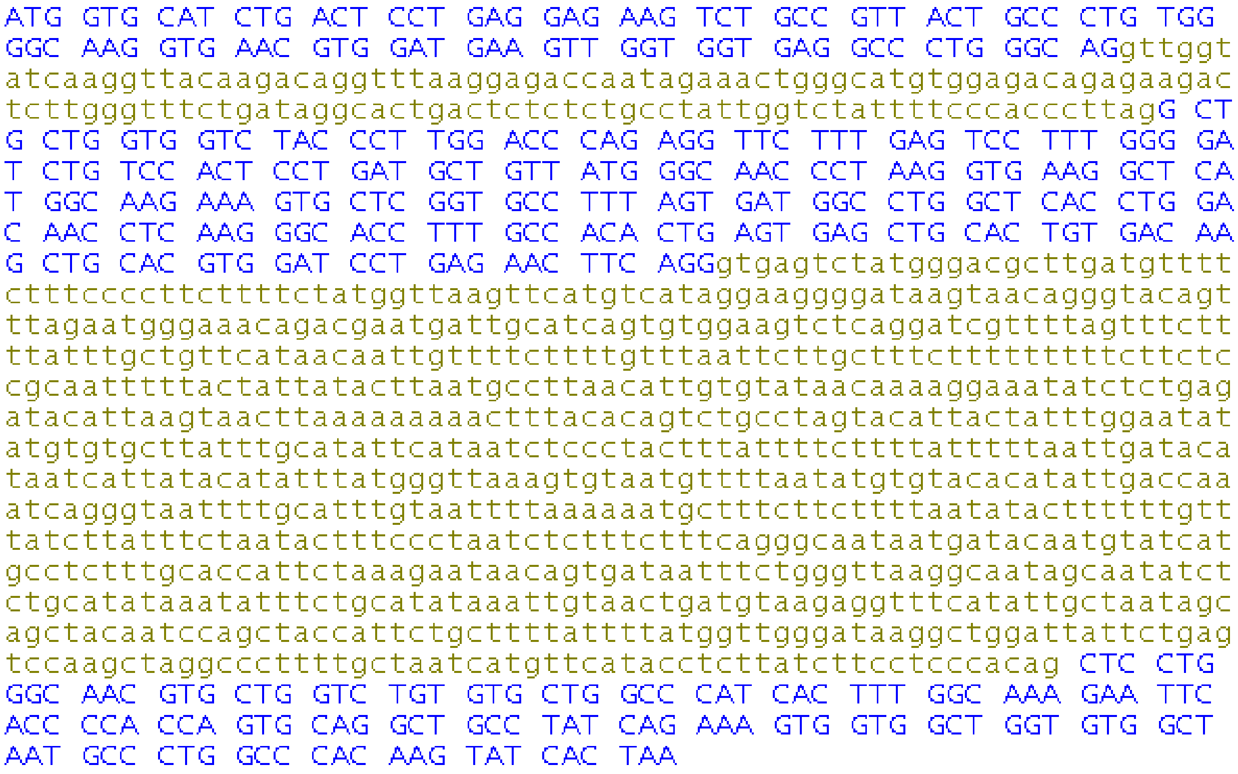
Section 3.2 demonstrates the differences between mRNA of a normal HBB allele and that of a mutant one with many examples.

3.3: HBB Mutations

An HBB allele that produces an insufficient amount of β-globins, is referred to as the β^+ type. An HBB allele that produces no β-globins at all is referred to as the β^0 type (Beta Thalassemia, 2009).

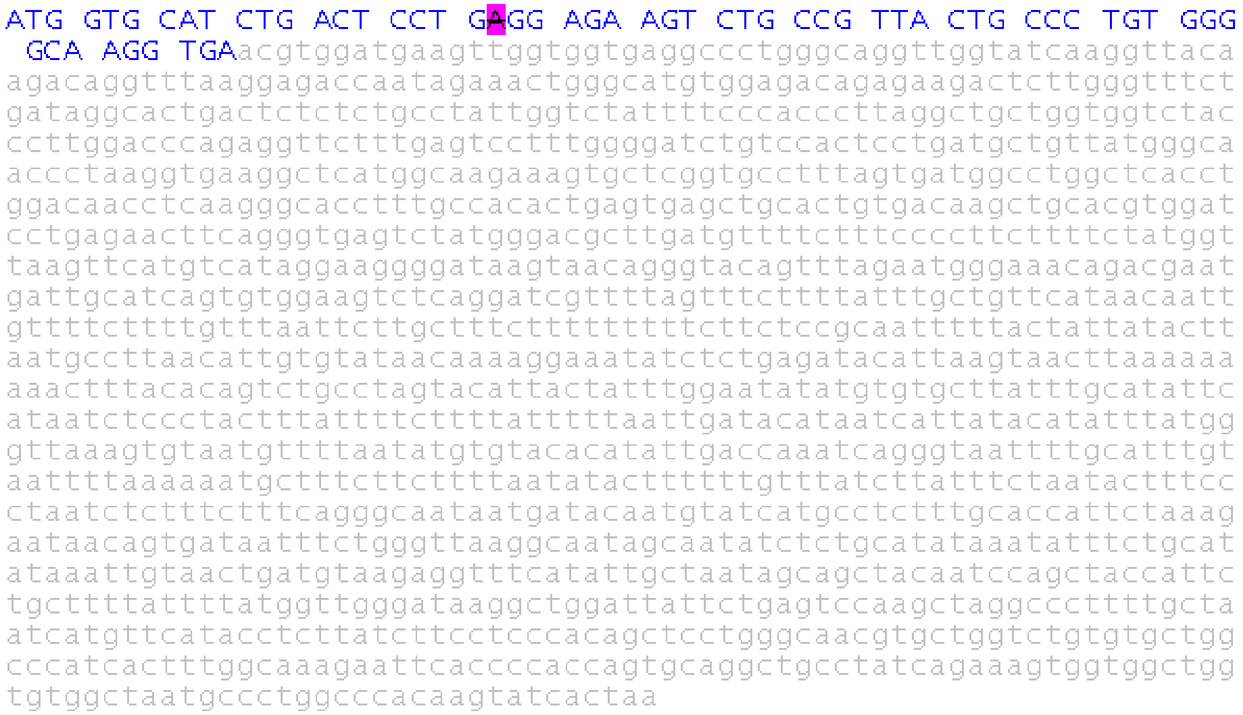
Mutations can occur anywhere in the DNA sequence of a gene. HBB mutations either produce non-functional β-globin protein (β^0) or a reduced amount of normal mRNA which could be translated into functional β-globin protein (β^+). There are three types of mutations that lead to β-thalassemia: base substitution, insertion, and deletion (Pierce, 2012).

A single nucleotide deletion in a gene leads to a frame-shift mutation. For example, a single-base-pair deletion in the sixth codon (GAG) of HBB leads to a frameshift mutation, and, ultimately, a premature stop codon due to the frame-shift. The mutant mRNA differs from the wild type allele after the first five amino acids, and the mutant polypeptide chain is only 18 amino acids long (Figure 3.7), compared with the wild type of 147 amino acids (Figure 3.6). This is a β^0 type mutation because little or no β-globin will be produced as a result.



```
ATG GTG CAT CTG ACT CCT GAG GAG AAG TCT GCC GTT ACT GCC CTG TGG
GGC AAG GTG AAC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AGgttggt
atcaaggttacaagacaggtttaaggagaccaatagaaactgggcatgtggagacagagaagac
tcttgggtttctgataggcactgactctctctgcctattggtctattttcccacccttagG CT
G CTG GTG GTC TAC CCT TGG ACC CAG AGG TTC TTT GAG TCC TTT GGG GA
T CTG TCC ACT CCT GAT GCT GTT ATG GGC AAC CCT AAG GTG AAG GCT CA
T GGC AAG AAA GTG CTC GGT GCC TTT AGT GAT GGC CTG GCT CAC CTG GA
C AAC CTC AAG GGC ACC TTT GCC ACA CTG AGT GAG CTG CAC TGT GAC AA
G CTG CAC GTG GAT CCT GAG AAC TTC AGGgtgagtctatgggacgcttgatgtttt
ctttccccttcttttctatggttaagttcatgtcataggaaggggataagtaacagggtacagt
ttagaatgggaaacagacgaatgattgcatcagtgtggaagtctcaggatcgttttagtttctt
ttatttgctgttcataacaattgttttcttttgtttaattcttgctttctttttttttcttctc
cgcaatttttactattatacttaatgccttaacattgtgtataacaaaaggaaatatctctgag
atacattaagtaacttaaaaaaaaactttacacagtctgcctagtacattactatttggaatat
atgtgtgcttatttgcatattcataatctccctactttattttcttttatttttaattgataca
taatcattatacatatttatgggttaaagtgtaatgttttaatatgtgtacacatattgaccaa
atcagggtaattttgcatttgtaattttaaaaaatgctttcttcttttaatatacttttttgtt
tatcttatttctaatactttccctaatctctttctttcagggcaataatgatacaatgtatcat
gcctctttgcaccattctaaagaataacagtgataatttctgggttaaggcaatagcaatatct
ctgcatataaatatttctgcatataaattgtaactgatgtaagaggtttcatattgctaatagc
agctacaatccagctaccattctgcttttattttatggttgggataaggctggattattctgag
tccaagctaggccctttgctaatcatgttcatacctcttatcttcctcccacag CTC CTG
GGC AAC GTG CTG GTC TGT GTG CTG GCC CAT CAC TTT GGC AAA GAA TTC
ACC CCA CCA GTG CAG GCT GCC TAT CAG AAA GTG GTG GCT GGT GTG GCT
AAT GCC CTG GCC CAC AAG TAT CAC TAA
```

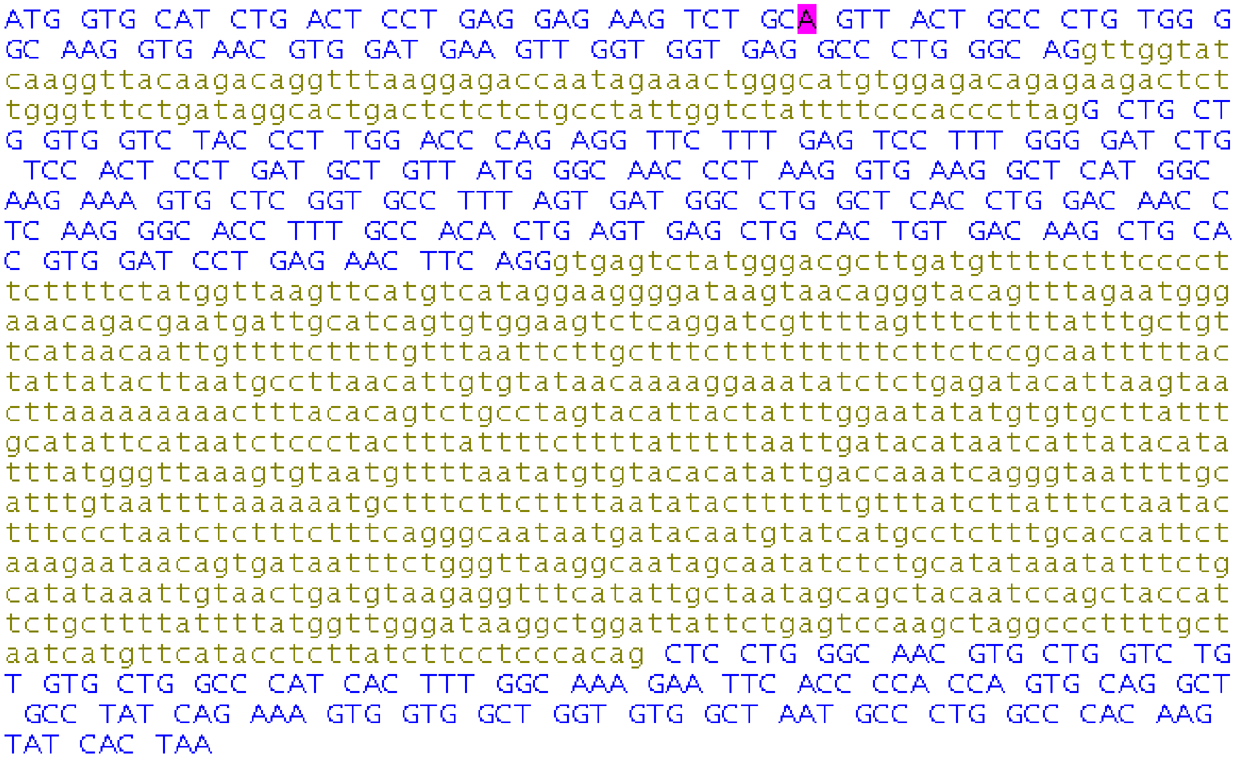
Figure 3.6: HBB wild type



```
ATG GTG CAT CTG ACT CCT G~~A~~GG AGA AGT CTG CCG TTA CTG CCC TGT GGG
 GCA AGG TGAacgtggatgaagttggtggtgaggccctgggcaggttggtatcaaggttaca
agacaggtttaaggagaccaatagaaactgggcatgtggagacagagaagactcttgggtttct
gataggcactgactctctctgcctattggtctattttcccacccttaggctgctggtggtctac
ccttggacccagaggttctttgagtcctttggggatctgtccactcctgatgctgttatgggca
accctaaggtgaaggctcatggcaagaaagtgctcggtgcctttagtgatggcctggctcacct
ggacaacctcaagggcacctttgccacactgagtgagctgcactgtgacaagctgcacgtggat
cctgagaacttcagggtgagtctatgggacgcttgatgttttctttccccttcttttctatggt
taagttcatgtcataggaaggggataagtaacagggtacagtttagaatgggaaacagacgaat
gattgcatcagtgtggaagtctcaggatcgttttagtttcttttatttgctgttcataacaatt
gttttcttttgtttaattcttgctttctttttttttcttctccgcaatttttactattatactt
aatgccttaacattgtgtataacaaaaggaaatatctctgagatacattaagtaacttaaaaaa
aaactttacacagtctgcctagtacattactatttggaatatatgtgtgcttatttgcatattc
ataatctccctactttattttcttttatttttaattgatacataatcattatacatatttatgg
gttaaagtgtaatgttttaatatgtgtacacatattgaccaaatcagggtaattttgcatttgt
aattttaaaaaatgctttcttcttttaatatactttttttgtttatcttatttctaatactttcc
ctaatctctttctttcagggcaataatgatacaatgtatcatgcctctttgcaccattctaaag
aataacagtgataatttctgggttaaggcaatagcaatatctctgcatataaatatttctgcat
ataaattgtaactgatgtaagaggtttcatattgctaatagcagctacaatccagctaccattc
tgcttttattttatggttgggataaggctggattattctgagtccaagctaggcccttttgcta
atcatgttcatacctcttatcttcctcccacagctcctgggcaacgtgctggtctgtgtgctgg
cccatcactttggcaaagaattcaccccaccagtgcaggctgcctatcagaaagtggtggctgg
tgtggctaatgccctggcccacaagtatcactaa
```

Figure 3.7: HBB Codon 6 (-A)

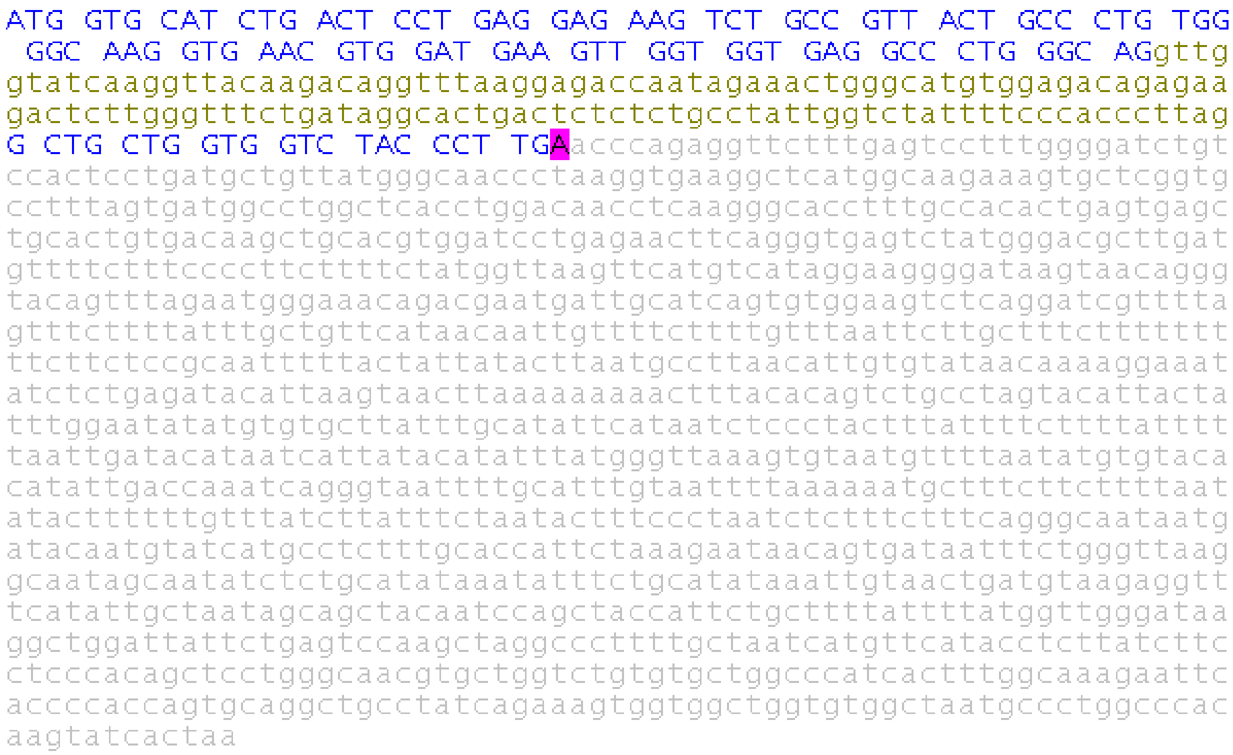
A single nucleotide change from one base to another in a gene can sometimes change the encoding of the existing polypeptide chain by one amino acid, the one encoded by the mutant codon. This is called a missense mutation. For example, the 10th codon of HBB normally encodes alanine (GCC), but a single base-pair substitution changes this to a serine codon (GCA). This mutation leads to β-thalassemia by causing a reduction in the synthesis of β-globin (Figure 3.8). This is a β^+ type mutation.



```
ATG GTG CAT CTG ACT CCT GAG GAG AAG TCT GCA GTT ACT GCC CTG TGG G
GC AAG GTG AAC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AGgttggtat
caaggttacaagacaggtttaaggagaccaatagaaactgggcatgtggagacagagaagactct
tgggtttctgataggcactgactctctctgcctattggtctattttcccacccttagG CTG CT
G GTG GTC TAC CCT TGG ACC CAG AGG TTC TTT GAG TCC TTT GGG GAT CTG
 TCC ACT CCT GAT GCT GTT ATG GGC AAC CCT AAG GTG AAG GCT CAT GGC
AAG AAA GTG CTC GGT GCC TTT AGT GAT GGC CTG GCT CAC CTG GAC AAC C
TC AAG GGC ACC TTT GCC ACA CTG AGT GAG CTG CAC TGT GAC AAG CTG CA
C GTG GAT CCT GAG AAC TTC AGGgtgagtctatgggacgcttgatgttttctttcccct
tcttttctatggttaagttcatgtcataggaaggggataagtaacagggtacagtttagaatggg
aaacagacgaatgattgcatcagtgtggaagtctcaggatcgttttagtttcttttatttgctgt
tcataacaattgttttcttttgtttaattcttgctttctttttttttcttctccgcaatttttac
tattatacttaatgccttaacattgtgtataacaaaaggaaatatctctgagatacattaagtaa
cttaaaaaaaaactttacacagtctgcctagtacattactatttggaatatatgtgtgcttattt
gcatattcataatctccctactttattttcttttatttttaattgatacataatcattatacata
tttatgggttaaagtgtaatgttttaatatgtgtacacatattgaccaaatcagggtaattttgc
atttgtaattttaaaaaatgctttcttcttttaatatacttttttgtttatcttatttctaatac
tttccctaatctctttctttcagggcaataatgatacaatgtatcatgcctctttgcaccattct
aaagaataacagtgataatttctgggttaaggcaatagcaatatctctgcatataaatatttctg
catataaattgtaactgatgtaagaggtttcatattgctaatagcagctacaatccagctaccat
tctgcttttattttatggttgggataaggctggattattctgagtccaagctaggccctttgct
aatcatgttcatacctcttatcttcctcccacag CTC CTG GGC AAC GTG CTG GTC TG
T GTG CTG GCC CAT CAC TTT GGC AAA GAA TTC ACC CCA CCA GTG CAG GCT
 GCC TAT CAG AAA GTG GTG GCT GGT GTG GCT AAT GCC CTG GCC CAC AAG
TAT CAC TAA
```

Figure 3.8: HBB Codon 10 (C>A)

A single nucleotide substitution can also lead to a premature stop codon that stops β-globin synthesis altogether. This is called a nonsense mutation because its mRNA is nonfunctional. For example, codon 37 normally encodes tryptophan (TGG), but becomes a stop codon (TGA) when the second G in the codon is mutated to an A. Similar to the frame-shift mutation discussed earlier, this nonsense mutation brings the β-globin polypeptide chain to an abrupt stop. As a result, this is a β^0 type mutation because very little β-globin protein will be produced (Figure 3.9).

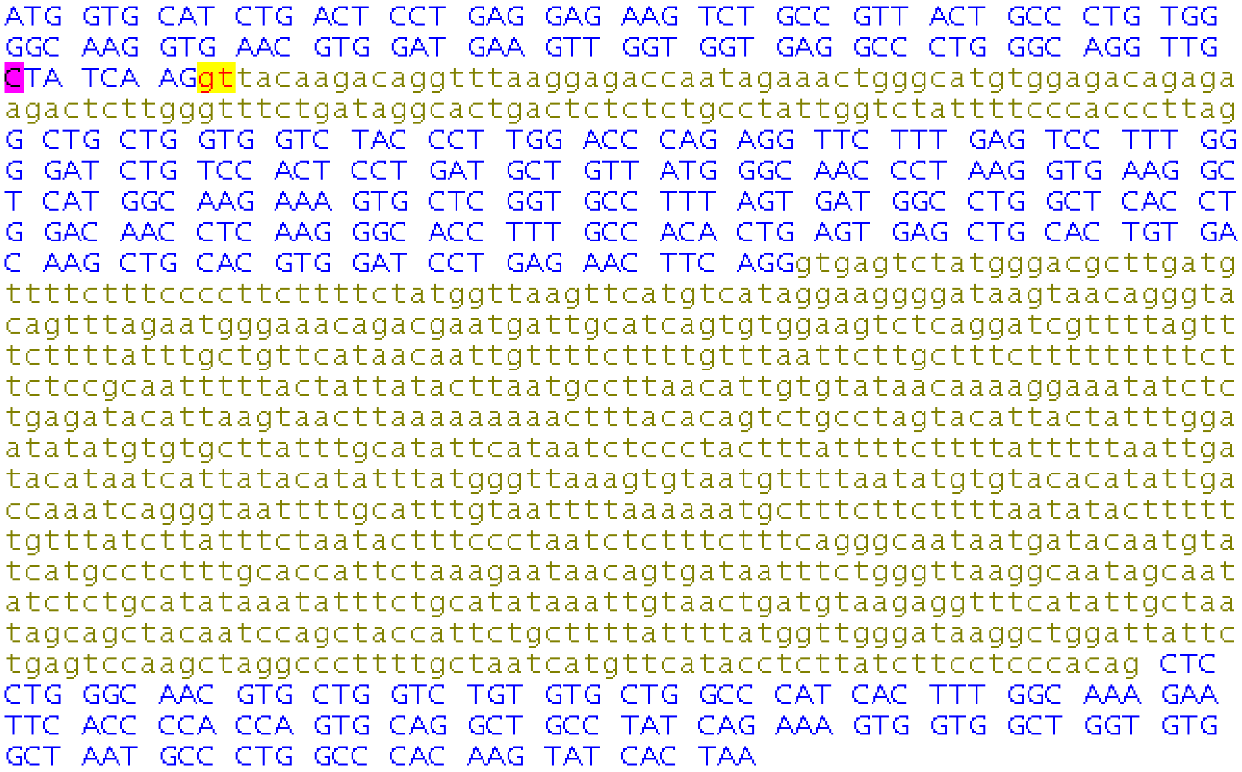


```
ATG GTG CAT CTG ACT CCT GAG GAG AAG TCT GCC GTT ACT GCC CTG TGG
 GGC AAG GTG AAC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AGgttg
gtatcaaggttacaagacaggtttaaggagaccaatagaaactgggcatgtggagacagagaa
gactcttgggtttctgataggcactgactctctctgcctattggtctattttcccacccttag
G CTG CTG GTG GTC TAC CCT TGAacccagaggttctttgagtcctttggggatctgt
ccactcctgatgctgttatgggcaaccctaaggtgaaggctcatggcaagaaagtgctcggtg
cctttagtgatggcctggctcacctggacaacctcaagggcacctttgccacactgagtgagc
tgcactgtgacaagctgcacgtggatcctgagaacttcagggtgagtctatgggacgcttgat
gttttctttccccttcttttctatggttaagttcatgtcataggaaggggataagtaacaggg
tacagtttagaatgggaaacagacgaatgattgcatcagtgtggaagtctcaggatcgtttta
gtttcttttatttgctgttcataacaattgttttcttttgtttaattcttgctttcttttttt
ttcttctccgcaatttttactattatacttaatgccttaacattgtgtataacaaaaggaaat
atctctgagatacattaagtaacttaaaaaaaaactttacacagtctgcctagtacattacta
tttggaatatatgtgtgcttatttgcatattcataatctccctactttattttcttttatttt
taattgatacataatcattatacatatttatgggttaaagtgtaatgttttaatatgtgtaca
catattgaccaaatcagggtaattttgcatttgtaattttaaaaaatgctttcttcttttaat
atacttttttgtttatcttatttctaatactttccctaatctctttctttcagggcaataatg
atacaatgtatcatgcctctttgcaccattctaaagaataacagtgataatttctgggttaag
gcaatagcaatatctctgcatataaatatttctgcatataaattgtaactgatgtaagaggtt
tcatattgctaatagcagctacaatccagctaccattctgcttttattttatggttgggataa
ggctggattattctgagtccaagctaggcccttttgctaatcatgttcatacctcttatcttc
ctcccacagctcctgggcaacgtgctggtctgtgtgctggcccatcactttggcaaagaattc
accccaccagtgcaggctgcctatcagaaagtggtggctggtgtggctaatgccctggcccac
aagtatcactaa
```

Figure 3.9: HBB Codon 37 (G>A)

Mutations do not only occur within the coding sequence. They can also be found in a gene's promoter region, enhancer region, as well as its introns. For example, any mutation in the CACC box upstream of the first exon will affect β-globin synthesis and lead to β-thalassemia because this conserved region necessitates efficient messenger RNA transcription (Forget & Hardison, 2009).

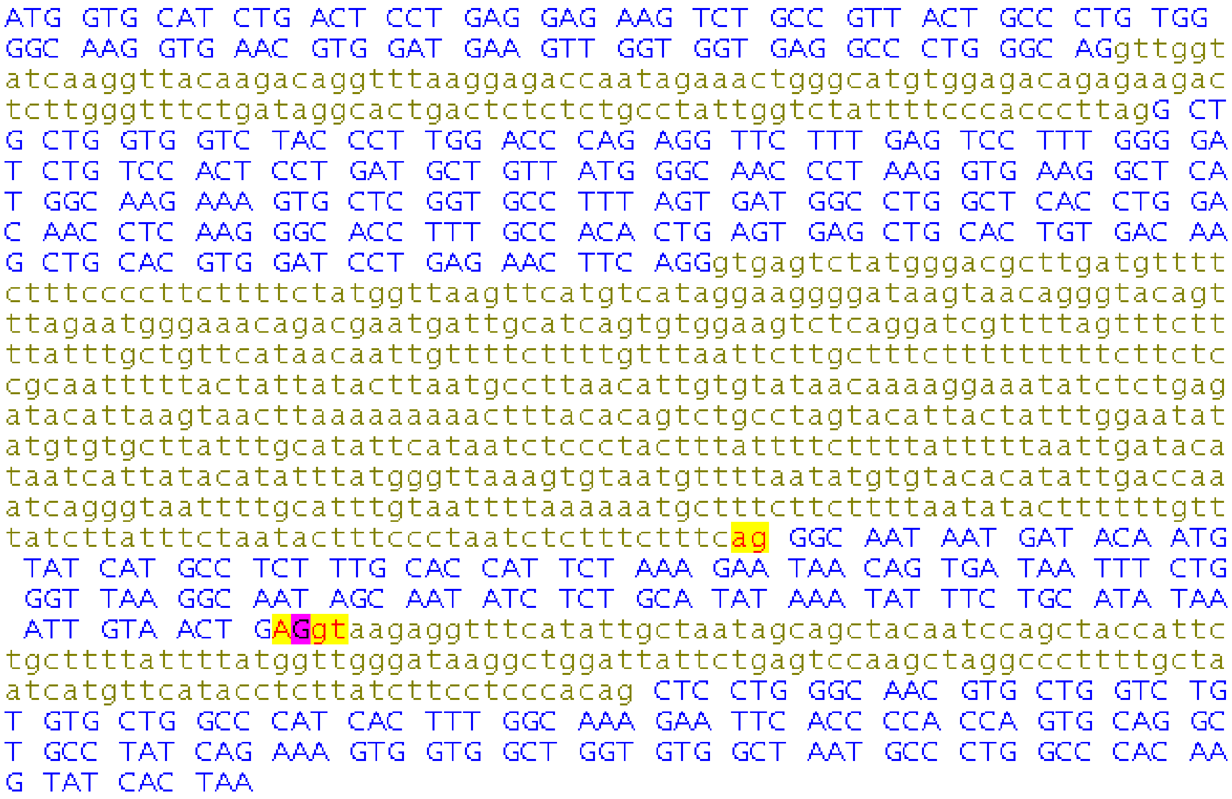
A mutation in an intron can also cause a variety of incorrect β-globin polypeptide chains because of the GT-AG rule, which requires an intron to start with a GT dinucleotide and end with an AG dinucleotide. Mutations occur at or near the GT-AG di-nucleotides which wrap up either end of an intron often activate what's called the cryptic splice sites, which cause the splicesome to splice the gene at the incorrect locations. For example, a mutation that changes the 5th base of intron 1 from G to C causes the first exon to continue past the normal 5' splice site and instead splice at a cryptic splice site a few nucleotides later (Figure 3.10). This is a β^+ type mutation.



```
ATG GTG CAT CTG ACT CCT GAG GAG AAG TCT GCC GTT ACT GCC CTG TGG
GGC AAG GTG AAC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AGG TTG
CTA TCA AGgttacaagacaggtttaaggagaccaatagaaactgggcatgtggagacagaga
agactcttgggtttctgataggcactgactctctctgcctattggtctattttcccacccttag
G CTG CTG GTG GTC TAC CCT TGG ACC CAG AGG TTC TTT GAG TCC TTT GG
G GAT CTG TCC ACT CCT GAT GCT GTT ATG GGC AAC CCT AAG GTG AAG GC
T CAT GGC AAG AAA GTG CTC GGT GCC TTT AGT GAT GGC CTG GCT CAC CT
G GAC AAC CTC AAG GGC ACC TTT GCC ACA CTG AGT GAG CTG CAC TGT GA
C AAG CTG CAC GTG GAT CCT GAG AAC TTC AGGgtgagtctatgggacgcttgatg
ttttctttccccttcttttctatggttaagttcatgtcataggaaggggataagtaacagggta
cagtttagaatgggaaacagacgaatgattgcatcagtgtggaagtctcaggatcgttttagtt
tcttttatttgctgttcataacaattgttttcttttgtttaattcttgctttcttttttttct
tctccgcaatttttactattatacttaatgccttaacattgtgtataacaaaaggaaatatctc
tgagatacattaagtaacttaaaaaaaaactttacacagtctgcctagtacattactatttgga
atatatgtgtgcttatttgcatattcataatctccctactttattttcttttatttttaattga
tacataatcattatacatatttatgggttaaagtgtaatgttttaatatgtgtacacatattga
ccaaatcagggtaattttgcatttgtaattttaaaaaatgctttcttcttttaatatactttt
tgtttatcttatttctaatactttccctaatctctttctttcagggcaataatgatacaatgta
tcatgcctctttgcaccattctaaagaataacagtgataatttctgggttaaggcaatagcaat
atctctgcatataaatatttctgcatataaattgtaactgatgtaagaggtttcatattgctaa
tagcagctacaatccagctaccattctgcttttattttatggttgggataaggctggattattc
tgagtccaagctaggcccttttgctaatcatgttcatacctcttatcttcctcccacag CTC
CTG GGC AAC GTG CTG GTC TGT GTG CTG GCC CAT CAC TTT GGC AAA GAA
TTC ACC CCA CCA GTG CAG GCT GCC TAT CAG AAA GTG GTG GCT GGT GTG
GCT AAT GCC CTG GCC CAC AAG TAT CAC TAA
```

Figure 3.10: HBB IVS-I-5 (G>C)

Sometimes, a mutation in an intron can introduce an additional exon. For example, the splicesome recognizes a 5’ cryptic splice site when the 705th base of intron 2 is changed from T to G because it is immediately followed by a T, making it a GT dinucleotide. It in turn activates a 3’ cryptic splice site more than a hundred bases upstream of this 5’ cryptic splice site where the mutation occurs. Intron 2 terminates at this 3’ cryptic splice site. An additional exon is accidently created. It starts immediately after the 3’ cryptic splice site and terminates at the cryptic 5’ splice site where the mutation occurs. Intron 3 spans from the 5’ cryptic splice site and terminates at the normal 3’ splice site where normally intron 2 ends. The resulting mRNA consists of the original exons 1, 2, 3, plus the additional exon created by the two cryptic splice sites within the normal intron 2 (Figure 3.11). This, again, has an adverse impact on the β-globin synthesis and is a β^+ type mutation (Treisman, Orkin, & Maniatis, 1983).



```
ATG GTG CAT CTG ACT CCT GAG GAG AAG TCT GCC GTT ACT GCC CTG TGG
GGC AAG GTG AAC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AGgttggt
atcaaggttacaagacaggtttaaggagaccaatagaaactgggcatgtggagacagagaagac
tcttgggtttctgataggcactgactctctctgcctattggtctattttcccacccttagG CT
G CTG GTG GTC TAC CCT TGG ACC CAG AGG TTC TTT GAG TCC TTT GGG GA
T CTG TCC ACT CCT GAT GCT GTT ATG GGC AAC CCT AAG GTG AAG GCT CA
T GGC AAG AAA GTG CTC GGT GCC TTT AGT GAT GGC CTG GCT CAC CTG GA
C AAC CTC AAG GGC ACC TTT GCC ACA CTG AGT GAG CTG CAC TGT GAC AA
G CTG CAC GTG GAT CCT GAG AAC TTC AGGgtgagtctatgggacgcttgatgtttt
ctttccccttcttttctatggttaagttcatgtcataggaaggggataagtaacagggtacagt
ttagaatgggaaacagacgaatgattgcatcagtgtggaagtctcaggatcgttttagtttctt
ttatttgctgttcataacaattgttttcttttgtttaattcttgctttctttttttttcttctc
cgcaatttttactattatacttaatgccttaacattgtgtataacaaaaggaaatatctctgag
atacattaagtaacttaaaaaaaaactttacacagtctgcctagtacattactatttggaatat
atgtgtgcttatttgcatattcataatctccctactttattttcttttatttttaattgataca
taatcattatacatatttatgggttaaagtgtaatgttttaatatgtgtacacatattgaccaa
atcagggtaattttgcatttgtaattttaaaaaatgctttcttcttttaatatacttttttgtt
tatcttatttctaatactttccctaatctctttctttcag GGC AAT AAT GAT ACA ATG
 TAT CAT GCC TCT TTG CAC CAT TCT AAA GAA TAA CAG TGA TAA TTT CTG
 GGT TAA GGC AAT AGC AAT ATC TCT GCA TAT AAA TAT TTC TGC ATA TAA
 ATT GTA ACT GAGgtaagaggtttcatattgctaatagcagctacaatccagctaccattc
tgcttttattttatggttgggataaggctggattattctgagtccaagctaggcccttttgcta
atcatgttcatacctcttatcttcctcccacag CTC CTG GGC AAC GTG CTG GTC TG
T GTG CTG GCC CAT CAC TTT GGC AAA GAA TTC ACC CCA CCA GTG CAG GC
T GCC TAT CAG AAA GTG GTG GCT GGT GTG GCT AAT GCC CTG GCC CAC AA
G TAT CAC TAA
```

Figure 3.11: HBB IVS-II-705 (T>G)

Mutations are both a blessing and a curse. They are the driving force behind evolution, by building new proteins with different exons and exon combinations. On the flip side, they cause genetic diseases, such as β-thalassemia, which plagues millions of the population because we have no cure at the moment.

Chapter 4 studies β-thalassemia in depth. Section 4.1 discusses its phenotypes, direct consequence of the mutations discussed in this chapter. Section 4.2 focuses on its epidemiology in North America, which has changed drastically in North America in the last few decades as a result of immigration.

Chapter 4: β-thalassemia

4.1: Introduction

β-thalassemia is a genetic blood disorder that causes the body to make an abnormal form of hemoglobin, the protein in red blood cells that carries oxygen, resulting in excessive destruction of red blood cells, leading to anemia (Beta Thalassemia, 2009).

From a world health point of view, β-thalassemia is the most important disorder of globin synthesis (Higgs, 2004). β-thalassemia is a fairly common genetic disease worldwide, but it occurs most frequently in people of Mediterranean, Middle Eastern, and South East Asian descent. The high mutant allele frequency in these populations is due to the fact that the β-thalassemia trait provides a selective advantage in the form of protection against malaria (Knight, 2009). The World Health Organization estimated that about 1.5% of the world population might be carriers of β-thalassemia and that around 60,000 severely affected infants are born each year (Higgs, Engel, & Stamatoyannopoulos, 2011).

β-thalassemia is caused by mutations in the β-globin gene on the short arm of chromosome 11. There are at present approximately two hundred documented β-thalassemia mutations. These mutations involve nucleotide modification in the hemoglobin β gene (HBB) which builds inappropriate β-globins sub-units in hemoglobin and causes devastating effects to the body.

An excess of α chains in precursor red blood cells causes defective red blood cell formation called dyserythropoiesis, which causes mature red blood cells membrane damage and haemolysis, the disintegration of red blood cells. These primary changes in erythropoisis, the formation of mature red blood cells, lead to all the secondary changes such as anemia, splenomegaly (enlargement of spleen), marrow expansion, bone deformities, hyper-metabolic state, and iron accumulation (Higgs, Engel, & Stamatoyannopoulos, Thalassemia, 2011).

Despite advanced research such as pharmacological agents, stem-cell transplantation, and gene therapy over the years, β-thalassemia still represents a major world-wide health problem and is still among the most common monogenic diseases affecting man (Higgs, 2004). For instance, results from therapeutic interventions in clinical trials for β-thalassemia by increasing expression of γ-globin or decreasing expression of α-globin to restore the balance between the α-globin and β-globin chains have not been sufficiently encouraging to develop large scale trials (Higgs, Engel, & Stamatoyannopoulos, Thalassemia, 2011). At the moment, there simply is not an effective cure for β-thalassemia. β-thalassemia patients are dependent on life-long blood transfusion, which supplies them with normal hemoglobin. Unfortunately, iron accumulation is a severe side effect of regular blood transfusion. Patients of blood transfusion also require regular iron chelation to remove excess iron from their blood.

Section 4.2 discusses the three phenotypes of β-thalassemia. Section 4.3 discusses disease's epidemiology in North America. This chapter concludes the studies of β-thalassemia and the mutations that cause it. The remaining of this thesis focuses on the design and implementation part of this project.

4.2: Phenotypes of β-thalassemia

Phenotype describes an observable characteristic or trait, contrast to genotype, which describes the genetic alleles of an individual that give him his observable phenotype. A phenotype ranges from appearance to structural, biochemical, physiological, or behavioral character. A phenotype is often considered as being the product of the genetic constitution of an individual and the environmental factors (Knight, 2009).

There are three broad clinical phenotypes observed in individuals with β-thalassemia: minor, intermedia, and major. They are categorized by the severity of the disease based on the expressivity of their HBB alleles, or the amount of β-globins synthesized in an individual (Knight, 2009).

Individuals with β-thalassemia minor carry one normal HBB allele and one mutant HBB allele of either β^0 or β^+ type. They carry the trait of β-thalassemia but do not always need active treatment. They are frequently monitored because, while β-thalassemia minor is not life-threatening on its own, there can be consequences of the mutation. These individuals suffer from mild to moderate anemia from red blood cell destruction by the spleen due to ineffective erythropoiesis, which can release additional iron into the bloodstream to cause iron overload (Knight, 2009).

Individuals with β-thalassemia intermedia carry two mutant HBB alleles of β^+ type. They have a significant dyserthropoietic anemia as their spleen removes their immature red blood cells that are in excess of α-globin chains. They suffer from anemia of a wide range of severities, and increased susceptibility to infection, iron overload, bone deformities and fractures, and enlargement of the spleen (Beta Thalassemia, 2009).

Individuals with β-thalassemia major also have two mutant HBB alleles, but one of them is of the β^0 type. β-thalassemia major manifests around six months of age in infants, causing an increased rate of susceptibility to infection and failure to thrive. Patients are dependent on blood transfusions but also rely on chelation therapy to remove excess iron from transfusion. Bone marrow transplantation may be curative but is infrequently available (Knight, 2009). Untreated β-thalassemia major eventually leads to death.

Section 4.3 discusses the epidemiology of β-thalassemia in North America.

4.3: Epidemiology of β-thalassemia in North America

Traditionally, β-thalassemia is common in the Mediterranean with carrier frequencies of up to 20% in some areas. It is also common in North Africa, the Middle East, India and Southeast Asia. The high allele frequency is associated with a selective advantage seen with protection from malarial infection (Knight, 2009).

The combination of increased number of Asian immigrants and improved treatment has altered the clinical spectrum of thalassemia in North America. Many genotypes that were previously considered to be uncommon in North America are increasingly diagnosed in many screening programs because of the dramatic shifts of affected populations to North America (Vichinsky, MacKlin, Waye, Lorey, & Olivieri, 2005).

The Thalassemia Clinical Research Network (TCRN) of the National Heart, Lung, and Blood Institute conducted a study that examined the demography and natural history of patients with thalassemia registered in the five largest treatment centers in North America. The study revealed a gradual but visibly significant change of the ethnic makeup of the β-thalassemia patients in these treatment centers, mirroring a trend of the ethnicity makeup of US and Canadian immigrants over the last 100 year. For instance, in the 1970s, 70% of newborns with β-thalassemia major in California were white, in constrast to 16% of the patients in the patients in the last decade. β-thalassemia historically affected the Caucasian population in North America, but now 60% of the children affected are from other ethnic backgrounds, illustrating the effects of immigration on the number and ethnicity of thalassemia births in North America (Vichinsky, MacKlin, Waye, Lorey, & Olivieri, 2005).

The rest of this thesis discusses the design and implementation of a database-driven website that illustrates the changes of the HBB sequence when it is affected by each known β-thalassemia mutation. Chapter 5 propounds the relational database management system (RDBMS) theory and presents the database design for this project. Chapter 6 examines a couple libraries of the PERL scripting language and details the website implementation.

Chapter 5: Database

5.1: Introduction

Database is a shared, integrated computer structure that stores a collection of data and metadata, or data about data (Coronel, 2011). The metadata describes the data, including their characteristics and the relationships that link them together. For instance, the metadata stores the name of a table, the names of each field in this table, the data type such as number or text of these fields, the constraints such as primary key and foreign key of these fields, and the access log of this table. The metadata helps the database make sense of its data.

Database design focuses on how the database structure will be used to store and mange data. The first step in designing a database is called data modeling. Data modeling is a process of creating a specific data model for a determined problem domain (Coronel, 2011). A data model is usually a relatively simple graphical representation of a complex real-world data structure. In the case of a relational database, which will be explained shortly, its data model is usually called the entity-relationship diagram (ERD).

All good database management systems must guarantee the integrity of transaction processing by meeting the ACID test (Gray, 1981). "A" stands for "atomicity," the all-or-nothing execution of transitions. "C" stands for "consistency," which means that all databases have consistency constraints, or expectations about the relationships among data elements. For instance, an account balances may not be negative after a transaction finishes. The transactions are expected to preserve the consistency of the database. "I" stands for "isolation," the fact that each transaction must appear to be executed as if no other transaction is executing at the same time. "D" stands for "durability," the condition that the effect on the database of a transaction never be lost once the transaction has completed, regardless of the circumstance (Garcia-Molina, 2009).

There are several data models in database design. This project uses the relational model. The relational model was introduced in 1970 by E.F. Codd of IBM (Codd, 1970). The relational data model is implemented through a sophisticated relational database management system (RDBMS). There are several commercial and open-source relational database management systems, but not every database passes the ACID test.

5.2: PostgreSQL

PostgreSQL is a powerful, open source object-relational database system. It runs on all major operation systems. It is fully ACID (atomicity, consistency, isolation, and durability) compliant, and has full support for foreign keys, joins, views, triggers, and stored procedures in multiple languages (PostgreSQL, 2012).

PostgreSQL has had a long history as a research project at the University of California at Berkeley (UCB). It was originally called Ingres in the late seventies and was meant to be an exercise in creating a database system according to the classic RDBMS (relational database management system) theory. A decade later, UCB Professor Michael Stonebraker started a follow-up project on Ingres called Postgres, playing off of its predecessor's name. Postgres broke new ground in database concepts with its exploration of the "object relational" technologies. Professor Stonebraker and his graduate students worked on Postgres for eight years before it was bought by Informix, which was purchased by IBM in 2001 (PostgreSQL, 2012).

However, in 1995, two Ph.D. students from Professor Stonebraker's lab, Andrew Yu and Jolly Chen, replaced Postgres' POSTQUEL query language with an extended subset of Structured Query Language (SQL) and renamed the system to Postgres95, which became radically transformed in the next eight years by a group of global developers outside of Berkeley and came to be known as PostgreSQL (PostgreSQL, 2012).

PostgreSQL has received numerous awards since 1999, including many years as the editor's choice of best database (Linux Journal, 2001). It is regularly chosen as the best choice of database in the open source community. Numerous commercial companies, education institutions, and government organizations use PostgreSQL, including the U.S. state department (PostgreSQL, 2012).

5.3: Schema

Database schema is a group of database objects such as tables and indices (Garcia-Molina, 2009). The schema for this project contains several tables, indexes, and functions. Functions format and load data into tables, which are used for storage and retrieval purposes. Each table is indexed with its primary key, discussed in the next paragraph, for data integrity and performance.

There are eight tables used in this project, six of which are "related" by primary keys and foreign keys (Figure 5.1). A primary key is one or more fields in a table that uniquely identifies each row in the table. A foreign key is a field in a table whose values match the primary key values in a different table (Coronel, 2011). Two tables are related if one table's foreign key references the other's primary key.

A relationship formed by the primary key of a table and the foreign key of another table can be one-to-one or one-to-many. In a one-to-one relationship, the primary key value of a table can appear at most once as the foreign key value in a different table. In a one-to-many relationship, the primary key value of a table can appear more than once as the foreign key value in a different table.

The RAW table contains the wild type HBB sequence. The HBB_WILD table also contains the wild type HBB sequence, but each nucleotide and its position in the sequence is stored in a separate row. There are 1843 rows in HBB_WILD, including some upstream and downstream nucleotides. The RAW and HBB_WILD tables are independent of the rest of the tables in the schema because they are tables for development, not production.

The MUTATION table contains identity information of the 212 β-thalassemia mutations, obtained from the HbVar Database at Pennsylvania State University. The MUTATION table drives the rest of the other tables.

The ETHNICITY table contains information pertaining to the ethnicity distribution of each mutation kept in the MUTATION table. It is necessary to use an additional table to store the ethnicity information because the relationship between mutations and their ethnicity distributions is that of one-to-many instead of one-to-one, making it impossible to store all the data in the same table without redundancy.

The COMMENT table contains sequence analysis information that explains the effect of each mutation.

The REFERENCE table stores information of the relevant published literature except the author names, which are kept in the AUTHOR table. The information of publications and authors are kept in separate tables because they, like the mutations and their ethnicity distributions, have a one-to-many relationship when papers are written by multiple authors.

The HBB_MUTE table has a similar structure as that of HBB_WILD. However, HBB_MUTE has approximately 372,385 rows, because HBB_MUTE keeps a set of the mutant HBB sequence for each of the 212 mutations, each of which generates approximately 1843 or more rows.

Raw data files of β-thalassemia mutations, their ethnicity distributions, comment on selected mutations, and published references for each documented mutation were provided by Ms. Belinda Giardine of the HbVar database at Pennsylvania State University. Raw data files were organized and formatted before being copied into the MUTATION, ETHNICITY, COMMENT, REFERENCE, and AUTHOR tables with the “copy” command.

The function load_hbb_wild() loads each base of the HBB sequence into a row from the RAW table to the HBB_WILD table. Each nucleotide in the HBB_WILD is identified by its position in the gene and the coding sequence. The function load_hbb_mute() loads a set of the HBB sequence for each mutation kept at the MUTATION table into HBB_MUTE, in the same format as has been done by load_hbb_wild() for the HBB_WILD table. As a result, there are approximately 372,385 rows in the HBB_MUTE table, for each mutation constitutes a set of 1843 or more rows.

There is an identifier for each set of rows that are part of the same mutation in the HBB_MUTE. Each set of HBB sequence in the HBB_MUTE table is exactly the same as that of the HBB_MUTE, except the mutant bases. In the case of a substitution, the mutant base for a given mutation is changed and noted in the row of the substituted nucleotide by the functions single_sub() and post_cds_single_sub(). In the case of an insertion, a new row for each base is added and the position of each new base is noted by the function cds_ins(). In the case of a deletion, rows of deleted bases are noted by the functions single_del() and multi_del(). In the case of mutations involving both deletion and insertion, functions single_delins() and multi_de_ins() inserts a new row for each new base and annotate each deleted row.

However, there is more to modifying the HBB_MUTE table in order to reflect the state of an individual HBB sequence than annotating each mutant base. Each mutation has a different effect on the wild type HBB sequence. Frame shift mutations tend to bring about a premature stop codon in the sequence. A premature stop codon as well as the rest of the sequence is noted. Likewise with nonsense mutations, both the bases of the premature stop codon and all the bases of the rest of the sequence are similarly annotated.

Single base substitutions are the most complex type of the β-thalassemia mutations. Sometimes they abolish splicing completely, sometimes they block it partially, and sometimes they activate what's called the cryptic splice sites and lead to erroneous splicing which leads to incorrect coding sequence. Finding cryptic splice sites for a mutation and annotate them in the HBB_MUTE table accordingly requires perusal of published literature covering the mutation. Modification of data in the HBB_MUTE table to reflect the state of the HBB sequence based on individual mutation is completely done manually one by one.

Chapter 6 discusses website implementation of this project. It goes over several topics including the server-side scripting language PERL, its modules that are used for this project, the cascading style sheet that handles the presentation of the web pages, and instructions on using the website.

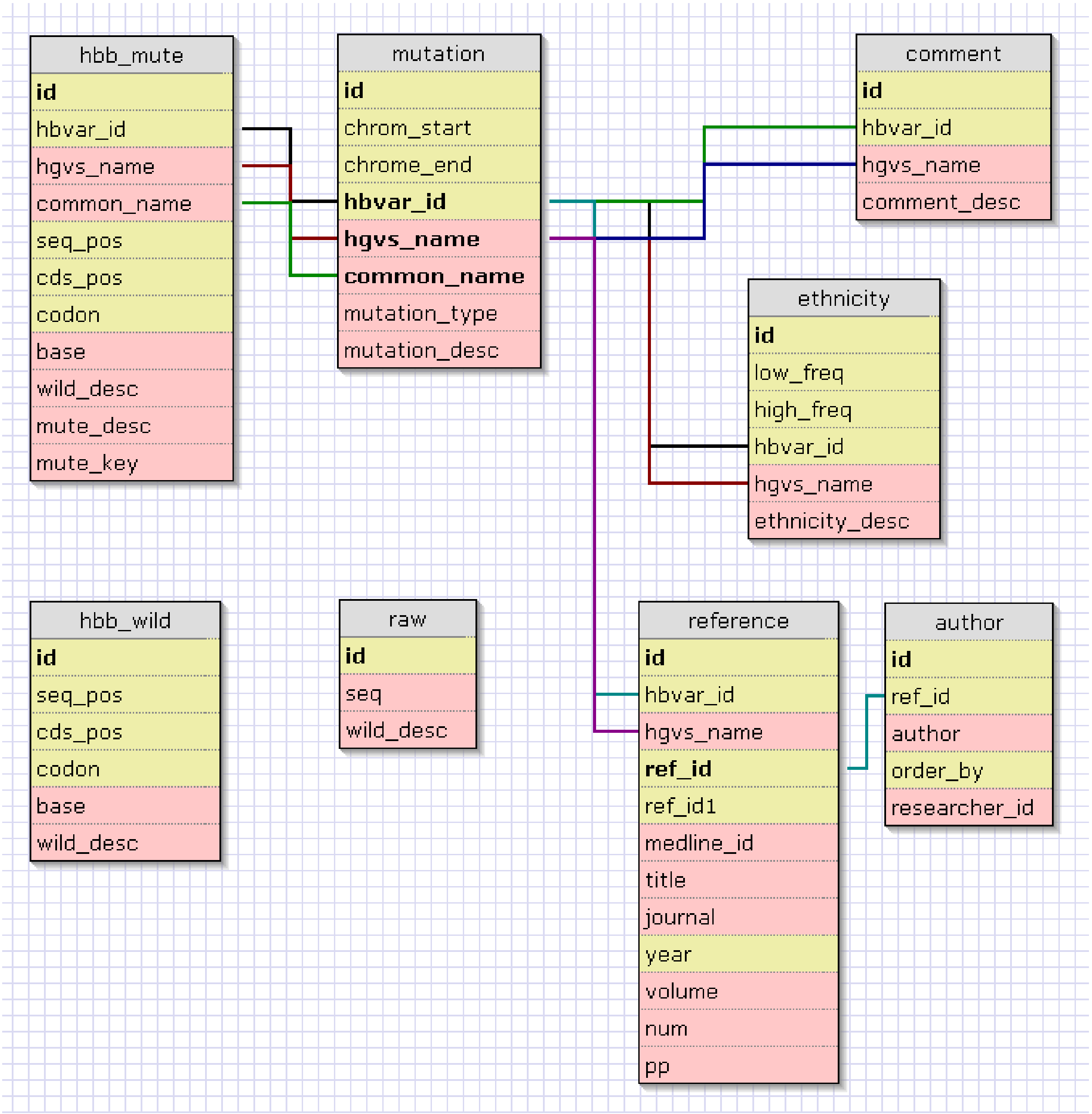


Figure 5.1: Database schema

Chapter 6: Web Server-Side Implementation

6.1: Introduction

All websites need to be hosted by a web server. My choice for this project is the open-source web server Apache. I run it on Windows.

A database-driven website can be implemented with a variety of server-side scripting languages, all of which ultimately generates HTML (Hypertext Markup Language) for display on a browser. My choice of language for the web page implementation is PERL (Practical Extraction and Reporting Language) and CSS (Cascade Style Sheet). Both are open-source and cross-platform languages.

PERL is an interpreted scripting language originally written by Larry Wall. It is an ideal tool of text manipulation (About PERL, 2012). As an interpreted language, it does not require compilation and therefore allows a rapid development cycle. PERL uses dynamic typing which allows the program to figure the best data type for the variable value at run time. PERL is especially made powerful by its vast library of modules in CPAN (Comprehensive PERL Archive Network). PERL has been used extensively in the field of bioinformatics. The first issue of The PERL Journal has an article titled “How PERL saved the Human Genome Project” (Richardson, 1999).

CSS is a style sheet language used to describe the look and feel of web pages. It allows the separation of content and presentation of markup languages such as HTML. CSS standards are maintained by W3C (World Wide Web Consortium).

Section 6.2 discusses the PERL module CGI, which is used for web page content. Section 6.3 discusses the PERL module DBI, which interfaces with the PosgreSQL database. Section 6.4 discusses CSS 3, the current CSS standard.

6.2: Website overview

The final website contains three main sections on the main page.

The first section includes the title of the page and a web form. The web form allows users to pick from a drop-down list a hemoglobin β gene (HBB) mutation and retrieves its sequence for display. PERL DBI’s statement handle queries the PostgreSQL database for a list of HBB mutations ordered by common name and the result is stored in a PERL hash. The PERL

CGI object then populates the drop-down menu from this hash. The "Display only CDS" checkbox is a form parameter that is either set or not. A submit button allows user to send the request for a specific mutated HBB sequence based on the choice from the drop-down menu and the parameter that specifies whether to display the coding sequence only.

The second section is the mutated sequence of the HBB mutation selected from the drop-down list above. While the rest of the web page is generally displayed in the Georgia font, the sequence is shown in Lucida Console, a mono-space font, so that each nucleotide lines up for a better presentation. Each codon in the sequence is also separated by a space and an HTML tooltip shows its number in the CDS when the mouse hovers over it. The codon number tooltip eliminates the mundane process of manually locating codons by number. Each segment in the sequence is marked by letter-case, highlights, colors, and CSS style. CDS is upper-case, whereas the rest of the sequence is in lower-case. The mutated nucleotides have red highlights and cryptic splice sites have yellow. UTR nucleotides are shown in purple letters, CDS nucleotides are shown in blue letters, and IVS nucleotides are shown in green letters. Any deleted nucleotide has a strikethrough. The mutated HBB sequence hides the UTR and IVS sections if the "Display only CDS" checkbox is checked.

The third section contains information pertaining to the mutated HBB sequence shown above. It includes comment, ethnicity distribution, and reference, each is displayed in an HTML table. All the data in this section come with permission n from the HbVar database at Pennsylvania State University (PSU). Comment is usually notes entered by the biologists or curators at the PSU to give further description of the mutation, such as its cause and effect. Ethnicity distribution clearly outlines the demographic data of known β-thalassemia patients associated with the particular mutations. Reference contains the information of published literatures that directly discuss the mutation. A disclaimer at the bottom of the web page attributes the source of information and gives credit to HbVar database at PSU.

6.3: PERL CGI

The PERL CGI (Common Gateway Interface) module generates the web content of this project. CGI.pm is Perl's widely deployed solution for processing and preparing HTTP (Hypertext Transfer Protocol) requests and responses. It handles form submissions, file uploads, reading and writing cookies, query string generation and manipulation, and processing and preparing HTTP headers (Perl CGI, 2012).

There are two styles of programming withCGI.pm: function-oriented style and object-oriented approach. Function-oriented programming uses subroutines instead of objects to accomplish tasks, such as retrieving CGI parameters and creating HTML tags. Object-oriented programming, in contrast, uses the CGI objects to create elements of a page. Each CGI object has a list of named parameters passed to it by server when the CGI script starts (Perl CGI, 2012). Each CGI object is modifiable because it corresponds to the "state" of the CGI script (Perl CGI, 2012). This project uses the object-oriented approach to implement the web pages.

$q = CGI->new; creates a new CGI object named $q. $q is then used to create HTTP header with $q->header, start the HTML with $q->start_html, and end the HTML $q->end_html.

$q->param('mutation'); returns the parameter named mutation, which is the unique HbVar ID from the HbVar database. This parameter drives the content of the main web page, which loads the correct mutant HBB sequence based on its value.

6.4: PERL DBI

The PERL DBI (Database Independent Interface) module interfaces with the databases. DBI.pm enables the website generated by CGI.pm to transparently access database – PostgreSQL in the case of this project (Perl DBI, 2012). It supports database connection and data manipulation through SQL (structured query language). DBI.pm defines three main types of objects called handles used to interact with databases: driver handle, database handle, and statement handle.

The driver handle represents the loaded driver and is created when the driver is loaded and initialized by DBI.pm. A driver handle is usually not referenced within a CGI script because the actual instantiation of the driver handle should normally happen "under the hood" of DBI when DBI->connect() is called (Programming the Perl DBI, 2012).

The database handle is child of its corresponding driver handle, the product of calling DBI->connect(). A database handle is usually referred to as $dbh (Programming the Perl DBI, 2012).

The statement handle is child of its corresponding database handle. The statement handle is the type of object that DBI defines for database interaction and manipulation. It encapsulates individual SQL statements to be executed within the database. Data within one statement is protected from tampering or modification by other statement handles. A statement handle is generally referred to as $sth.

The technique of using a combination of Perl CGI and DBI modules ensures a simplistic backend website development process.

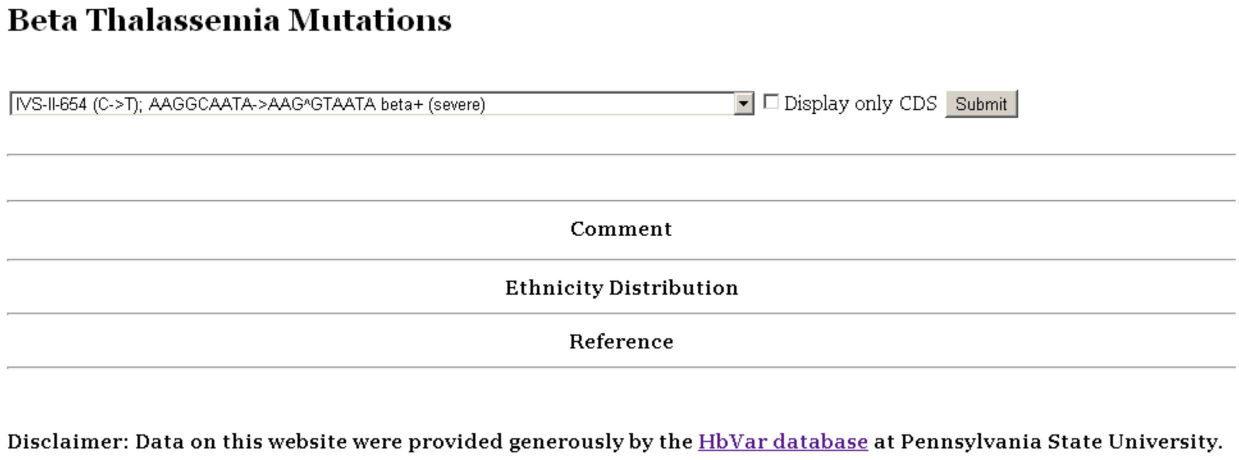
6.5: CSS 3

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics of a document written in a markup language (Cascading Style Sheets, 2012). It has become a popular tool in web design in the past few years because it helps separate the web content from its presentation, enabling web designers to focus on both independently. The cascading in CSS refers to a priority scheme that determines which style rules apply if more than one rule matches against a particular element, so the results are predictable (Cascading Style Sheets, 2012).

This project references an external CSS file, apart from the CGI script that generates the HTML web pages, to further separate the presentation from content. The CSS file defines the layout, colors, and fonts used in each different tag in the HTML page. It allows multiple tags to share the same formatting and reduces repetition in the structural content. For instance, CSS sets the maximum page width; it defines the font information and alignment of each segment shown on the web page; and it selectively highlight, underline, and strikethrough key nucleotides in the HBB sequence.

6.6: Query Instructions

The drop-down menu at the top of the website contains a list of all the β-thalassemia mutations in alphabetical order. Select the mutation of interest from the drop-down menu shown in Figure 6.1. Mark the “Display only CDS” checkbox next to the drop-down menu, if you are only interested in viewing the mutated coding sequence. Click the Submit button (Figure 6.1).



Beta Thalassemia Mutations

IVS-II-654 (C->T); AAGGCAATA->AAG^GTAATA beta+ (severe)

Comment

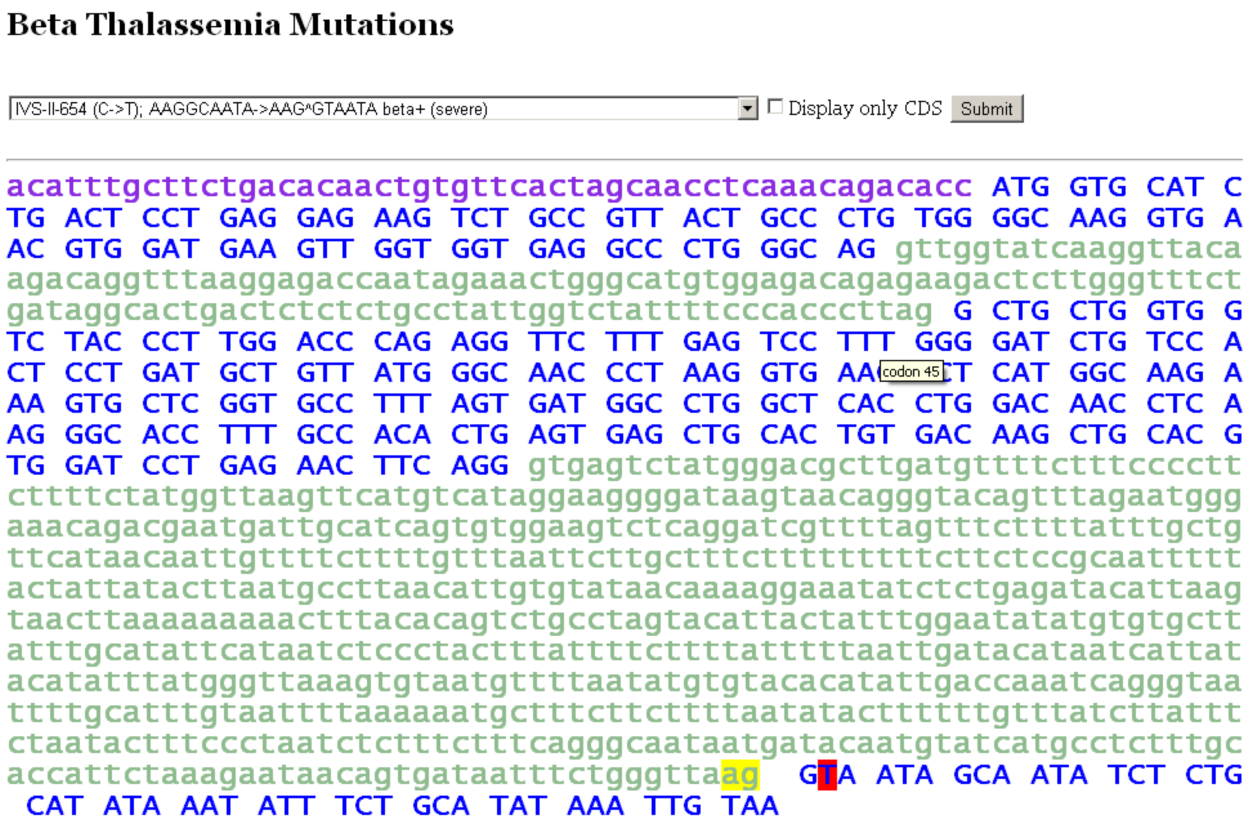
Ethnicity Distribution

Reference

Disclaimer: Data on this website were provided generously by the HbVar database at Pennsylvania State University.

Figure 6.1: Querying a mutation

A mutated HBB sequence is shown in Figure 6.2 for the IVS-II-654 (C > T) mutation, showing both the coding sequence and the non-coding regions. Lowercase letters in purple denotes the un-translated region. Uppercase letters in blue denotes the coding sequence (CDS). Groups of three nucleotides in the CDS are separated from one another to signify individual codon. Hover the mouse over each codon and its position will display in a bubble tooltip (Figure 6.2). Lowercase letters in green denotes the intervening sequence (IVS). Letters highlighted in red is the mutated nucleotide. Letters highlighted in yellow are the cryptic splice sites the mutation inactivates.



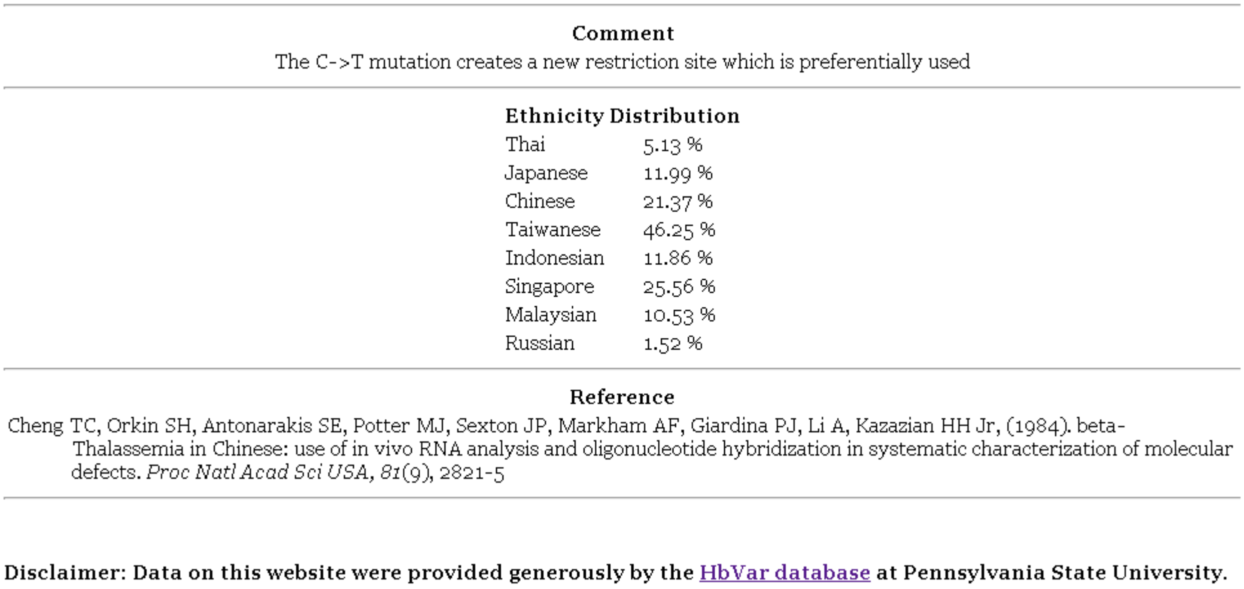
Beta Thalassemia Mutations

IVS-II-654 (C->T); AAGGCAATA->AAG^GTAATA beta+ (severe)

```
acatttgcttctgacacaactgtgttcactagcaacctcaaacagacacc ATG GTG CAT C
TG ACT CCT GAG GAG AAG TCT GCC GTT ACT GCC CTG TGG GGC AAG GTG A
AC GTG GAT GAA GTT GGT GGT GAG GCC CTG GGC AG gttggtatcaaggttaca
agacaggtttaaggagaccaatagaaactgggcatgtggagacagagaagactcttgggtttct
gataggcactgactctctctgcctattggtctattttcccacccttag G CTG CTG GTG G
TC TAC CCT TGG ACC CAG AGG TTC TTT GAG TCC TTT GGG GAT CTG TCC A
CT CCT GAT GCT GTT ATG GGC AAC CCT AAG GTG AA codon 45 T CAT GGC AAG A
AA GTG CTC GGT GCC TTT AGT GAT GGC CTG GCT CAC CTG GAC AAC CTC A
AG GGC ACC TTT GCC ACA CTG AGT GAG CTG CAC TGT GAC AAG CTG CAC G
TG GAT CCT GAG AAC TTC AGG gtgagtctatgggacgcttgatgttttctttcccctt
cttttctatggttaagttcatgtcataggaaggggataagtaacagggtacagtttagaatggg
aaacagacgaatgattgcatcagtgtggaagtctcaggatcgttttagtttcttttatttgctg
ttcataacaattgttttcttttgtttaattcttgctttctttttttttcttctccgcaatttt
actattatacttaatgccttaacattgtgtataacaaaaggaaatatctctgagatacattaag
taacttaaaaaaaaactttacacagtctgcctagtacattactatttggaatatatgtgtgctt
atttgcatattcataatctccctactttattttcttttatttttaattgatacataatcattat
acatatttatgggttaaagtgtaatgttttaatatgtgtacacatattgaccaaatcagggtaa
ttttgcatttgtaattttaaaaaatgctttcttcttttaatatactttttttgtttatcttattt
ctaatactttccctaatctctttctttcagggcaataatgatacaatgtatcatgcctctttgc
accattctaaagaataacagtgataatttctgggttaag  GTA ATA GCA ATA TCT CTG
 CAT ATA AAT ATT TCT GCA TAT AAA TTG TAA
```

Figure 6.2: Query result

Below the sequence three sections are displayed: comment, ethnicity distribution, and reference (Figure 6.3). The comments are notes of the HbVar database curators at the Pennsylvania State University. It gives higher-level detail of the effect by the mutation. For example, it often explains how the mutation activates cryptic splice sites and describes the amount of mRNA produced as a result. The ethnicity distribution describes the percentage of patients in each population who have this particular mutation. The percentage associated with each population means the portion of that population's β-thalassemia patients who have this mutation; it does not mean the portion of all the patients affected by this mutation belong to certain ethnic groups. This is why the percentages do not add up to one hundred. The reference contains published literatures pertaining to the particular mutation. All the data from the comment, ethnicity distribution, and reference sections come from HbVar database. A disclaimer gives credit to the HbVar database is shown at the bottom of the page.



Comment

The C->T mutation creates a new restriction site which is preferentially used

Ethnicity Distribution

Thai	5.13 %
Japanese	11.99 %
Chinese	21.37 %
Taiwanese	46.25 %
Indonesian	11.86 %
Singapore	25.56 %
Malaysian	10.53 %
Russian	1.52 %

Reference

Cheng TC, Orkin SH, Antonarakis SE, Potter MJ, Sexton JP, Markham AF, Giardina PJ, Li A, Kazazian HH Jr, (1984). beta-Thalassemia in Chinese: use of in vivo RNA analysis and oligonucleotide hybridization in systematic characterization of molecular defects. *Proc Natl Acad Sci USA, 81*(9), 2821-5

Disclaimer: Data on this website were provided generously by the HbVar database at Pennsylvania State University.

Figure 6.3: Comment, ethnicity distribution, and reference

Chapter 7 concludes this thesis with a summary and maintenance plans.

Chapter 7: Conclusion

7.1: Summary

This project has been a great opportunity of learning the mutations that lead to β-thalassemia, which is considered a well-studied and important genetic disorder due to immigration to the US in the past decades. β-thalassemia is primarily caused by single-point mutations that produce a structurally unfit amino acid, or prematurely terminates the HBB sequence, or activates cryptic splice sites that cause incorrect splicing of the gene. Many of these mutations are deceivingly simple nucleotide changes, but their effects are devastating and, after decades of research, we still don't have an effect cure on a large scale.

While there are many published literatures that discuss individual β-thalassemia mutations, no one has compiled all these mutations in one place and show how the entire HBB sequence is affected as by each mutation. This is precisely the goal of my project. I wanted to create an educational website that people who are new to bioinformatics or any discipline can learn how β-thalassemia mutations cause the HBB sequence to change and produce a limited amount of normal messenger RNA as a result.

This website is free to the public. It has a very minimal learning curve so virtually anyone can learn to navigate on the spot. It runs on any platform with maximum efficiency and minimum amount of maintenance, thanks to indexed tables in the database and server-side scripting, which dynamically renders each page without plugin and browser scripting support. The mutated HBB sequence that displays after each query is run is solely my work based on literature research. The comment, ethnicity distribution, and reference data come from the distinguished HbVar database.

7.2: Future direction

Data in the database should sync with HbVar once a year as self-validation. This can be accomplished with a script that does string comparison by parsing the source code of each β-thalassemia mutation page on the HbVar website and that of the β-thalassemia mutation website. It is recommended that permission from the HbVar database be obtained before attempting to parse their source code of their web pages. It may be the case that HbVar would prefer to provide dump files as they have kindly done for my project so far.

Appendix A: Annotated Sequence of the HBB Gene

```
Promoter Region

      tgtggagcca caccctaggg ttggccaatc tactcccagg agcagggagg
      gcaggagcca gggctgggca taaaagtcag ggcagagcca tctattgctt

Exon 1

1     ACATTTGCTT CTGACACAAC TGTGTTCACT AGCAACCTCA AACAGACACC
51    ATGGTGCATC TGACTCCTGA GGAGAAGTCT GCCGTTACTG CCCTGTGGGG
101   CAAGGTGAAC GTGGATGAAG TTGGTGGTGA GGCCCTGGGC AG

Intron 1

      GTTGGTATCA AGGTTACAAG ACAGGTTTAA GGAGACCAAT AGAAACTGGG
      CATGTGGAGA CAGAGAAGAC TCTTGGGTTT CTGATAGGCA CTGACTCTCT
      CTGCCTATTG GTCTATTTTC CCACCCTTAG

Exon 2

143                                                GCTGCTGG
151   TGGTCTACCC TTGGACCCAG AGGTTCTTTG AGTCCTTTGG GGATCTGTCC
201   ACTCCTGATG CTGTTATGGG CAACCCTAAG GTGAAGGCTC ATGGCAAGAA
251   AGTGCTCGGT GCCTTTAGTG ATGGCCTGGC TCACCTGGAC AACCTCAAGG
300   GCACCTTTGC CACACTGAGT GAGCTGCACT GTGACAAGCT GCACGTGGAT
351   CCTGAGAACT TCAGG

Intron 2

      GTGAGTCTAT GGGACGCTTG ATGTTTTCTT TCCCCTTCTT TTCTATGGTT
      AAGTTCATGT CATAGGAAGG GGATAAGTAA CAGGGTACAG TTTAGAATGG
      GAAACAGACG AATGATTGCA TCAGTGTGGA AGTCTCAGGA TCGTTTTAGT
      TTCTTTTATT TGCTGTTCAT AACAATTGTT TTCTTTTGTT TAATTCTTGC
      TTTCTTTTTT TTTCTTCTCC GCAATTTTTA CTATTATACT TAATGCCTTA
      ACATTGTGTA TAACAAAAGG AAATATCTCT GAGATACATT AAGTAACTTA
      AAAAAAAACT TTACACAGTC TGCCTAGTAC ATTACTATTT GGAATATATG
      TGTGCTTATT TGCATATTCA TAATCTCCCT ACTTTATTTT CTTTTATTTT
      TAATTGATAC ATAATCATTA TACATATTTA TGGGTTAAAG TGTAATGTTT
      TAATATGTGT ACACATATTG ACCAAATCAG GGTAATTTTG CATTTGTAAT
      TTTAAAAAAT GCTTTCTTCT TTTAATATAC TTTTTTGTTT ATCTTATTTC
      TAATACTTTC CCTAATCTCT TTCTTTCAGG GCAATAATGA TACAATGTAT
      CATGCCTCTT TGCACCATTC TAAAGAATAA CAGTGATAAT TTCTGGGTTA
      AGGCAATAGC AATATCTCTG CATATAAATA TTTCTGCATA TAAATTGTAA
      CTGATGTAAG AGGTTTCATA TTGCTAATAG CAGCTACAAT CCAGCTACCA
      TTCTGCTTTT ATTTTATGGT TGGGATAAGG CTGGATTATT CTGAGTCCAA
      GCTAGGCCCT TTTGCTAATC ATGTTCATAC CTCTTATCTT CCTCCCACAG

Exon 3

366                    CTCCT GGGCAACGTG CTGGTCTGTG TGCTGGCCCA
401   TCACTTTGGC AAAGAATTCA CCCCACCAGT GCAGGCTGCC TATCAGAAAG
451   TGGTGGCTGG TGTGGCTAAT GCCCTGGCCC ACAAGTATCA CTAAGCTCGC
501   TTTCTTGCTG TCCAATTTCT ATTAAAGGTT CCTTTGTTCC CTAAGTCCAA
551   CTACTAAACT GGGGGATATT ATGAAGGGCC TTGAGCATCT GGATTCTGCC
```

```
601   TAATAAAAAA CATTTATTTT CATTGC

Downstream Element

      aatgatgtat ttaaattatt tctgaatatt ttactaaaaa gggaatgtgg
      gaggtcagtg catttaaaac ataaagaaat gaagagctag ttcaaacctt
      gggaaaatac actatatctt aaactccatg aaagaa
```

Appendix B: Database Loading Script

```
-------------------------------------------------------------------------------
---
-- Drop tables
-------------------------------------------------------------------------------
---
drop table mutation cascade;
drop table ethnicity cascade;
drop table comment cascade;
drop table hbb_wild cascade;
drop table hbb_mute cascade;
drop table raw cascade;
drop table reference cascade;
drop table author cascade;


-------------------------------------------------------------------------------
---
-- Static tables
-------------------------------------------------------------------------------
---

--
-- mutation
--
create table mutation (
      id serial primary key,
      chrom_start integer,
      chrom_end integer,
      hbvar_id integer,
      common_name text,
      hgvs_name text,
      mutation_type text,
      mutation_desc text
);

--
-- ethnicity
--
create table ethnicity (
      id serial primary key,
      hgvs_name text,
      hbvar_id integer,
      low_freq decimal,
      high_freq decimal,
      ethnicity_desc text
);

--
-- comment
--
create table comment (
      id serial primary key,
      hgvs_name text,
      hbvar_id integer,
      comment_desc text
);

--
-- Reference
--
create table reference (
  id serial primary key,
  hbvar_id integer,
  ref_id integer,
  ref_id1 integer,
  medline_id text,
  title text,
  journal text,
  year integer,
  volume text,
  num text,
  pp text
);

--
-- Author
--
create table author (
  id serial primary key,
  ref_id integer,
  author text,
  order_by integer,
  researcher_id text
);


-----------------------------------------------------------------------------
---
-- Load static tables
-----------------------------------------------------------------------------
---

--
-- copy into mutation
--
copy mutation(chrom_start, chrom_end, hbvar_id, common_name, hgvs_name,
mutation_type)
from 'C:\Program Files\PostgreSQL\9.1\mutation_no_header_good_hbVarId.txt';
```

```
--
-- massage mutation data
--

update mutation set common_name = regexp_replace(common_name, '"', '');
update mutation set common_name = regexp_replace(common_name, '; *$', '');
update mutation set common_name = regexp_replace(common_name, 'Dominant.*',
'');

--
-- copy into ethnicity
--
copy ethnicity(hgvs_name, hbvar_id, low_freq, high_freq, ethnicity_desc)
from 'C:\Program Files\PostgreSQL\9.1\ethnicity_no_header_only_beta_null.txt';

--
-- copy into comment
--
copy comment(hgvs_name, hbvar_id, comment_desc)
from 'C:\Program Files\PostgreSQL\9.1\comment_no_header.txt';

--
-- massage comment data
--
update comment set comment_desc = regexp_replace(comment_desc, 'null', '');

--
-- copy into reference
--
copy reference(hbvar_id, ref_id, ref_id1, medline_id, title, journal, year,
volume, num, pp)
from 'C:\Program Files\PostgreSQL\9.1\refs_no_header.txt';

--
-- copy into author
--
copy author(ref_id, author, order_by, researcher_id)
from 'C:\Program Files\PostgreSQL\9.1\auths_no_header.txt';


------------------------------------------------------------------------------
---
-- Working Tables
------------------------------------------------------------------------------
---

--
-- raw
--
create table raw (
      id serial primary key,
      seq text,
      wild_desc text
);

--
```

```
-- hbb_wild
--
create table hbb_wild (
      id serial primary key,
      seq_pos integer,
      cds_pos integer,
      codon integer,
      base text,
      wild_desc text
);

--
-- hbb_mute
--
create table hbb_mute (
      id serial primary key,
      hbvar_id integer,
      common_name text,
      hgvs_name text,
      seq_pos integer,
      cds_pos integer,
      codon integer,
      base text,
      wild_desc text,
      mute_desc text,
      mute_key text
);


-----------------------------------------------------------------------------
---
-- Pl/PgSQL functions for loading working tables
-----------------------------------------------------------------------------
---

--
-- load_hbb_wild()
--
create or replace function load_hbb_wild() returns void as $$
declare
  i integer;
  j integer;
  k integer;
  b text;
  s text;
  s_len integer;
  c integer;
begin

  --j := -150;    -- cds_pos
  select -sum(length(seq)) into j from raw where id in (1, 2);
  k := 1;   -- seq_pos

  -- Upstream
  i := 1;
  select seq into s from raw where id = 1;
  select length(s) into s_len from raw where id = 1;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc) values (k, j, b,
'Upstream');
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- 5' UTR
  i := 1;
  select seq into s from raw where id = 2;
  select length(s) into s_len from raw where id = 2;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc) values (k, j, b,
'5'' UTR');
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- CDS 1
  i := 1;
  j := 1;
  select seq into s from raw where id = 3;
  select length(s) into s_len from raw where id = 3;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values (k,
j, b, 'CDS 1', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- IVS 1
  i := 1;
  select seq into s from raw where id = 4;
  select length(s) into s_len from raw where id = 4;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, 'IVS 1');
    i = i + 1;
    k = k + 1;
  end loop;

  -- CDS 2
  i := 1;
  select seq into s from raw where id = 5;
  select length(s) into s_len from raw where id = 5;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values (k,
j, b, 'CDS 2', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- IVS 2
  i := 1;
  select seq into s from raw where id = 6;
  select length(s) into s_len from raw where id = 6;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, 'IVS 2');
    i = i + 1;
    k = k + 1;
  end loop;

  -- CDS 3
  i := 1;
  select seq into s from raw where id = 7;
  select length(s) into s_len from raw where id = 7;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values
(k,j, b, 'CDS 3', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- 3' UTR
  i := 1;
  select seq into s from raw where id = 8;
  select length(s) into s_len from raw where id = 8;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, '3'' UTR');
    i = i + 1;
    k = k + 1;
  end loop;

  -- Downstream
  i := 1;
  select seq into s from raw where id = 9;
  select length(s) into s_len from raw where id = 9;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b,
'Downstream');
    i = i + 1;
    k = k + 1;
  end loop;

end;
$$ language plpgsql;

--
-- load_hbb_mute()
--
create or replace function load_hbb_mute() returns void as $$
declare
  i integer;
  c text;
  h text;
  s hbb_wild%rowtype;
begin
  for i, c, h in select distinct hbvar_id, common_name, hgvs_name from
mutation
  loop
    for s in select * from hbb_wild
    loop
      insert into hbb_mute (hbvar_id, hgvs_name, common_name, seq_pos,
cds_pos, base, wild_desc, mute_desc, codon) values (i, h, c, s.seq_pos,
s.cds_pos, s.base, s.wild_desc, s.wild_desc, s.codon);
    end loop;
  end loop;
end;
$$ language plpgsql;


-----------------------------------------------------------------------------
---
-- Load working tables
-----------------------------------------------------------------------------
---

--
-- insert into raw
--
insert into raw (seq, wild_desc) values
('CTGTGGAGCCACACCCTAGGGTTGGCCAATCTACTCCCAGGAGCAGGGAGGGCAGGAGCCAGGGCTGGGCATAAA
AGTCAGGGCAGAGCCATCTATTGCTT', 'Upstream');
insert into raw (seq, wild_desc) values
('ACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACAGACACC', '5'' UTR');
```

```
insert into raw (seq, wild_desc) values
('ATGGTGCATCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGTTGGT
GGTGAGGCCCTGGGCAG', 'CDS 1');
insert into raw (seq, wild_desc) values
('GTTGGTATCAAGGTTACAAGACAGGTTTAAGGAGACCAATAGAAACTGGGCATGTGGAGACAGAGAAGACTCTTG
GGTTTCTGATAGGCACTGACTCTCTCTGCCTATTGGTCTATTTTCCCACCCTTAG', 'IVS 1');
insert into raw (seq, wild_desc) values
('GCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCCACTCCTGATGCTGTTAT
GGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGTGCCTTTAGTGATGGCCTGGCTCACCTGGACAACC
TCAAGGGCACCTTTGCCACACTGAGTGAGCTGCACTGTGACAAGCTGCACGTGGATCCTGAGAACTTCAGG',
'CDS 2');
insert into raw (seq, wild_desc) values
('GTGAGTCTATGGGACGCTTGATGTTTTCTTTCCCCTTCTTTTCTATGGTTAAGTTCATGTCATAGGAAGGGGATA
AGTAACAGGGTACAGTTTAGAATGGGAAACAGACGAATGATTGCATCAGTGTGGAAGTCTCAGGATCGTTTTAGTTT
CTTTTATTTGCTGTTCATAACAATTGTTTTCTTTTGTTTAATTCTTGCTTTCTTTTTTTTTCTTCTCCGCAATTTTT
ACTATTATACTTAATGCCTTAACATTGTGTATAACAAAAGGAAATATCTCTGAGATACATTAAGTAACTTAAAAAAA
AACTTTACACAGTCTGCCTAGTACATTACTATTTGGAATATATGTGTGCTTATTTGCATATTCATAATCTCCCTACT
TTATTTTCTTTTATTTTTAATTGATACATAATCATTATACATATTTATGGGTTAAAGTGTAATGTTTTAATATGTGT
ACACATATTGACCAAATCAGGGTAATTTTGCATTTGTAATTTTAAAAAATGCTTTCTTCTTTTAATATACTTTTTTG
TTTATCTTATTTCTAATACTTTCCCTAATCTCTTTCTTTCAGGGCAATAATGATACAATGTATCATGCCTCTTTGCA
CCATTCTAAAGAATAACAGTGATAATTTCTGGGTTAAGGCAATAGCAATATCTCTGCATATAAATATTTCTGCATAT
AAATTGTAACTGATGTAAGAGGTTTCATATTGCTAATAGCAGCTACAATCCAGCTACCATTCTGCTTTTATTTTATG
GTTGGGATAAGGCTGGATTATTCTGAGTCCAAGCTAGGCCCTTTTGCTAATCATGTTCATACCTCTTATCTTCCTCC
CACAG', 'IVS 2');
insert into raw (seq, wild_desc) values
('CTCCTGGGCAACGTGCTGGTCTGTGTGCTGGCCCATCACTTTGGCAAAGAATTCACCCCACCAGTGCAGGCTGCC
TATCAGAAAGTGGTGGCTGGTGTGGCTAATGCCCTGGCCCACAAGTATCACTAA', 'CDS 3');
insert into raw (seq, wild_desc) values
('GCTCGCTTTCTTGCTGTCCAATTTCTATTAAAGGTTCCTTTGTTCCCTAAGTCCAACTACTAAACTGGGGGATAT
TATGAAGGGCCTTGAGCATCTGGATTCTGCCTAATAAAAAACATTTATTTTCATTGC', '3'' UTR');
insert into raw (seq, wild_desc) values
('AATGATGTATTTAAATTATTTCTGAATATTTTACTAAAAAGGGAATGTGGGAGGTCAGTGCATTTAAAACATAAA
GAAATGAAGAGCTAGTTCAAACCTTGGGAAAATACACTATATCTTAAACTCCATGAAAGAA',  'Downstream');

--
-- load hbb_wild
--
select load_hbb_wild();

--
-- load hbb_mute
--
select load_hbb_mute();


-----------------------------------------------------------------------------
---
-- Pl/PgSQL function for modifying HBB_MUTE
-----------------------------------------------------------------------------
---

--
-- single_sub()
--
create or replace function single_sub() returns void as $$
declare
  i integer;
  p integer;
  s text;
  sp integer;
  w text;
  m text;
begin
  for i, p, s, w, m in select hbvar_id,
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[1],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[2],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[3],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[4]
                       from mutation
  loop
    -- get the seq_pos of p + s
    if s = '' then
      select seq_pos into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    else
      select seq_pos + s::integer into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    end if;
    -- substitute this base
    update hbb_mute set base = m, mute_key = 'sub'
    where hbvar_id = i
    and seq_pos = sp
    and base = w;
  end loop;
end;
$$ language plpgsql;

--
-- post_cds_single_sub()
--
create or replace function post_cds_single_sub() returns void as $$
declare
  i integer;
  d integer;
  p integer;
  w text;
  m text;
begin
  for i, d, w, m in select hbvar_id,
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[1],
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[2],
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[3]
                     from mutation
  loop
    -- find the seq_pos of the last base in CDS
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = 444;

    update hbb_mute set base = m, mute_key = 'sub'
    where hbvar_id = i
    and seq_pos = p + d
    and base = w;
  end loop;
end;
$$ language plpgsql;

--
-- single_del()
--
create or replace function single_del() returns void as $$
declare
  i integer;
  p integer;
  s text;
  sp integer;
  w text;
begin
  for i, p, s, w in select hbvar_id,
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[1],
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[2],
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[3]
                 from mutation
  loop
    -- get seq_pos of this base
    if s = '' then
      select seq_pos into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    else
      select seq_pos + s::integer into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    end if;
    -- delete this base
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos = sp
    and base = w;
  end loop;
end;
$$ language plpgsql;

--
-- multi_del()
--
create or replace function multi_del() returns void as $$
declare
```

```
  i integer;
  f integer;
  t integer;
  fm text;--integer;
  tm text;--integer;
  fs integer;
  ts integer;
begin
  for i, f, fm, t, tm in select hbvar_id,
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[1],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[2],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[3],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[4]
                 from mutation
  loop
    -- get the seq_pos of f + fm and t + tm
    if fm = '' then
      select seq_pos into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    else
      select seq_pos + fm::integer into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    end if;
    if tm = '' then
      select seq_pos into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    else
      select seq_pos + tm::integer into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    end if;

    -- delete these bases
    if fs is null then
      fs = 0;
    end if;
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos between fs and ts;

  end loop;
end;
$$ language plpgsql;

--
-- cds_ins()
--
create or replace function cds_ins() returns void as $$
declare
  i integer;
  f integer;
  t integer;
  p integer;
  l integer;
  j integer;
  m text;
  c text;
  h text;
begin
  for i, c, h, f, t, m in select hbvar_id, common_name, hgvs_name,
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[1],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[2],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[3]
                    from mutation
  loop
    -- find the seq_pos of f
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = f;

    -- update the rest of the seq_pos
    l = length(m);
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > p;

    -- insert these bases
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base,
mute_key)
      values (i, c, h, p + j, substring(m from j for 1), 'ins');
    end loop;
  end loop;
end;
$$ language plpgsql;

--
-- single_delins()
--
create or replace function single_delins() returns void as $$
declare
  i integer;
  f integer;
  p integer;
  l integer;
  j integer;
  m text;
  c text;
  h text;
begin
  for i, c, h, f, m in select distinct hbvar_id, common_name, hgvs_name,
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)delins([ACGT]+)'))[1],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)delins([ACGT]+)'))[2]
                       from mutation
  loop
    -- delete
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and cds_pos = f;

    -- find the seq_pos of f
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = f;

    -- update the rest of the seq_pos
    l = length(m) + 1;
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > p;

    -- insert these bases
    l = length(m);
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base,
mute_key)
      values (i, c, h, p + j, substring(m from j for 1), 'ins');
    end loop;

  end loop;
end;
$$ language plpgsql;

--
-- multi_delins()
--
create or replace function multi_delins() returns void as $$
declare
  i integer;
  c text;
  h text;
  f integer;
  t integer;
  fm text;--integer;
  tm text;--integer;
  fs integer;
  ts integer;
  m text;
  l integer;
  j integer;
begin
  for i, c, h, f, fm, t, tm, m in select distinct hbvar_id, common_name,
hgvs_name,
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[1],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[2],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[3],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[4],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[5]
                            from mutation
  loop
    -- get the seq_pos of f + fm and t + tm
    if fm = '' then
      select seq_pos into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    else
      select seq_pos + fm::integer into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    end if;
    if tm = '' then
      select seq_pos into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    else
      select seq_pos + tm::integer into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    end if;

    -- delete these bases
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos between fs and ts;

    -- update the rest of the seq_pos
    l = length(m) + (ts - fs + 1);
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > ts;

    -- insert these bases
    l = length(m);
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base, 
mute_key)
      values (i, c, h, ts + j, substring(m from j for 1), 'ins');
    end loop;

  end loop;
end;
$$ language plpgsql;
```

```
---------------------------------------------------------------------------
---
-- Additional comments
---------------------------------------------------------------------------
---
insert into comment (hbvar_id, comment_desc) values (256, 'The AAC->AGC
mutation at codon 19 creates an alternative splicing site between codons 17
and 18, reducing the efficiency of the normal donor site at IVS-I to about
60%.');
update comment set comment_desc = 'This T->C mutation at nt 6 of IVS-I
reduces the efficiency of splicing at the 5'' site (see scheme)<BR>
<TABLE>
  <TR>
    <TD>Codon #</TD>
    <TD>30</TD>
    <TD>IVS-I</TD>
  </TR>
  <TR>
    <TD>Normal</TD>
    <TD>AG^</TD>
    <TD><u>GT</u>TGG<u><i>T</i></u>ATCAAG<u>GT</u>T</TD>
  </TR>
  <TR>
    <TD>IVS-I-6 (T->C)</TD>
    <TD>AG^</TD>
    <TD><u>GT</u>TGG<u><i>C</i></u>ATCAAG<u>GT</u>T</TD>
  </TR>
</TABLE>' where hbvar_id = 826;


---------------------------------------------------------------------------
---
-- Modfy HBB_MUTE with mutations
---------------------------------------------------------------------------
---
select single_sub();
select post_cds_single_sub();
select single_del();
select multi_del();
select cds_ins();
select single_delins();
select multi_delins();

--
-- single base substitution (only CDS should be Gone)
--
-- 256
update hbb_mute set mute_key = 'CSS' where hbvar_id = 256 and seq_pos between
206 and 207;
update hbb_mute set mute_desc = 'IVS 1' where hbvar_id = 256 and seq_pos
between 206 and 243;
update hbb_mute set mute_key = 'STOP' where hbvar_id = 256 and seq_pos
between 410 and 412;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 256 and seq_pos
between 413 and 1708;
-- 281, 805, 825
```

```
update hbb_mute set mute_key = 'CSS' where hbvar_id in (281, 805, 825) and
seq_pos between 228 and 229;
update hbb_mute set mute_desc = 'IVS 1' where hbvar_id in (281, 805, 825) and
seq_pos between 228 and 243;
update hbb_mute set mute_key = 'STOP' where hbvar_id in (281, 805, 825) and
seq_pos between 463 and 465;

update hbb_mute set mute_desc = 'Gone' where hbvar_id in (281, 805, 825) and
seq_pos between 466 and 1708;

-- 290
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id = 290 and wild_desc =
'IVS 1';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 290 and seq_pos
between 335 and 337;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 290 and seq_pos
between 338 and 1708;
-- 775 - 780
update hbb_mute set mute_desc = '5'' UTR' where hbvar_id between 775 and 780
and seq_pos between 152 and 215;
update hbb_mute set mute_key = 'START' where hbvar_id between 775 and 780 and
seq_pos between 216 and 218;
update hbb_mute set mute_key = 'STOP' where hbvar_id between 775 and 780 and
seq_pos between 463 and 465;
update hbb_mute set mute_desc = 'Gone' where hbvar_id between 775 and 780 and
seq_pos between 466 and 1708;
-- 791
update hbb_mute set mute_key = 'STOP' where hbvar_id = 791 and codon = 15 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 791 and seq_pos
between 200 and 1708;
-- 793
update hbb_mute set mute_key = 'STOP' where hbvar_id = 793 and codon = 15 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 793 and seq_pos
between 200 and 1708;
-- 800
update hbb_mute set mute_key = 'STOP' where hbvar_id = 800 and codon = 17 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 800 and seq_pos
between 206 and 1708;
-- 802
update hbb_mute set mute_key = 'STOP' where hbvar_id = 802 and codon = 22 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 802 and seq_pos
between 221 and 1708;
-- 808
update hbb_mute set mute_key = 'STOP' where hbvar_id = 808 and codon = 26 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 808 and seq_pos
between 233 and 1708;
-- 814
update hbb_mute set mute_key = 'CSS' where hbvar_id = 814 and seq_pos = 240;
-- seq_pos 241 is sub
update hbb_mute set mute_desc = 'IVS 1' where hbvar_id = 814 and seq_pos
between 240 and 243;
```

```
update hbb_mute set mute_key = 'STOP' where hbvar_id = 814 and seq_pos
between 547 and 549;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 814 and seq_pos
between 550 and 1708;
-- 815 - 821
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id between 815 and 821
and wild_desc = 'IVS 1';
update hbb_mute set mute_key = 'STOP' where hbvar_id between 815 and 821 and
seq_pos between 335 and 337;
update hbb_mute set mute_desc = 'Gone' where hbvar_id between 815 and 821 and
seq_pos between 338 and 1708;

-- 822 has 2 more CSS, one at codon 25 and one at IVS-I 13
--update hbb_mute set mute_key = 'CSS' where hbvar_id = 822 and seq_pos
between 206 and 207;
--update hbb_mute set mute_desc = 'IVS 1' where hbvar_id = 822 and seq_pos
between 206 and 243;
-- 824, 826
update hbb_mute set mute_key = 'CSS' where hbvar_id in (824, 826) and seq_pos
between 256 and 257;
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id in (824, 826) and
seq_pos between 244 and 255;
-- 827
update hbb_mute set mute_key = 'CSS' where hbvar_id = 827 and seq_pos = 354;
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 827 and seq_pos
between 355 and 373;
update hbb_mute set mute_key = 'STOP' where hbvar_id = 827 and seq_pos
between 371 and 373;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 827 and seq_pos
between 374 and 1708;
-- 828
update hbb_mute set mute_key = 'CSS' where hbvar_id = 828 and seq_pos = 358;
-- seq_pos 359 is sub
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 828 and seq_pos
between 360 and 373;
update hbb_mute set mute_key = 'STOP' where hbvar_id = 828 and seq_pos
between 463 and 465;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 828 and seq_pos
between 466 and 1708;
-- 838
update hbb_mute set mute_key = 'STOP' where hbvar_id = 838 and codon = 35 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 838 and seq_pos
between 390 and 1708;
-- 841
update hbb_mute set mute_key = 'STOP' where hbvar_id = 841 and codon = 37 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 841 and seq_pos
between 396 and 1708;
-- 845
update hbb_mute set mute_key = 'STOP' where hbvar_id = 845 and codon = 39 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 845 and seq_pos
between 402 and 1708;
-- 853
update hbb_mute set mute_key = 'STOP' where hbvar_id = 853 and codon = 43 and
mute_key is null;
```

```
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 853 and seq_pos
between 414 and 1708;
-- 866
update hbb_mute set mute_key = 'STOP' where hbvar_id = 866 and codon = 61 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 866 and seq_pos
between 468 and 1708;
-- 880
update hbb_mute set mute_key = 'STOP' where hbvar_id = 880 and codon = 90 and
mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 880 and seq_pos
between 555 and 1708;
-- 884
update hbb_mute set mute_desc = 'CDS2' where hbvar_id = 884 and wild_desc =
'IVS 2';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 884 and seq_pos
between 615 and 617;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 884 and seq_pos
between 618 and 1708;
-- 885
update hbb_mute set mute_desc = 'CDS2' where hbvar_id = 885 and wild_desc =
'IVS 2';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 885 and seq_pos
between 615 and 617;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 885 and seq_pos
between 618 and 1708;
-- 889
update hbb_mute set mute_key = 'CSS' where hbvar_id = 889 and seq_pos between
1247 and 1248;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 889 and seq_pos
between 1249 and 1446;
update hbb_mute set mute_key = 'STOP' where hbvar_id = 889 and seq_pos
between 1294 and 1296;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 889 and seq_pos
between 1297 and 1708;
--890
update hbb_mute set mute_key = 'CSS' where hbvar_id = 890 and seq_pos between
1174 and 1175;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 890 and seq_pos
between 1176 and 1340;
update hbb_mute set mute_key = 'CSS' where hbvar_id = 890 and seq_pos = 1300;
-- seq_pos 1301 is sub
update hbb_mute set mute_desc = 'IVS 3' where hbvar_id = 890 and seq_pos
between 1302 and 1446;
update hbb_mute set mute_key = 'CSS' where hbvar_id = 890 and seq_pos between
1302 and 1303;
update hbb_mute set mute_desc = 'CDS 4' where hbvar_id = 890 and wild_desc =
'CDS 3';
-- 891
update hbb_mute set mute_key = 'CSS' where hbvar_id = 891 and seq_pos between
1174 and 1175 and mute_key is null; -- new
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 891 and seq_pos
between 1176 and 1340;
update hbb_mute set mute_key = 'CSS' where hbvar_id = 891 and seq_pos between
1341 and 1342 and mute_key is null; -- new
update hbb_mute set mute_desc = 'IVS 3' where hbvar_id = 891 and seq_pos
between 1341 and 1446;
```

```
update hbb_mute set mute_desc = 'CDS 4' where hbvar_id = 891 and wild_desc =
'CDS 3';
-- 901
update hbb_mute set mute_key = 'CSS' where hbvar_id = 901 and seq_pos between
1174 and 1175;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 901 and seq_pos
between 1176 and 126;
update hbb_mute set mute_key = 'STOP' where hbvar_id = 901 and seq_pos
between 1224 and 1226;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 901 and seq_pos
between 1227 and 1708;
-- 939, 940, 942, 943, 944
update hbb_mute set mute_desc = 'IVS 2' where hbvar_id in (939, 940, 942, 943,
944) and wild_desc = 'CDS 3';
-- 948
update hbb_mute set mute_key = 'STOP' where hbvar_id = 948 and codon = 112
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 948 and seq_pos
between 1471 and 1708;
-- 951
update hbb_mute set mute_key = 'STOP' where hbvar_id = 951 and codon = 121
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 951 and seq_pos
between 1498 and 1708;
-- 961
update hbb_mute set mute_key = 'STOP' where hbvar_id = 961 and codon = 127
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 961 and seq_pos
between 1516 and 1708;
-- 2528
update hbb_mute set mute_key = 'STOP' where hbvar_id = 2528 and codon = 59
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 2528 and seq_pos
between 462 and 1708;
-- 2529
update hbb_mute set mute_key = 'STOP' where hbvar_id = 2529 and codon = 37
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 2529 and seq_pos
between 396 and 1708;
-- 2548
update hbb_mute set mute_key = 'STOP' where hbvar_id = 2548 and codon = 132
and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 2548 and seq_pos
between 1531 and 1708;

--
-- ins
--
-- 786, 787, 790
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id in (786, 787, 790) and
mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (786, 787, 790) and
seq_pos between 218 and 220;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (786, 787, 790) and
seq_pos between 221 and 1708 + 1;
-- 807
```

```
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id = 807 and mute_key =
'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 807 and seq_pos
between 230 and 232 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 807 and seq_pos
between 233 and 1708 + 1;
-- 809, 810, 847, 851
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id in (809, 810, 847, 851)
and mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (809, 810, 847, 851)
and seq_pos between 411 and 413 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (809, 810, 847, 851)
and seq_pos between 414 and 1708 + 1;
-- 836
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 836 and mute_key =
'ins';
-- 850, 856
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id in (850, 856) and
mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (850, 856) and
seq_pos between 438 and 440 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (850, 856) and
seq_pos between 441 and 1708 + 1;
-- 857
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 857 and mute_key =
'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 857 and seq_pos
between 441 and 443 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 857 and seq_pos
between 444 and 1708 + 4;
-- 860, 862, 864
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id in (860, 862, 864) and
mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (860, 862, 864) and
seq_pos between 459 and 461 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (860, 862, 864) and
seq_pos between 462 and 1708 + 1;
-- 869
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 869 and mute_key =
'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 869 and seq_pos
between 501 and 503 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 869 and seq_pos
between 504 and 1708 + 1;
-- 870
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 870 and mute_key =
'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 870 and seq_pos
between 498 and 500 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 870 and seq_pos
between 501 and 1708 + 1;
-- 876, 877
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id in (876, 877, 878) and
mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (876, 877, 878) and
seq_pos between 552 and 554 and mute_key is null;
```

```
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (876, 877, 878) and
seq_pos between 555 and 1708 + 1;
-- 881
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 881 and mute_key =
'ins';
update hbb_mute set mute_key = 'NEW' where hbvar_id = 881 and seq_pos between
1576 + 2 and 1608 and mute_key is null;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 881 and seq_pos
between 1708 + 2 and 1608;
-- 882
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id = 882 and mute_key =
'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id = 882 and seq_pos
between 585 and 587 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 882 and seq_pos
between 588 and 1708 + 1;
-- 945, 950
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id in (945, 950) and
mute_key = 'ins';
update hbb_mute set mute_key = 'STOP' where hbvar_id in (945, 950) and
seq_pos between 1549 and 1551 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (945, 950) and
seq_pos between 1552 and 1708 + 1;

--
-- del
--
-- 781
update hbb_mute set mute_key = 'STOP' where hbvar_id = 781 and seq_pos
between 162 and 164 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 781 and seq_pos
between 165 and 1708;
-- 783, 785
update hbb_mute set mute_key = 'STOP' where hbvar_id in (783, 785) and
seq_pos between 217 and 219 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (783, 785) and
seq_pos between 220 and 1708;
-- 784, 789, 798, 799
update hbb_mute set mute_key = 'STOP' where hbvar_id in (784, 789, 798, 799)
and seq_pos between 207 and 209 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (784, 789, 798, 799)
and seq_pos between 210 and 1708;
-- 803, 811, 813, 835, 839, 840, 843, 846, 848, 849, 854, 855, 858, 861, 865
update hbb_mute set mute_key = 'STOP' where hbvar_id in (803, 811, 813, 835,
839, 840, 843, 846, 848, 849, 854, 855, 858, 861, 865) and seq_pos between
463 and 465 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (803, 811, 813, 835,
839, 840, 843, 846, 848, 849, 854, 855, 858, 861, 865) and seq_pos between
466 and 1708;
-- 842
update hbb_mute set mute_key = 'STOP' where hbvar_id = 842 and seq_pos
between 393 and 402 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 842 and seq_pos
between 403 and 1708;
-- 844
update hbb_mute set mute_key = 'STOP' where hbvar_id = 844 and seq_pos
between 410 and 412 and mute_key is null;
```

```
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 844 and seq_pos
between 413 and 1708;
-- 867, 873, 874, 875
update hbb_mute set mute_key = 'STOP' where hbvar_id in (867, 873, 874, 875)
and seq_pos between 547 and 549 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (867, 873, 874, 875)
and seq_pos between 550 and 1708;
-- 868
update hbb_mute set mute_key = 'STOP' where hbvar_id = 868 and seq_pos
between 497 and 499 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 868 and seq_pos
between 500 and 1708;
-- 879
update hbb_mute set mute_key = 'STOP' where hbvar_id = 879 and seq_pos
between 563 and 565 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 879 and seq_pos
between 566 and 1708;
-- 941
update hbb_mute set mute_desc = 'IVS 2' where hbvar_id = 941 and seq_pos
between 1447 and 1708;
-- 947, 952, 954, 955, 957
update hbb_mute set mute_key = 'NEW' where hbvar_id in (947, 952, 954, 955,
957) and seq_pos between 1576 and 1606 and mute_key is null;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id in (947, 952, 954, 955,
957) and seq_pos between 1708 and 1606;
-- 958, 2510
update hbb_mute set mute_key = 'STOP' where hbvar_id in (958, 2510) and
seq_pos between 1548 and 1550 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id in (958, 2510) and
seq_pos between 1551 and 1708;
-- 2538
update hbb_mute set mute_key = 'STOP' where hbvar_id = 2538 and seq_pos
between 551 and 553 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 2538 and seq_pos
between 554 and 1708;
-- 2808
update hbb_mute set mute_key = 'STOP' where hbvar_id = 2808 and seq_pos
between 463 and 565 and mute_key is null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 2808 and seq_pos
between 466 and 1708;

--
-- delins
--
-- 782
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id = 782 and seq_pos
between 167 and 197;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 782 and seq_pos
between 188 and 1747;
-- 804
update hbb_mute set mute_desc = 'CDS 1' where hbvar_id = 804 and seq_pos
between 226 and 228;
update hbb_mute set mute_key = 'Stop' where hbvar_id = 804 and seq_pos
between 467 and 469 and mute_key is not null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 804 and seq_pos
between 470 and 1711;
-- 871
```

```
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 871 and seq_pos =503;
update hbb_mute set mute_key = 'Stop' where hbvar_id = 871 and seq_pos
between 553 and 555 and mute_key is not null;
update hbb_mute set mute_desc = 'Gone' where hbvar_id = 871 and seq_pos
between 556 and 1713;
-- 883
update hbb_mute set mute_desc = 'CDS 2' where hbvar_id = 883 and seq_pos
between 585 and 595;
update hbb_mute set mute_key = 'Stop' where hbvar_id = 883 and seq_pos
between 1618 and 1620 and mute_key is not null;
update hbb_mute set mute_desc = 'CDS 3', mute_key = 'New' where hbvar_id =
883 and seq_pos between 1590 and 1620;
-- 949
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 949 and seq_pos =
1476;
update hbb_mute set mute_key = 'Stop' where hbvar_id = 949 and seq_pos
between 1607 and 1609 and mute_key is not null;
update hbb_mute set mute_desc = 'CDS 3', mute_key = 'New' where hbvar_id =
949 and seq_pos between 1579 and 1609;
-- 963
update hbb_mute set mute_key = 'del' where hbvar_id = 963 and cds_pos between
385 and 388;
update hbb_mute set seq_pos = seq_pos + 5 where hbvar_id = 963 and seq_pos >
1519;
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1520,
'C', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1521,
'C', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1522,
'A', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1523,
'C', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1524,
'A', 'ins');
update hbb_mute set mute_key = 'del' where hbvar_id = 963 and cds_pos between
397 and 407;
update hbb_mute set seq_pos = seq_pos + 11 where hbvar_id = 963 and seq_pos >
1543;
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1544,
'A', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1545,
'A', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1546,
'A', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1547,
'G', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1548,
'T', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1549,
'G', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1550,
'G', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1551,
'T', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1552,
'G', 'ins');
```

```
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1553,
'G', 'ins');
insert into hbb_mute (hbvar_id, seq_pos, base, mute_key) values (963, 1554,
'C', 'ins');
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 963 and seq_pos
between 1520 and 1524 and mute_key is not null;
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 963 and seq_pos
between 1544 and 1554 and mute_key is not null;
update hbb_mute set mute_desc = 'CDS 3', mute_key = 'New' where hbvar_id =
963 and seq_pos between 1592 and 1641;
-- 964
update hbb_mute set mute_desc = 'CDS 3' where hbvar_id = 964 and seq_pos
between 1545 and 1548;



--
-- What other mutation has not been done?
--
select hbvar_id, hgvs_name from mutation where hbvar_id in (
select distinct hbvar_id from hbb_mute where mute_key is null
except
select distinct hbvar_id from hbb_mute where mute_key is not null) order by 2;
```

Appendix C: Database Functions

```
-- Function: cds_ins()

-- DROP FUNCTION cds_ins();

CREATE OR REPLACE FUNCTION cds_ins()
  RETURNS void AS
$BODY$
declare
  i integer;
  f integer;
  t integer;
  p integer;
  l integer;
  j integer;
  m text;
  c text;
  h text;
begin
  for i, c, h, f, t, m in select hbvar_id, common_name, hgvs_name,
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[1],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[2],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)_(\\d+)ins([ACGT]+)'))[3]
```

```
                    from mutation
  loop
    -- find the seq_pos of f
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = f;

    -- update the rest of the seq_pos
    l = length(m);
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > p;

    -- insert these bases
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base, 
mute_key)
      values (i, c, h, p + j, substring(m from j for 1), 'ins');
    end loop;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION cds_ins()
  OWNER TO postgres;



-- Function: chk_cds_single_del()

-- DROP FUNCTION chk_cds_single_del();

CREATE OR REPLACE FUNCTION chk_cds_single_del()
  RETURNS void AS
$BODY$
declare
  a integer;
  b integer;
  c integer;
  d integer;
  e text;
  f text;
  g text;
  h text;
  i integer;
  j integer;
  x integer;
  stop text;
begin
  for a, b, c in select hbvar_id, codon, cds_pos
                 from hbb_mute
                 where mute_type = 'single_del()'
                 and type like 'CDS%'
  loop
    select min(cds_pos) into x from hbb_mute where hbvar_id = a and codon = b;

    if c = x then d = 1;
    elsif (c = x + 1) then d = 2;
    else d = 3;
    end if;

    select base into e from hbb_mute where hbvar_id = a and codon = b and
cds_pos = c;
    select base into f from hbb_mute where hbvar_id = a and codon = b and
cds_pos = c + 1;
    select base into g from hbb_mute where hbvar_id = a and codon = b and
cds_pos = c + 2;
    select base into h from hbb_mute where hbvar_id = a and codon = b + 1 and
cds_pos = c + 3;

    stop = '';
    if d = 1 then  stop = f || g || h;
    elsif d = 2 then stop = e || g || h;
    else stop = e || f || h;
    end if;

    if stop in ('TGA', 'TAG', 'TAA') then
      update hbb_mute set mute_desc = 'STOP' where hbvar_id = a and codon = b;
    end if;

    c = c + 4;
    select seq_pos into i from hbb_mute where hbvar_id = a and cds_pos = c;
    select min(seq_pos) into j from hbb_mute where hbvar_id = a and cds_pos
is null and seq_pos > i;

    while (i < j) loop
      select base into e from hbb_mute where hbvar_id = a and cds_pos is not
null and seq_pos = i;
      select base into f from hbb_mute where hbvar_id = a and codon is not
null and seq_pos = i + 1;
      select base into g from hbb_mute where hbvar_id = a and codon is not
null and seq_pos = i + 2;
      stop = e || f || g;
      if stop in ('TGA', 'TAG', 'TAA') then
        raise notice '%', stop;
        update hbb_mute set mute_desc = 'STOP' where hbvar_id = a and seq_pos
between i and i + 2;
        update mutation set notes = 'YY' where hbvar_id = a;
        exit;
      end if;
      i = i + 3;
    end loop;

  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION chk_cds_single_del()
  OWNER TO postgres;
```

```
-- Function: chk_cds_single_sub()

-- DROP FUNCTION chk_cds_single_sub();

CREATE OR REPLACE FUNCTION chk_cds_single_sub()
  RETURNS void AS
$BODY$
declare
  a integer;
  b integer;
  c text;
  stop text;
begin
  for a, b in select hbvar_id, codon
              from hbb_mute
              where mute_type = 'single_sub()'
              and type like 'CDS%'
  loop
    stop = '';
    for c in select base
             from hbb_mute
             where hbvar_id = a
             and codon = b
             order by seq_pos
    loop
      stop = stop || c;
    end loop;
    raise notice '%', stop;
    if stop in ('TGA', 'TAG', 'TAA') then
      update hbb_mute set mute_desc = 'STOP' where hbvar_id = a and codon = b;
    end if;
    update mutation set notes = 'Y' where hbvar_id = a;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION chk_cds_single_sub()
  OWNER TO postgres;


-- Function: load_hbb_mute()

-- DROP FUNCTION load_hbb_mute();

CREATE OR REPLACE FUNCTION load_hbb_mute()
  RETURNS void AS
$BODY$
declare
  i integer;
  c text;
  h text;
  s hbb_wild%rowtype;
begin
```

```
  for i, c, h in select distinct hbvar_id, common_name, hgvs_name from
mutation
  loop
    for s in select * from hbb_wild
    loop
      insert into hbb_mute (hbvar_id, hgvs_name, common_name, seq_pos,
cds_pos, base, wild_desc, mute_desc, codon) values (i, h, c, s.seq_pos,
s.cds_pos, s.base, s.wild_desc, s.wild_desc, s.codon);
    end loop;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION load_hbb_mute()
  OWNER TO postgres;


-- Function: load_hbb_wild()

-- DROP FUNCTION load_hbb_wild();

CREATE OR REPLACE FUNCTION load_hbb_wild()
  RETURNS void AS
$BODY$
declare
  i integer;
  j integer;
  k integer;
  b text;
  s text;
  s_len integer;
  c integer;
begin

  --j := -150;    -- cds_pos
  select -sum(length(seq)) into j from raw where id in (1, 2);
  k := 1;   -- seq_pos

  -- Upstream
  i := 1;
  select seq into s from raw where id = 1;
  select length(s) into s_len from raw where id = 1;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc) values (k, j, b,
'Upstream');
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- 5' UTR
  i := 1;
  select seq into s from raw where id = 2;
  select length(s) into s_len from raw where id = 2;
  while i <= s_len loop
```

```
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc) values (k, j, b,
'5'' UTR');
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- CDS 1
  i := 1;
  j := 1;
  select seq into s from raw where id = 3;
  select length(s) into s_len from raw where id = 3;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values (k,
j, b, 'CDS 1', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- IVS 1
  i := 1;
  select seq into s from raw where id = 4;
  select length(s) into s_len from raw where id = 4;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, 'IVS 1');
    i = i + 1;
    k = k + 1;
  end loop;

  -- CDS 2
  i := 1;
  select seq into s from raw where id = 5;
  select length(s) into s_len from raw where id = 5;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values (k,
j, b, 'CDS 2', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- IVS 2
  i := 1;
  select seq into s from raw where id = 6;
  select length(s) into s_len from raw where id = 6;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, 'IVS 2');
    i = i + 1;
    k = k + 1;
  end loop;

  -- CDS 3
  i := 1;
  select seq into s from raw where id = 7;
  select length(s) into s_len from raw where id = 7;
  while i <= s_len loop
    if mod(j, 3) = 1 then
      c = (j + 2 - 3)/3;
    elsif mod(j, 3) = 2 then
      c = (j + 1 - 3)/3;
    else
      c = (j + 0 - 3)/3;
    end if;
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, cds_pos, base, wild_desc, codon) values
(k,j, b, 'CDS 3', c);
    i = i + 1;
    j = j + 1;
    k = k + 1;
  end loop;

  -- 3' UTR
  i := 1;
  select seq into s from raw where id = 8;
  select length(s) into s_len from raw where id = 8;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b, '3'' UTR');
    i = i + 1;
    k = k + 1;
  end loop;

  -- Downstream
  i := 1;
  select seq into s from raw where id = 9;
  select length(s) into s_len from raw where id = 9;
  while i <= s_len loop
    select into b substring(s from i for 1);
    insert into hbb_wild (seq_pos, base, wild_desc) values (k, b,
'Downstream');
    i = i + 1;
    k = k + 1;
  end loop;
```

```
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION load_hbb_wild()
  OWNER TO postgres;


-- Function: multi_del()

-- DROP FUNCTION multi_del();

CREATE OR REPLACE FUNCTION multi_del()
  RETURNS void AS
$BODY$
declare
  i integer;
  f integer;
  t integer;
  fm text;--integer;
  tm text;--integer;
  fs integer;
  ts integer;
begin
  for i, f, fm, t, tm in select hbvar_id,
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[1],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[2],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[3],
                                (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)_([-]?\\d+)([+-]?\\d*)del([ACGT]*)$'))[4]
                 from mutation
  loop
    -- get the seq_pos of f + fm and t + tm
    if fm = '' then
      select seq_pos into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    else
      select seq_pos + fm::integer into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    end if;
    if tm = '' then
      select seq_pos into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    else
      select seq_pos + tm::integer into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    end if;

    -- delete these bases
    if fs is null then
```

```
      fs = 0;
    end if;
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos between fs and ts;

  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION multi_del()
  OWNER TO postgres;


-- Function: multi_delins()

-- DROP FUNCTION multi_delins();

CREATE OR REPLACE FUNCTION multi_delins()
  RETURNS void AS
$BODY$
declare
  i integer;
  c text;
  h text;
  f integer;
  t integer;
  fm text;--integer;
  tm text;--integer;
  fs integer;
  ts integer;
  m text;
  l integer;
  j integer;
begin
  for i, c, h, f, fm, t, tm, m in select distinct hbvar_id, common_name,
hgvs_name,
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[1],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[2],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[3],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[4],
                          (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([+-
]?\\d*)_([-]?\\d+)([+-]?\\d*)delins([ACGT]+)$'))[5]
                            from mutation
  loop
    -- get the seq_pos of f + fm and t + tm
    if fm = '' then
      select seq_pos into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    else
      select seq_pos + fm::integer into fs from hbb_mute
      where hbvar_id = i
      and cds_pos = f;
    end if;
    if tm = '' then
      select seq_pos into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    else
      select seq_pos + tm::integer into ts from hbb_mute
      where hbvar_id = i
      and cds_pos = t;
    end if;

    -- delete these bases
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos between fs and ts;

    -- update the rest of the seq_pos
    l = length(m) + (ts - fs + 1);
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > ts;

    -- insert these bases
    l = length(m);
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base,
mute_key)
      values (i, c, h, ts + j, substring(m from j for 1), 'ins');
    end loop;

  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION multi_delins()
  OWNER TO postgres;


-- Function: post_cds_single_sub()

-- DROP FUNCTION post_cds_single_sub();

CREATE OR REPLACE FUNCTION post_cds_single_sub()
  RETURNS void AS
$BODY$
declare
  i integer;
  d integer;
  p integer;
  w text;
  m text;
begin
```

```
  for i, d, w, m in select hbvar_id,
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[1],
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[2],
                            (regexp_matches(hgvs_name,
E'HBB:c\\.\\*\\+(\\d+)([ACGT])>([ACGT])'))[3]
                     from mutation
  loop
    -- find the seq_pos of the last base in CDS
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = 444;

    update hbb_mute set base = m, mute_key = 'sub'
    where hbvar_id = i
    and seq_pos = p + d
    and base = w;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION post_cds_single_sub()
  OWNER TO postgres;


-- Function: single_del()

-- DROP FUNCTION single_del();

CREATE OR REPLACE FUNCTION single_del()
  RETURNS void AS
$BODY$
declare
  i integer;
  p integer;
  s text;
  sp integer;
  w text;
begin
  for i, p, s, w in select hbvar_id,
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[1],
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[2],
                        (regexp_matches(hgvs_name, E'HBB:c\\.([-]?\\d+)([-
+]?\\d*)del([ACGT])'))[3]
                 from mutation
  loop
    -- get seq_pos of this base
    if s = '' then
      select seq_pos into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    else
```

```
      select seq_pos + s::integer into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    end if;
    -- delete this base
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and seq_pos = sp
    and base = w;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION single_del()
  OWNER TO postgres;


-- Function: single_delins()

-- DROP FUNCTION single_delins();

CREATE OR REPLACE FUNCTION single_delins()
  RETURNS void AS
$BODY$
declare
  i integer;
  f integer;
  p integer;
  l integer;
  j integer;
  m text;
  c text;
  h text;
begin
  for i, c, h, f, m in select distinct hbvar_id, common_name, hgvs_name,
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)delins([ACGT]+)'))[1],
                          (regexp_matches(hgvs_name,
E'HBB:c\\.(\\d+)delins([ACGT]+)'))[2]
                       from mutation
  loop
    -- delete
    update hbb_mute set mute_key = 'del'
    where hbvar_id = i
    and cds_pos = f;

    -- find the seq_pos of f
    select seq_pos into p
    from hbb_mute
    where hbvar_id = i
    and cds_pos = f;

    -- update the rest of the seq_pos
    l = length(m) + 1;
    update hbb_mute set seq_pos = seq_pos + l
    where hbvar_id = i
    and seq_pos > p;

    -- insert these bases
    l = length(m);
    j = 0;
    while j < l loop
      j = j + 1;
      insert into hbb_mute (hbvar_id, common_name, hgvs_name, seq_pos, base,
mute_key)
      values (i, c, h, p + j, substring(m from j for 1), 'ins');
    end loop;

  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION single_delins()
  OWNER TO postgres;


-- Function: single_sub()

-- DROP FUNCTION single_sub();

CREATE OR REPLACE FUNCTION single_sub()
  RETURNS void AS
$BODY$
declare
  i integer;
  p integer;
  s text;
  sp integer;
  w text;
  m text;
begin
  for i, p, s, w, m in select hbvar_id,
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[1],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[2],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[3],
                              (regexp_matches(hgvs_name, E'HBB:c\\.([-
]?\\d+)([+-]?\\d*)([ACGT])>([ACGT])'))[4]
                       from mutation
  loop
    -- get the seq_pos of p + s
    if s = '' then
      select seq_pos into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    else
      select seq_pos + s::integer into sp from hbb_mute
      where hbvar_id = i
      and cds_pos = p;
    end if;
    -- substitute this base
    update hbb_mute set base = m, mute_key = 'sub'
    where hbvar_id = i
    and seq_pos = sp
    and base = w;
  end loop;
end;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION single_sub()
  OWNER TO postgres;
```

Appendix D: Database Tables

```
-- Table: author

-- DROP TABLE author;

CREATE TABLE author
(
  id serial NOT NULL,
  ref_id integer,
  author text,
  order_by integer,
  researcher_id text,
  CONSTRAINT author_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE author
  OWNER TO postgres;


-- Table: comment

-- DROP TABLE comment;

CREATE TABLE comment
(
  id serial NOT NULL,
  hgvs_name text,
  hbvar_id integer,
  comment_desc text,
  CONSTRAINT comment_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE comment
  OWNER TO postgres;
```

```
-- Table: ethnicity

-- DROP TABLE ethnicity;

CREATE TABLE ethnicity
(
  id serial NOT NULL,
  hgvs_name text,
  hbvar_id integer,
  low_freq numeric,
  high_freq numeric,
  ethnicity_desc text,
  CONSTRAINT ethnicity_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ethnicity
  OWNER TO postgres;



-- Table: hbb_mute

-- DROP TABLE hbb_mute;

CREATE TABLE hbb_mute
(
  id serial NOT NULL,
  hbvar_id integer,
  common_name text,
  hgvs_name text,
  seq_pos integer,
  cds_pos integer,
  codon integer,
  base text,
  wild_desc text,
  mute_desc text,
  mute_key text,
  CONSTRAINT hbb_mute_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE hbb_mute
  OWNER TO postgres;



-- Table: hbb_wild

-- DROP TABLE hbb_wild;

CREATE TABLE hbb_wild
```

```
(
  id serial NOT NULL,
  seq_pos integer,
  cds_pos integer,
  codon integer,
  base text,
  wild_desc text,
  CONSTRAINT hbb_wild_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE hbb_wild
  OWNER TO postgres;



-- Table: mutation

-- DROP TABLE mutation;

CREATE TABLE mutation
(
  id serial NOT NULL,
  chrom_start integer,
  chrom_end integer,
  hbvar_id integer,
  common_name text,
  hgvs_name text,
  mutation_type text,
  mutation_desc text,
  CONSTRAINT mutation_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE mutation
  OWNER TO postgres;



-- Table: "raw"

-- DROP TABLE "raw";

CREATE TABLE "raw"
(
  id serial NOT NULL,
  seq text,
  wild_desc text,
  CONSTRAINT raw_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE "raw"
  OWNER TO postgres;
```

```
-- Table: reference

-- DROP TABLE reference;

CREATE TABLE reference
(
  id serial NOT NULL,
  hbvar_id integer,
  ref_id integer,
  ref_id1 integer,
  medline_id text,
  title text,
  journal text,
  year integer,
  volume text,
  num text,
  pp text,
  CONSTRAINT reference_pkey PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE reference
  OWNER TO postgres;
```

Appendix E: Web CGI Script

```
#!C:/Perl/bin/perl.exe

use strict;
use warnings;
use DBI;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use Switch;

#
# function to print argument "\n"
#
sub println {
      print @_, "\n";
}

#
# hash that contains all the mutations
#
my %mutations = ();
my @mutations_order = ();
push(@mutations_order, '');
```

```
#
# flag that indicates the segment in the gene we are printing
#
#my $segment = "UTR";

#
# connect to PostgreSQL
#
my $dbh = DBI->connect(
      "DBI:Pg:dbname=postgres",
      "postgres",
      "mutation",
      {RaiseError => 1}) || die("Could not connect to database.");

#
# statment handle
#
my $sth = $dbh->prepare("select distinct hbvar_id, common_name from mutation
order by common_name");

#
# execute the SQL statement in the statment handle
#
$sth->execute();

#
# store each key and value returned by the statement handle in %mutation
#
while (my $ref = $sth->fetchrow_hashref()) {
      #$mutations{$ref->{'hbvar_id'}} = $ref->{'hgvs_name'};
      $mutations{$ref->{'hbvar_id'}} = $ref->{'common_name'};
      push(@mutations_order, $ref->{'hbvar_id'});
}

my $q = new CGI;

print $q->header;
print $q->start_html(
      -title => 'Beta-thalassemia Mutations',
      -style => {-src => '/style.css'});

println("<h2>Beta Thalassemia Mutations</h2><br />");

print $q->start_form(
      -name => 'main_form',
      -method => 'GET',
      -enctype => &CGI::URL_ENCODED,      # form variables show on URL
      -action => 'beta.pl'
);
print $q->popup_menu(
      -name => 'mutation',
      -values => \@mutations_order, # \ indicates passing by reference
                              # takes an array for values
                              #-default => '0',
      -labels => \%mutations        # takes a hash for visible labels
);
```

```
print $q->checkbox(
      -name => 'cds_only',
      -checked => 0,
      #-value => 1,
      -label => 'Display only CDS'
);
print("&nbsp;");
print $q->submit(
      -name => 'submit_form',
      -value => 'Submit'
);

print $q->end_form;

my $m_id = $q->param('mutation');
my $m_cds = $q->param('cds_only');
my $mutation_name = $mutations{$m_id};
if (!$m_id) {
      $m_id = '0';
      $mutation_name = '0';
}

println("<br />");
print $q->hr;

#
# function that prints one mutant sequence
#
sub print_seq {

  my $cnt = 0;
  my $codon = 0;
  my $segment = "UTR";

  $sth = $dbh->prepare("select * from hbb_mute where hbvar_id = $m_id and
(mute_desc like '%UTR' or mute_desc like 'CDS%' or mute_desc like 'IVS%')
order by seq_pos");
  $sth->execute();

  while (my $ref = $sth->fetchrow_hashref()) {
    my $base = $ref->{'base'};
    my $wild = $ref->{'wild_desc'};
    my $mute = $ref->{'mute_desc'};
    my $key = $ref->{'mute_key'};
    my $class = '';

    switch($mute) {
      # CDS
      case /^CDS/ {
        if ($segment ne "CDS") {    # check segment to print a space
          print("<span class='space'>&nbsp;</span>");
          $segment = "CDS";
        }
        if ($cnt eq 3) {            # check codon to print a space
          $cnt = 0;
          $codon = $codon + 1;
          print("<span class='space'>&nbsp;</span>");
        }
      $cnt = $cnt + 1;        # increment $cnt for codon
      switch ($key) {
            case ['sub', 'ins'] { $class = $key; }
            case 'del' { $class = $key; $cnt = $cnt - 1; }
            else { $class = 'cds'; }
      }
        print("<a href='#' title='codon $codon'><span
class=$class>$base</span></a>");
      }
      # IVS
      case /^IVS/ {
        if ($segment ne "IVS") {    # check segment to print a space
          print("<span class='space'>&nbsp;</span>");
          $segment = "IVS";
        }
        $base = lc($base);
      switch ($key) {
            case ['sub', 'ins', 'del', 'CSS'] { $class = $key; }
            else { $class = 'ivs'; }
      }
        print("<span class=$class>$base</span>");
      }
      # UTR
      case /UTR$/ {
        if ($segment ne "UTR") {    # check segment to print a space
          print("<span class='space'>&nbsp;</span>");
          $segment = "UTR";
        }
        $base = lc($base);
      switch ($key) {
            case ['sub', 'ins', 'del'] { $class = $key; }
            else { $class = 'utr'; }
      }
        print("<span class=$class>$base</span>");
      }
      # Gone not showing at the moment
      else {
        if ($segment ne "Gone") {   # check segment to print a space
          print("<span class='space'>&nbsp;</span>");
          $segment = "Gone";
        }
        $base = lc($base);
      $class = 'gone';
        print("<span class=$class>$base</span>");
      }
    }

  }
  println("<br />");

}

#
# function that prints one mutant sequence's CDS only
#
sub print_cds {
  my $cnt = 0;
  my $codon = 0;

  $sth = $dbh->prepare("select * from hbb_mute where hbvar_id = $m_id and
mute_desc like 'CDS%' order by seq_pos");
  $sth->execute();

  while (my $ref = $sth->fetchrow_hashref()) {
    my $base = $ref->{'base'};
    my $wild = $ref->{'wild_desc'};
    my $mute = $ref->{'mute_desc'};
    my $key = $ref->{'mute_key'};

    if ($cnt eq 3) {
      $cnt = 0;
      $codon = $codon + 1;
      print("<span class='space'>&nbsp;</span>");
    }
    $cnt = $cnt + 1;

        if ($key) {                 # check $key to determine CSS class
          if ($key eq 'sub' || $key eq 'ins') {
            print("<a href='#' title='codon $codon'><span
class='key'>$base</span></a>");
          } elsif ($key eq 'del') {
            print("<span class='del'>$base</span>");
            $cnt = $cnt - 1;
          } elsif ($key eq 'CSS') {
            print("<span class='cryptic'>$base</span>");
          } else {
            print("<a href='#' title='codon $codon'><span
class='cds'>$base</span></a>");
          }
        } else {
          print("<a href='#' title='codon $codon'><span
class='cds'>$base</span></a>");
        }
  }
  println("<br />");

}

if ($m_cds) {
      print_cds();
} else {
      print_seq();
}
print $q->hr;

#
# function that prints comment
#
$sth = $dbh->prepare("select * from comment where hbvar_id = $m_id");
$sth->execute();
println("<table>");
println("<tr>");
println("<th>");
println("Comment");
println("</th>");
println("</tr>");
println("<tr>");
println("<td>");
while (my $ref = $sth->fetchrow_hashref()) {
  println($ref->{'comment_desc'});
}
println("</td>");
println("</tr>");
println("</table>");
print $q->hr;

#
# funtion that prints ethnicity
#
$sth = $dbh->prepare("select * from ethnicity where hbvar_id = $m_id");
$sth->execute();
println("<table>");
println("<tr>");
println("<th colspan=2>");
println("Ethnicity Distribution");
println("</th>");
println("</tr>");
while (my $ref = $sth->fetchrow_hashref()) {
  println("<tr>");
  println("<td>");
  println($ref->{'ethnicity_desc'});
  println("</td>");
  println("<td>");
  if ($ref->{'high_freq'}) {
    println($ref->{'high_freq'});
    println("%");
  }
  println("</td>");
  println("</tr>");
}
println("</table>");
print $q->hr;

#
# funtion that prints reference
#
$sth = $dbh->prepare("select distinct r.id, title, journal, year, author,
volume, num, pp, order_by from reference r, author a where r.ref_id =
a.ref_id and hbvar_id = $m_id order by r.id, a.order_by");
$sth->execute();
my $author = '';
my $title = '';
my $journal = '';
my $volume = '';
my $year = '';
my $num = '';
my $pp = '';
println("<table>");
println("<tr>");
println("<th>");
println("Reference");
println("</th>");
println("</tr>");
while (my $ref = $sth->fetchrow_hashref()) {
  if ($title eq '') {
    $author = $ref->{'author'};
  } elsif ($title ne $ref->{'title'}) {
    println("<tr>");
    println("<td>");
    println("<span class='hanging_indent'>$author, ($year). $title
<I>$journal, $volume</I>($num), $pp</span>");
    println("</td>");
    println("</tr>");
    $author = $ref->{'author'};
  } else {
    $author = $author . ', ' . $ref->{'author'};
  }
  $title = $ref->{'title'};
  $journal = $ref->{'journal'};
  $volume = $ref->{'volume'};
  $year = $ref->{'year'};
  $num = $ref->{'num'};
  $pp = $ref->{'pp'};
}
if ($title ne '') {
  println("<tr>");
  println("<td>");
  println("<span class='hanging_indent'>$author, ($year). $title <I>$journal,
$volume</I>($num), $pp</span>");
  println("</td>");
  println("</tr>");
}
println("</table>");

print $q->hr;

println("<br />");
println("<h4>Disclaimer: Data on this website were provided generously by the
<a href='http://globin.bx.psu.edu/hbvar/menu.html'>HbVar database</a> at
Pennsylvania State University.</h4><br />");

print $q->end_html;
$dbh->disconnect();
exit;
```

Appendix F: Web CSS Script

```
body {
      width: 960px;
      font-family: Georgia, "Times New Roman", Serif;
}
table {
      margin-left: auto;
      margin-right: auto;
      font-family: Georgia, "Times New Roman", Serif;
}
h2, h4      {
      font-family: Georgia, "Times New Roman", Serif;
}
span {      /* enable line wrap */
      display: inline-block;
}
.utr {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: blueviolet;
      font-weight: bold;
}
.cds {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: blue;
      font-weight: bold;
}
.ivs {
      font-family: "Lucida Console", Courier, monospace;
      font-size: x-large;
      color: darkseagreen;
      font-weight: bold;
}
.gone {
      font-family: "Lucida Console", Courier, monospace;
      font-size: x-large;
      color: silver;
      font-weight: bold;
}
.CSS {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: darkseagreen;
      background-color: yellow;
      font-weight: bold;
}
.cds_key {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: blue;
      background-color: red;
      font-weight: bold;
}
.ivs_key {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: darkseagreen;
      background-color: red;
      font-weight: bold;
}
```

```
.ivs_css {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: darkseagreen;
      background-color: yellow;
      font-weight: bold;
}
.utr_key {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: blueviolet;
      background-color: red;
      font-weight: bold;
}
.ins, .sub {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      color: white;
      background-color: red;
      font-weight: bold;
}
.del {
      font-family: "Lucida Console", Monaco, monospace;
      font-size: x-large;
      text-decoration: line-through;
      background-color: red;
      font-weight: bold;
}
.space {
      font-family: "Lucida Console", Courier, monospace;
      font-size: x-large;
      font-weight: bold;
}
.hanging_indent {
      padding-left: 50px;
      text-indent: -50px;
}
```

Bibliography

About PERL. (2012, 8 1). Retrieved from The PERL programming language: http://www.perl.org/about.html

Agouti, I. B. (2008). Molecular basis of β-thalassemia in Morocco: Possible origins of the molecular heterogeneity. *Genetic Testing*.

An Overview of Hemoglobin. (2002, 4 10). Retrieved from Information Center for Sickle Cell and Thalassemic Disorder: http://sickle.bwh.harvard.edu/hemoglobin.html

Beta Thalassemia. (2009, 7). Retrieved from Genetics Home Reference: http://ghr.nlm.nih.gov/condition/beta-thalassemia

Cascading Style Sheets. (2012, 9 9). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

Codd, E. (1970). A relational model of data for large shared databanks. *Commutations of the ACM*.

Coronel, C. M. (2011). *Database Systems Design, Implementation, and Management.* Boston: Cengage Learning.

Forget, B., & Hardison, R. (2009). Chapter 3: The Normal Structure and Regulation of Human Globin Gene Clusters. In *Disorder of Hemoglobins: Genetics, Pathophysiology, and Clinical Management.* Cambridge Publishing.

Forget, B., Martin, S., Higgs, D., Nagel, R., & Bunn, H. (2000). *Molecular mechanisms of beta thalassemia.* Cambridge: Cambridge University Press.

Garcia-Molina, H. U. (2009). *Database Systems the Complete Book.* Upper Saddle River: Pearson Prentice Hall.

Gray, J. (1981). The transaction concept: virtues and limitations. *Proceedings of Seventh International Conference on ery large Databases.* Cupertino: Tandem Computers Inc.

Higgs, D. (2004). Gene Regulation in Hematopoiesis: New Lessons from Thalassemia. *Hematology*.

Higgs, D., Engel, J., & Stamatoyannopoulos, G. (2011). Thalassemia. *The Lancet*.

HTML & CSS. (2012, 07 04). Retrieved from World Wide Web Consortium: http://www.w3.org/standards/webdesign/htmlcss.html

Knight, J. (2009). *Human Genetic Diversity.* Oxford University Press.

Linux Journal. (2001). Retrieved from Editor's Choice 2001: http://www.linuxjournal.com/article/5525

Makhoul, N. W. (2005). Genetic heterogeneity of β-thalassemia in Lebanon reflects historic and recent population migration. *Annals of Human Genetics*.

Perl CGI. (2012, 5 1). Retrieved from Perl Programming Documentation: http://perldoc.perl.org/CGI.html

Perl DBI. (2012, 5 1). Retrieved from Perl DBI: http://dbi.perl.org

Pierce, B. (2012). *Genetics: a conceptual approach.* New York: W. H. Freeman Co.

PostgreSQL. (2012, 5 1). Retrieved from PostgreSQL: http://www.postgresql.org

Programming the Perl DBI. (2012, 5 1). Retrieved from Oreiley: http://oreilly.com/catalog/perldbi/chapter/ch04.html

Richardson, M. (1999, 5 1). *Larry Wall - the PERL guru*. Retrieved from Linux Journal: http://www.linuxjournal.com/article/3394

RNA splcing. (2012, 7 4). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/RNA_splicing

St. Clair, C. V. (2009). *Exploring Bioinformatics.* Sudbury: Jones and Bartlett Publishers.

Treisman, R., Orkin, S., & Maniatis, T. (1983). Specific Transcription and RNA Splicing Defects in Five Cloned Beta-Thalassemia Genes. *Nature*.

Vichinsky, E., MacKlin, E., Waye, J., Lorey, F., & Olivieri, N. (2005). Changes in the Epidemiology of Thalassemia in North America: A New Minority. *Pediatrics*.