The handle http://hdl.handle.net/1887/45045 holds various files of this Leiden University dissertation.

**Author:** Vis, J.K.  
**Title:** Algorithms for the description of molecular sequences  
**Issue Date:** 2016-12-21
Chapter 4

Extraction of HGVS Variant Descriptions from Protein Sequences

Frameshift variants are an important class of variants when considering protein sequences. Small deletions/insertions in DNA sequences may result in (large) frameshifts thereby large changes in the protein sequence and its function. We propose a method of finding and annotating frameshift variants, in the spirit of the standard nomenclature of the Human Genome Variation Society (HGVS), from two protein sequences without considering the underlying DNA sequences. Our method is able to efficiently compute HGVS descriptions for protein sequences with frameshift annotations for all species codon table. Furthermore, we show that this method can be effectively used to find promising novel evolutional events. We propose an addition to the HGVS nomenclature for accommodating the (complex) frameshift variants that can be described with our method. Our method is available in the Description Extractor package for Mutalyzer: https://pypi.python.org/pypi/description-extractor. The C++ source code and Python interface are accessible at: https://github.com/mutalyzer/description-extractor.
Chapter 4. HGVS Protein Descriptions

4.1 Introduction

The Human Genome Variation Society publishes nomenclature guidelines [den Dunnen et al., 2000] for unambiguous sequence variant descriptions used in clinical reports, literature and genetic databases. To check and interpret these descriptions the Mutalyzer program suite [Wildeman et al., 2008] has been built with as main purpose the automatic checking disambiguation and correction of variant descriptions. In [Vis et al., 2015] an efficient algorithm for automated DNA/RNA variant description extraction is presented. Here, we propose an extension of this algorithm which provides the automated description extraction of protein sequences without direct knowledge of the underlying DNA/RNA sequences. In particular, it provides frameshift variant detection. A frameshift variant is introduced by the insertion or deletion in a coding sequence which length is not divisible by three. In addition, we can annotate protein sequences formed by an inversion and shifted variants of inversions as well.

A frameshift variant is a genetic mutation due to insertions or deletions on a DNA sequence that is consequently translated into a protein by encoding each triplet of nucleotides into an amino acid. The key to introducing frameshift lies in this triplet-based structure also known as the reading frame. Any insertion of deletion with a length not divisible by three introduces a huge effect on the protein level; all (or most) of the following amino acids will be different from the unmodified ones. In this chapter we do not consider the transcription process (DNA to RNA) nor the splicing process. The term DNA level means the coding region of the DNA for a particular protein.

As protein descriptions are supported by HGVS, there is a need for their automatic construction. A naive approach could be the HGVS description extractor from [Vis et al., 2015] on the DNA level and convert its output to the protein level. Unfortunately this approach is not feasible. First, many different variants on the DNA level could lead to the same variant on the protein level. There should be a guarantee that all of these DNA level variants are indeed converted to the same protein variant. Given the complexity of these variants it is far from trivial (maybe even impossible) to give such a guarantee. Second,
4.1. Introduction

sometimes the DNA level is not accessible or unknown. In this case, a protein level description would still be useful. Thirdly, considering only the protein level gives us the possibility of allowing for useful annotation in terms of frameshifts.

Usually reading frame modifications are analyzed on the DNA level [Sheetlin et al., 2014], however, we focus here on frameshift detection on the protein level. Our algorithm is closely related to the methods described in [Gîrdea et al., 2010]. There, a graph-based approach is used to perform back-translation to the DNA level and induce frameshift mutation from there. Although theoretically our approach is the same, we allow for more types of frameshifts and we simplify the frameshift detection calculation by precalculation of look-up tables such that the frameshift detection can be performed directly on the protein level. Furthermore, the alignment step is not part of the frameshift detection algorithm, but rather performed in advance. In our case the frameshift detection can be seen as a post-processing step on deletion/insertion variants. These optimizations result in an efficient algorithm.

4.1.1 Frameshift variants

We extend the traditional definition of frameshift mutations to cater for more general frameshift variants. Whereas in the traditional definition only two types of frameshift are recognized, i.e., $+1$ and $+2$, we allow also for frameshifts in combination with inversions on the DNA level. Indeed, the reverse complement can be regarded as a frame modification in the protein. Furthermore, the reverse complement can be combined with the usual $+1$ and $+2$ frameshifts bringing the total number of types of frameshifts to five, see Table 4.1.

The $+1$ and $+2$ frameshifts are caused by insertions or deletions with a length modulo $3 \neq 0$. We have more possibilities for the inverse types; the symmetry around the frame boundaries is important. For pure inversions the “overhang”, i.e., the number of bases partially covering a codon, of the frame boundaries should be equal. When the overhang is symmetrical the inversed inserted sequence preserves the original frame boundaries. Unequal overhangs result in inverse $+1$ or $+2$ types (see Table 4.1), e.g., if the overhang on the right hand
Table 4.1: The five types of frameshifts with examples on the DNA sequence and the resulting protein sequence. AB026906.1 is used as a reference sequence.

<table>
<thead>
<tr>
<th>Type</th>
<th>DNA Variant</th>
<th>Protein Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>274_275insC</td>
<td>...MALFPGCSPHSSWSLGWTSCY*</td>
</tr>
<tr>
<td></td>
<td>274_275del</td>
<td>...MLFPGCSPHSSWSLGWTSCY*</td>
</tr>
<tr>
<td>+2</td>
<td>274_275insCC</td>
<td>...MATIPWLQPSLVMVTDKLLLT...</td>
</tr>
<tr>
<td></td>
<td>274del</td>
<td>...MTIPWLQPSLVMVTDKLLLT...</td>
</tr>
<tr>
<td>inverse</td>
<td>274_309inv</td>
<td>...MTMKSEGCSQGIVHWGLGQVTD...</td>
</tr>
<tr>
<td>inverse +1</td>
<td>275_309inv</td>
<td>...MDHEE*</td>
</tr>
<tr>
<td></td>
<td>272_309inv</td>
<td>...NHEE*</td>
</tr>
<tr>
<td>inverse +2</td>
<td>276_309inv</td>
<td>...MEP*</td>
</tr>
<tr>
<td></td>
<td>273_309inv</td>
<td>...IP*</td>
</tr>
</tbody>
</table>

side is larger, the inverted sequence is also shifted to the right and vice versa for a left hand side overhang.

### 4.1.2 Complex frameshift variants

Traditionally when a frameshift is detected the remainder of the protein sequence is not annotated any further. The assumption being that the resulting protein will differ substantially from its reference. However, combinations of variants on the DNA level can alter the frameshift into an other type of frameshift. They can even restore the original reading frame leaving just a part of the protein sequence modified. For instance consider the following combination of DNA level variants: AB026906.1.c.[274del;288del;301del].
With the corresponding protein sequences:

\[ R = \text{MDYSLAAALTLHGH} \]
\[ S = \text{MTIPWRSPHF} - \text{HGH} \]

The first deletion will induce a +2 frameshift variant. This frameshift is later modified by the second deletion into a +1 frameshift. The third deletion restores the original reading frame. We propose that the aforementioned example is annotated as:

\[ 2\_7|2\;7\_11|1 \]

The remainder of this chapter is organized as follows. In Section 4.2 we formally introduce the problem of frameshift detection and we introduce an algorithm to efficiently compute frameshift variants between two protein sequences. Section 4.3 contains the experiments, followed by a discussion of additional uses and properties in Section 4.4 and the conclusions in Section 4.5.

### 4.2 Methods

Based on the extraction algorithm introduced in Chapter 3, we added support for frameshift variant detection by performing the extraction algorithm in two phases. The first phase follows the original algorithm: given two strings \( R \) and \( S \) find the longest string that is a substring of \( R \) as well as \( S \). Remove this string from the problem, and continue recursively with both prefixes \( R_{\text{pre}} \) and \( S_{\text{pre}} \) and both suffixes \( R_{\text{suf}} \) and \( S_{\text{suf}} \). The recursion ends when either of the two strings is empty or no common substring could be found. For protein sequences we only use the traditional edit operations: deletion, insertion and substitution. Transpositions are not considered. Due to the relative short length of protein sequences there is no benefit in using the LCS\(_k\) algorithm.

In the second phase, the regions of change (deletions/insertions) are recursively partitioned into frame shifted regions and regions that cannot be described in terms of frameshifts. We adapted the LCS algorithm from finding exact string matches to match all possible (combinations) of frameshifts, again using a greedy approach. Note that we can only add the frameshift variants as
annotation; not as a true variant without violating the property of unambiguity. Without the exact amino acid sequence we cannot reconstruct the observed sequence from the reference given its variants.

Formally the problem can be defined as follows. Let \( \Sigma_N = \{A, C, G, T\} \) be the nucleotide alphabet and the amino acid alphabet \( \Sigma_A = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y, \ast\} \). A triplet or _codon_ \( c = \langle b_1, b_2, b_3 \rangle \), where \( b_1, b_2, b_3 \in \Sigma_N \), is mapped to an amino acid (from \( \Sigma_A \)) by the surjective _translation_ function \( f : \Sigma_N \times \Sigma_N \times \Sigma_N \rightarrow \Sigma_A \). An example of the translation function is given in Table 4.2. In general, the codon table can be different for (applications within) different species as well as within species. For the human genome the codon table in Table 4.2 is commonly used and shall be used as an example further in this chapter.

**Table 4.2: An inverse DNA codon table for the human genome.**

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>Codon(s)</th>
<th>Amino Acid</th>
<th>Codon(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>GCA, GCC, GCG, GCT</td>
<td>N</td>
<td>AAC, AAT</td>
</tr>
<tr>
<td>C</td>
<td>TGC, TGT</td>
<td>P</td>
<td>CCA, CCC, CGG, CCT</td>
</tr>
<tr>
<td>D</td>
<td>GAC, GAT</td>
<td>Q</td>
<td>CAA, CAG</td>
</tr>
<tr>
<td>E</td>
<td>GAA, GAG</td>
<td>R</td>
<td>AGA, AGG, CGA, CGC, CGG, CGT</td>
</tr>
<tr>
<td>F</td>
<td>TTC, TTT</td>
<td>S</td>
<td>AGC, AGT, TCA, TCC, TCG, TCT</td>
</tr>
<tr>
<td>G</td>
<td>GGA, GGC, GGG, GCT</td>
<td>T</td>
<td>ACA, ACC, AGG, ACG, ACT</td>
</tr>
<tr>
<td>H</td>
<td>CAC, CAT</td>
<td>V</td>
<td>GTA, GTC, GTG, GTT</td>
</tr>
<tr>
<td>I</td>
<td>ATA, ATC, ATT</td>
<td>W</td>
<td>TGG</td>
</tr>
<tr>
<td>K</td>
<td>AAA, AAG</td>
<td>Y</td>
<td>TAC, TAT</td>
</tr>
<tr>
<td>L</td>
<td>CTA, CTC, CTT, CTG, TTG</td>
<td>* (stop)</td>
<td>TAA, TAG, TGA</td>
</tr>
<tr>
<td>N (start)</td>
<td>ATG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now we define the frameshift functions:

\[
fs_{+1}(c_1, c_2, c_s) = \begin{cases} 
\text{true} & \text{if } \langle b_{13}, b_{21}, b_{22} \rangle = c_s, \text{ with } b_{1i} \in c_1 \land b_{2j} \in c_2 \\
\text{false} & \text{otherwise}
\end{cases}
\]

\[
\hat{fs}_{+1}(a_1, a_2, a_s) = \bigvee_{\forall c_1 \in f^{-1}_{a_1}} \bigvee_{\forall c_2 \in f^{-1}_{a_2}} \bigvee_{\forall c_s \in f^{-1}_{a_s}} fs_{+1}(c_1, c_2, c_s).
\]

The functions for +2 frameshifts are analogous to \( fs_{+1} \) and \( \hat{fs}_{+1} \).
4.2. Methods

\[ f_{\text{inv}}(c, c_s) = \begin{cases} 
\text{true} & \text{if } \langle \text{inv}(b_3), \text{inv}(b_2), \text{inv}(b_1) \rangle = c_s, \text{ with } b_i \in c \\
\text{false} & \text{otherwise} 
\end{cases} \]

where \( \text{inv} \) is the complement of a nucleotide, i.e., \( A \Leftrightarrow T \) and \( C \Leftrightarrow G \).

\[ \hat{f}_{\text{inv}}(a, a_s) = \bigvee_{c \in f^{-1}_a} f_{\text{inv}}(c, c_s) \bigvee_{c_s \in f^{-1}_a} f_{\text{inv}}(c_s, c) \]

In Figure 4.1, an example frameshift calculation is given.

\[ R = \text{MDYSLAAALTLHGH} \]
\[ S = \text{MTIPWRSPHF-HGH} \]

\[ \begin{array}{cccc}
D & Y & +2 \\
GAC & TAC & G & ACT & AC \\
GAC & TAT & G & ACT & AT \\
GAT & TAC & G & ATT & AC \\
GAT & TAT & G & ATT & AT \\
\end{array} \]

\[ T = \{ \text{ACA, ACC, ACG, ACT} \} \]

\[ f_{s+2}(\text{GAC, TAC, ACT}) = \text{true} \rightarrow \hat{f}_{s+2}(D, Y, T) = \text{true} \]

Figure 4.1: Example calculation of a +2 frameshift. This example can be continued by calculating \( \hat{f}_{s+2}(Y, S, I) \), etc.

The functions for the inverse frameshifts are analogous to \( f_{s+1} \) and \( \hat{f}_{s+1} \) with the inverse codon reordering analogous to \( f_{\text{inv}} \).

The results of all frameshift functions can be precomputed and stored in a look-up table. Moreover, all types of frameshift can be simultaneously checked by encoding the five types as a bit array. We use a \( |\Sigma_A| \times |\Sigma_A| \times |\Sigma_A| \) look-up table corresponding to the signature of the \( \hat{f}_s \) functions. Although the signature
of the inverse frameshift functions is different, we can still use the same look-up table. This renders a frameshift check equivalent to character matching in string comparison. As a consequence frameshift detection is efficient. Note that frameshift types can be overlapping for a certain combination of amino acids, e.g., P can be found as either frameshift +1 or +2 in SL.

For each deletion/insertion a frameshift extraction is started based on the aforementioned frameshift functions. Again we follow a greedy approach: we select the longest same-type frameshift. In the event of a multi-type frameshift it is annotated as such. The frameshift extraction is then continued on the remaining prefixes and suffixes of the original deletion/insertions until one of the prefixes of suffixes is empty, much like the original extraction routine. The possibility of having multi-type frameshifts introduces an uncertainty of the actual frameshift type boundary when considering a complex frameshift variant. When we consider a variant where a frameshift +2 is altered to a frameshift +1 for instance by deleting a single nucleotide on the DNA level we cannot accurately predict the actual frameshift type boundary when some of the amino acids show a multi-type frameshift, i.e., +2 and +1. The longest part of the complex frameshift will be dominant in this case by including as much amino acids as possible. As we assume no knowledge of the DNA level this is optimal in this situation.

4.2.1 Probability calculation

To add to the frameshift annotation we can estimate the probability of the frameshift by using information from the codon table as well as additional information about the distribution of codon usage in a species. When this additional information is absent, we can approximate the probability by using the distribution of the amino acids and assume a uniform probability distribution for each amino acid over its codons. If this information is also absent, we can still approximate the probability by assuming a uniform distribution of amino acids. This calculation can be refined by incorporating different distributions when comparing two different species. In general, the most simple approximation, i.e., no extra information (only using the codon table), will yield a usable
approximation of the frameshift probability:

\[ 1 - \frac{1}{\Sigma A^\ell} \]

where \( \ell \) is the length of the frameshift. As is apparent, the probability will increase strongly with the length of the frameshift.

Next, we consider an example LR with frameshift +2. We take from Table 4.2 \( L = \{ \text{CTA, CTC, CTG, CTT, TTA, TTG} \} \) and \( R = \{ \text{AGA, AGG, CGA, CGC, CGG, CGT} \} \) and we assume a uniform distribution over the codons. The probability calculation follows the original frameshift calculation quite closely. The following frameshifted codons are observed:

- \( 8 \times \text{TAC (Y)} \)
- \( 8 \times \text{TGC (C)} \)
- \( 4 \times \text{TAA (*)} \)
- \( 4 \times \text{TCC (S)} \)
- \( 4 \times \text{TGA (*)} \)
- \( 4 \times \text{TTC (F)} \)
- \( 2 \times \text{TCA (S)} \)
- \( 2 \times \text{TTA (L)} \)

This corresponds to the following probabilities \( P(c_s|c_1c_2) \):

- \( P(\text{C}|LR) = \frac{8}{64} \)
- \( P(\text{Y}|LR) = \frac{8}{64} \)
- \( P(\text{*}|LR) = \frac{4+4}{64} \)
- \( P(\text{S}|LR) = \frac{4+2}{64} \)
- \( P(\text{F}|LR) = \frac{4}{64} \)
- \( P(\text{L}|LR) = \frac{2}{64} \)

The probability of a frameshifted sequence is the product of the individual probabilities. In the event of a multi-type frameshift the combined probabilities can be precomputed. When we consider the aforementioned example with added codon distributions \( P(c) \):
\[ L = \begin{cases} 
CTA & 0.22 \\
CTC & 0.20 \\
CTG & 0.28 \\
CTT & 0.15 \\
TTA & 0.05 \\
TTG & 0.10 
\end{cases} \quad R = \begin{cases} 
AGA & 0.10 \\
AGG & 0.05 \\
CGA & 0.20 \\
CGC & 0.22 \\
CGG & 0.15 \\
CGT & 0.28 
\end{cases} \]

We can calculate the frameshift +2 probability:

\[ P(c_s|a_1a_2) = \sum_{\forall c_1 \in f^{-1}a_1 \text{ where } b_{12}=b_{13}=b_{22}} P(c_1) \cdot \sum_{\forall c_2 \in f^{-1}a_2 \text{ where } b_{21}=b_{23}} P(c_2) \]

Again we observe the following codons:

\[
\begin{align*}
8 \times \text{TAC (Y)} & \quad (0.22 + 0.05)(0.2 + 0.22 + 0.15 + 0.28) \approx 0.23 \\
8 \times \text{TGC (C)} & \quad (0.28 + 0.1)(0.2 + 0.22 + 0.15 + 0.28) \approx 0.32 \\
4 \times \text{TAA (*)} & \quad (0.22 + 0.05)(0.1 + 0.05) \approx 0.04 \\
4 \times \text{TCC (S)} & \quad 0.2(0.2 + 0.22 + 0.15 + 0.28) \approx 0.17 \\
4 \times \text{TGA (*)} & \quad (0.28 + 0.1)(0.1 + 0.05) \approx 0.06 \\
4 \times \text{TTC (F)} & \quad (0.05 + 0.1)(0.2 + 0.22 + 0.15 + 0.28) \approx 0.13 \\
2 \times \text{TCA (S)} & \quad 0.2(0.1 + 0.05) \approx 0.03 \\
2 \times \text{TTA (L)} & \quad 0.15(0.1 + 0.05) \approx 0.02
\end{align*}
\]

With the resulting probabilities on the protein level:

\[ P(a_s|a_1a_2) = \sum_{\forall c_s \in f^{-1}a_s} P(c_s|a_1a_2) \]

Given below:

\[
\begin{align*}
P(C|LR) & \approx 0.32 \\
P(Y|LR) & \approx 0.23 \\
P(*)|LR) & \approx 0.04 + 0.06 \approx 0.11 \\
P(S|LR) & \approx 0.17 + 0.03 \approx 0.20 \\
P(F|LR) & \approx 0.13 \\
P(L|LR) & \approx 0.02
\end{align*}
\]
The afore mentioned example is quite extreme in its given (fictitious) codon distribution. Real-life examples are usually much closer to the uniformly distributed estimations.

As before with the frameshift loop-up table, we can precalculate the probability loop-up table as well. This results in an efficient calculation of the frameshift annotations. For implementations using floating point arithmetic one should be aware of the small numbers that result from this calculation and its inherently corresponding errors. In our implementation we prefer fixed point calculation using binary scaling.

### 4.2.2 Back-translation

The technique we have employed to calculate frameshifts and its probabilities can also be used for accurately predicting the DNA sequences underlying the protein. The process of translating amino acid sequences to DNA sequences is known as back-translation. In general, the surjective codon function makes it difficult to accurately predict the underlying DNA sequence, i.e., there is a lot of ambiguity resulting in an explosion of the possible DNA sequences. Consider the protein sequence DYSLA, with no further information we can translate this sequence to its underlying DNA: GAYTAYWSNYTNGCN, where we used the degenerate form to encode the nucleotide variation. When we count the number of DNA sequences they amount to $2,048$.

When we know (or we just calculated) that the protein sequence DYSLA is actually a frameshift $+2$ with the sequence TIPWR, we can, for both sequences, construct a much more accurate back-translation: GACTATCCYTGGCG, with only two possible DNA sequences and ACTATWCCYTGGCGG, with four DNA sequences. The possibilities for each nucleotide are considerably reduced by the restriction of the partially overlapping codons.

### 4.3 Experiments

We performed computer experiments to demonstrate the performance of our algorithm and its corresponding annotation. All experiments involve multiple
genes that are pair-wise analyzed for frameshift overlap. We use the description extraction algorithm from Chapter 3, but we report only on frameshift variants. In all experiments where multiple genes are pair-wise analyzed we use Bonferroni correction for multiple testing with the significance threshold:

\[ 1 - \frac{\alpha}{m}, \]

where \( m \) is the number of tests that are performed and \( \alpha \) is fixed at 0.05. This gives us a conservative bound on the statistical significance. For a pair-wise analysis with \( n \) genes we typically do \( n^2 - n \) tests. Although the Bonferroni correction is too restrictive, we consider a full analysis of multiple testing on this data beyond the scope of this chapter.

4.3.1 Intra-species frameshifts in \( E. \ coli \) K-12

In this experiment we analyze 4,305 genes\(^1\) pair-wise of \( E. \ coli \) K-12 taken from Uniprot [UniProt Consortium, 2015]. We are interested in the self overlap (in terms of frameshifts) of this species. For this species we obtained a codon frequency table from [Nakamura et al., 2000] and we used this additional information for the probability calculation as described in Section 4.2.1. After Bonferroni correction, we selected every pair that contained at least one statistically significant frameshift variant.

The average gene length (on the protein level) is approximately 315 amino acids. The total computation time was around 48 hours on a desktop PC (3.4 GHz and 16 GB RAM) using a single thread. Which results in an average extraction time of 0.01 per pair of genes. This process could trivially be parallelized.

Out of the 18,528,720 pair-wise tests, 245 pairs had a significant frameshift overlap. Within these pairs 359 unique proteins are present. In principle the frameshift overlap property is symmetrical, however, it is often possible to extend a frameshift with one amino acid in one of the directions. In such a case the added amino acid has the same frameshift overlap by chance. Often we see symmetrical frameshift overlaps with a length difference of one. For

\(^1\)Not all of the annotated protein sequences encode for a gene.
large frameshifts this is usually not a problem in terms of its significance, but smaller frameshifts can be selected as being significant by adding one amino acid. This accounts for the odd number of significant pairs as well as the larger than expected number of unique proteins. In Table 4.3 the top 10 frameshift overlaps are given.

Table 4.3: The 10 most significant frameshift overlaps in *E. coli* K-12. Only unique pairs are given. For all these pairs the corresponding symmetrical overlap is also present and significant. The calculated numerical significance is 1 for all of these pairs.

<table>
<thead>
<tr>
<th>Protein</th>
<th>Protein</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>P75617_YAAW</td>
<td>P28697_HTGA</td>
<td>inverse +2</td>
<td>195</td>
</tr>
<tr>
<td>Q9Z3A0_YJGW</td>
<td>P39349_YJGX</td>
<td>inverse</td>
<td>84</td>
</tr>
<tr>
<td>P0AE48_YTFP</td>
<td>P08339_Y4223</td>
<td>inverse +1</td>
<td>84</td>
</tr>
<tr>
<td>P76158_RZPQ</td>
<td>C1P601_RZOQ</td>
<td>+2</td>
<td>83</td>
</tr>
<tr>
<td>P27838_CYAY</td>
<td>P11291_Y3808</td>
<td>inverse</td>
<td>68</td>
</tr>
<tr>
<td>P77551_RZPR</td>
<td>P58042_RZOR</td>
<td>+2</td>
<td>61</td>
</tr>
<tr>
<td>P75719_RZPD</td>
<td>P58041_RZOD</td>
<td>+2</td>
<td>59</td>
</tr>
<tr>
<td>P75712_ALLP</td>
<td>P75711_YBBBV</td>
<td>+2</td>
<td>45</td>
</tr>
<tr>
<td>Q47536_YAIP</td>
<td>P75697_YAIX</td>
<td>+2</td>
<td>42</td>
</tr>
<tr>
<td>P76066_YDAW</td>
<td>P77551_RZPR</td>
<td>+1</td>
<td>36</td>
</tr>
</tbody>
</table>

The longest significant result in Table 4.3, the pair P75617_YAAW and P28697_HTGA, is described as an overprinting event in [Delaye et al., 2008]. In this case, the protein P28697_HTGA forms a complete frameshift overlap with part of the P75617_YAAW protein. Another striking observation is the relative long lengths of the frameshift overlaps especially when taking into account the corresponding DNA sequences, given the expected length of longest common substring between two random strings [Abbasi, 1997]. The technique for finding frameshifts can be applied to initiate further research like [Delaye et al., 2008] aimed towards finding evidence for the evolution of novel genes. The names containing RZ seem to be somewhat related genes, however, although
the selected pairs show a strong significance, no other combinations have a significant frameshift overlap, suggesting that frameshift variants are an important evolutionary mechanism. Finally, from the data from the complete experiment (data not shown in Table 4.3), we observe that almost all pairs of proteins show a longer than expected frameshift overlap; around 15 amino acids on average.

The significant proteins were used to check whether any clustering of frameshift overlaps could be found. We calculated a $359 \times 359$ distance matrix, where distance is defined as $d(r, s) = r_{fs}/\max(|r|, |s|)$, with $r_{fs}$ the length of the longest frameshift variant. It turns out that almost all of the frameshift overlaps occur in pairs. And, in contrast to what the RZ proteins in Table 4.3 might suggest, there are no clusters to be found.

### 4.3.2 Inter-species frameshifts between *E. coli K-12* and *S. enterica*

The results from the experiment in Section 4.3.1 show a strong evidence for intra-species frameshift overlaps. In this experiment we investigate whether the same evidence can be found inter-species. We use the same data for *E. coli* as in Section 4.3.1 and we added 4,533 protein sequences from *S. enterica* taken from Uniprot [UniProt Consortium, 2015]. Instead of a full pair-wise comparison we calculated the maximum frameshift overlap for each of the 4,305 *E. coli* proteins with any of the 4,533 *S. enterica* proteins. For the frameshift probability calculation we used the same codon probabilities for *E. coli* as in Section 4.3.1. No such data was available for *S. enterica*, so we used a slightly less accurate probability calculation based on the probabilities of the occurrence of the amino acids (taken from the actual protein sequences). The top 10 frameshift overlaps are given in Table 4.4. The significance of the frameshift overlaps is given as $1 - P$, where $P$ is the probability of the frameshift.

Although the results are not as striking as in Section 4.3.1, there are still significant inter-species frameshift overlaps. Again, this method can be used to identify promising candidates for further evolitional research.

From the data from the complete experiment (data not shown in Table 4.4),
Table 4.4: The 10 most significant frameshift overlaps between *E. coli* K-12 and *S. enterica*. Only unique pairs are given. For all these pairs the corresponding symmetrical overlap is also present and significant. The significance is given as $1 - P$ in order to show some meaningful value.

<table>
<thead>
<tr>
<th>E. coli</th>
<th>S. enterica</th>
<th>Type</th>
<th>Significance</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>P08339_Y4223</td>
<td>Q7CP87_Q7CP87</td>
<td>inverse</td>
<td>+1</td>
<td>1.06e–42</td>
</tr>
<tr>
<td>P75712_ALLP</td>
<td>Q8ZR81_Q8ZR81</td>
<td>+2</td>
<td>5.83e–30</td>
<td>34</td>
</tr>
<tr>
<td>P0CF79_INSF1</td>
<td>Q8ZRJ4_Q8ZRJ4</td>
<td>inverse</td>
<td>6.75e–27</td>
<td>30</td>
</tr>
<tr>
<td>P11291_Y3808</td>
<td>P56978_CYAY</td>
<td>inverse</td>
<td>3.69e–24</td>
<td>28</td>
</tr>
<tr>
<td>Q79CP2_YGIA</td>
<td>Q7CPS1_YGIB</td>
<td>+2</td>
<td>6.62e–21</td>
<td>27</td>
</tr>
<tr>
<td>P0ADP5_YIGM</td>
<td>P0A2Q4_MESALTY</td>
<td>inverse</td>
<td>+1</td>
<td>3.12e–19</td>
</tr>
<tr>
<td>P0ACW2_YDBJ</td>
<td>Q8ZP90_Q8ZP90</td>
<td>inverse</td>
<td>+1</td>
<td>3.15e–19</td>
</tr>
<tr>
<td>P0AC96_GNTU</td>
<td>Q8ZLG4_Q8ZLG4</td>
<td>inverse</td>
<td>4.32e–23</td>
<td>26</td>
</tr>
<tr>
<td>P76323_INTG</td>
<td>P26462_FLIE</td>
<td>inverse</td>
<td>1.73e–22</td>
<td>25</td>
</tr>
<tr>
<td>P58042_RZOR</td>
<td>Q8ZQ98_Q8ZQ98</td>
<td>+1</td>
<td>3.11e–21</td>
<td>25</td>
</tr>
</tbody>
</table>

only 16 protein sequences from *E. coli* do not have any non-trivial frameshift overlap with any other protein sequence of *S. enterica*. The shortest non-trivial overlap is of length 7. The average length of frameshift overlaps is slightly less than in our previous experiment; around 12 amino acids.

4.3.3 Quality of the frameshift annotations

In this experiment we look at the quality of the reported frameshifts. We selected 578 DNA variants reported in the LOVD DMD Database [Aartsma-Rus et al., 2006] in coding regions that lead to a (predicted) frameshift variant. For all of these variants we checked whether the frameshift variant reported by our method corresponds to the frameshift variant that can be inferred from the DNA variant. The variants can be classified into 4 disjunct sets: deletions with 394 variants, duplications with 194 variants, insertions with 19 variants, and deletions/insertions with 26 variants. Note that there are no complex frameshift variants present in this database as it contains mainly on simplex

DNA variants. These variants can possibly be combined to construct a complex frameshift.

For this experiment we do not consider (predicted) frameshift variants with a length 1 or 2, because no reasonable annotation based on only the protein sequence can be expected. These amount to 15 cases in this dataset. In these cases often an ambiguous result is returned with our method. Usually, a combination of type +1 and type +2 is reported. One could argue that a frameshift with a length less than a certain length, depending on the input size, should not really be considered to be a frameshift. In either case it is in general impossible to distinguish such small frameshifts by looking only at the protein sequences, in particular frameshift variants of length 1 or 2.

### Deletions

Of all 394 deletion variants, only 2 unexpected results were found: c.980del with length 8 and c.3747del with length 6. Both should be characterized as type +2, however type inverse +1 was reported. Interestingly, the average length of frameshift variants is around twice the length of the misreported variants.

### Duplications

Of the 194 duplication variants we have 3 wrongly reported type inverse +1 frameshifts all resulting in frameshifts of length 6: c.4634dup, c.5697dup, and c.6848dup. In addition, we have one variant for which no frameshift was detected: c.1540dup, instead the trivial deletion/insertion was returned.

### Insertions and deletion/insertions

A minority of the variants in the DMD database leading to a frameshift variant is either an insertion or a deletion/insertion. There are 2 mis-classified variants reported for insertions: c.9672_9673ins with length 6, and c.10406_10407ins with length 4. Again, inversions were reported.

Overall, 7 of the 578 generated frameshifts seem to be unexpected when taking
the DNA variant into account. All of these frameshifts have a length of 8 amino acids or less, which is much less than the average frameshift length (21) of all variants in the LOVD DMD Database. Note that in all cases the reported frameshift annotation was not wrong (on the protein level), just unexpected from the inference of the DNA variants.

4.4 Discussion

In this section we explore the effects of the back-translation method described in Section 4.2.2 and we provide a suggestion for the extension of the HGVS nomenclature [den Dunnen et al., 2000] for the description of frameshift variants.

4.4.1 Back-translation

Here we elaborate on the back-translation method at its effects on real-life data. When considering a naive back-translation, i.e., no additional information available, a striking pattern is observed when measuring the ambiguity, i.e., the number of codons that represent a particular amino acid, per amino acid, see Figure 4.2.

Figure 4.2: The ambiguity of the first 66 nucleotides of protein P28697_HTGA of *E. coli*. The first two nucleotides of each codon typically have a low ambiguity (one or two), while the last nucleotide has a high ambiguity.

In Table 4.5 we show the average ambiguity for each nucleotide on each of the three positions in a codon using the five different frameshift types. We selected
the most significant frameshift for the intra-species frameshifts overlaps in *E. coli*. Note that for the naive method the selection should not make any difference. The different types of frameshift show no significant variation in the distribution of the ambiguity.

Table 4.5: The ambiguity per nucleotide position in a codon.

<table>
<thead>
<tr>
<th>Method</th>
<th>Nucleotide 1</th>
<th>Nucleotide 2</th>
<th>Nucleotide 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>1.25</td>
<td>1.07</td>
<td>3.15</td>
</tr>
<tr>
<td>Frameshift</td>
<td>1.10</td>
<td>1.01</td>
<td>1.06</td>
</tr>
</tbody>
</table>

The results in Table 4.5 show that back-translation method, explained in Section 4.2.2, using frameshift information results in a near perfect prediction of the individual nucleotides.

### 4.4.2 Proposed HGVS Descriptions

The HGVS nomenclature [den Dunnen et al., 2000] dealing with protein descriptions is, in our opinion, less well defined and structured as its DNA counterpart. Especially so for the description of frameshift variants. Our main objection is the fact that in the current standard, information is lost when describing a frameshift variant because only the length of the frameshift is included in the description, but the actual amino acid sequence is omitted, e.g., p.Arg97ProfsTer23. Furthermore, all description regarding protein sequences are rather verbose using the 3-letter abbreviations for amino acids (including capitalization). This makes human interpretation difficult, e.g., p.Lys2_Met3insGluSerLys, especially since most of the HGVS operators are also described by a 3-letter symbol (del, ins, etc.).

Currently, only two types of frameshift can be described with the HGVS nomenclature: +1 and +2, but they are not distinguished, and there is no way of describing a complex frameshift variant.

In order to accommodate for more comprehensive and complex frameshift variants, we propose the following structure and properties for protein descriptions:
4.4. Discussion

- all protein sequences are described using the single letter codes for amino acids;
- positions are not prefixed with the (redundant) reference amino acid (c.f. DNA descriptions);
- the same basic operators are used as for DNA description with the exception of inversion;
- frameshift descriptions are given as deletions/insertions with extra annotation so that no information is lost when describing a protein sequence.

Given a HGVS frameshift description using the current standard:

\[ p.\text{Cys10Valfs} \times 16 \]

\[ \text{can be written as: } q.10\_3685\text{delins10}_2\text{VMKEKMFKRKHSQNG}|2. \]

In this format the transposition notation from Chapter 3 is used to denote the frameshifts reference coordinates; the reference sequence from position 10 until position 3685 is replaced by the reference sequence from position 10 until position 24 with a frameshift of type +2 denoted by the operator |. These operators can be chained for the annotation of a multi-type frameshift variant. The actual amino acid sequence is also included (\text{VMKEKMFKRKHSQNG}) as in general many of such sequences could exist.

For more complex frameshift variants we can use the full transposition notation for alleles:

\[ q.10\_3685\text{delins}[Q\text{SHK}; 10\_15\text{KEKFMS}|1|2; A\text{AA}]. \]

In this case some parts of the inserted sequence could not be (effectively) described as frameshifts. A frameshift variant is described as 10\_15\text{KEKFMS}|1|2. In this case this is a multi-type frameshift variant; type +1 and type +2. Note that the frame numbering of the NCBI is different: both frame 0 and 1 denote the ‘normal’ reading frame, 2 corresponds to the second base offset, and 3 corresponds to the third base offset. This numbering is quite different from the current HGVS nomenclature. At this point it is unclear how to harmonize the different numberings.

The proposed format is implemented in the Description Extractor package for Mutalyzer \cite{Wildeman2008} (see Chapter 3) and is available at: https://github.com/mutalyzer/description-extractor.
4.5 Conclusions

We introduced a method for efficiently finding different types of frameshift variants in protein sequences without using any DNA level information. This method has as additional advantage that it can calculate the probability of a frameshift on several different levels depending sometimes on additional information available. It can use any codon table as basis, and the frameshift calculations can be precomputed for such a codon table resulting in efficient computation of the description and annotation.

This method can also be effectively used to generate a back-translation to DNA sequences, both for the reference sequence and for observed sequence.

We have suggested that large frameshifts are relatively common intra-species within *E. coli* and to a lesser extent inter-species. This method is able to identify promising candidates for evolutional research.

The generated frameshift descriptions on a real-life LOVD database show that the quality of the reported descriptions is high with only few unexpected results for very short length frameshifts.

The HGVS nomenclature has to be updated to accommodate for these new (complex) frameshift variants.