THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ON THE ORIGIN OF SEQUENCES: COMPUTATIONAL ANALYSIS OF SOMATIC HYPERMUTATION FOR PROBABILISTIC IMMUNOGLOBULIN PREDECESSOR IDENTIFICATION

BENJAMIN J. LENGERICH
SPRING 2015

A thesis submitted
in partial fulfillment
of the requirements for
baccalaureate degrees
in Computer Science and Mathematics
with honors in Computer Science

Reviewed and approved* by the following:

Raj Acharya
Head, Department of Computer Science and Engineering
Thesis Supervisor

Jesse Barlow
Professor, Department of Computer Science and Engineering
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

The human immune system uses complex mechanisms to generate enough antibody diversity to effectively protect against a wide array of potential antigens. These mechanisms obfuscate the germline predecessors of mature antibodies, making it difficult to produce a comprehensive model of the immune system. Interestingly, this problem also arises in the production of next-generation anti-viral software that use biomimicry to model malicious software as a recombination of attack patterns. Current methods fail to solve this problem in bicomputation because they ignore somatic hypermutation, one of the key methods of diversity generation. In this thesis, computational analysis is performed to develop a better model of somatic hypermutation, which is then used to improve antibody predecessor identification performance.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Prof. Raj Acharya for introducing me to the problem and the tremendous support and understanding along the way.

I would also like to thank Raunaq Malhotra for the starting point and fruitful discussions, as well as Prof. Mary Poss for help understanding the biological context.

Chapter 1

Introduction

1.1 Problem

The human immune system provides protection against a diverse set of antigens. For protection to be effective, an antibody must be available to pair with every potential antigen. As there is an unbounded number of potential antigens but finite storage space in the genome, it is not possible for the body to encode a gene for every antibody that might be needed. Thus, antibody diversity is generated through a two stage process: (1) V(D)J recombination followed by (2) somatic hypermutation.

The mechanisms of these methods will be discussed in detail, but here it suffices to say that this combination ensures that mature antibodies are significantly different from their predecessor genes, obfuscating the ancestral history of observed antibodies.

A number of methods have been proposed to overcome this obstacle; however, each has significant shortcomings. Initially, methods were proposed to align observed and germline nucleotide sequences, using this information to identify closest matching alleles. Unfortunately, these methods did not provide a meaningful way to evaluate different rearrangements. Recently, probabilistic methods have been designed to learn dependencies of each gene type and use such learned characteristics to identify gene segment boundaries. While such methods have enjoyed moderate success, they have ignored the aspect of somatic hypermutation, allowing the mechanism to lower predictive accuracy. The aim of this project is to reverse this disadvantage. By analyzing somatic hypermutation, unknown information about the mechanics of somatic hypermutation can be discovered and applied to improve prediction of germline predecessors of mature antibodies.

1.2 Motivation

The motivation for this project is 3-fold: (1) to increase accuracy of antigen and immune response models, (2) to unearth characteristics of somatic hypermutation, the mechanisms of which are mostly unknown, and (3) to advance toward a comprehensive and dynamic anti-viral software based on biomimicry.

The connection to the final motivational aspect is apparent when we reconsider the typical notion of a software virus. Rather than static software developed in isolation, malicious software can be viewed as a combination of attack patterns with significant post-recombination modification. With the use of advanced tools to identify the ancestral history of highly modified code, it may be possible to more accurately identify malicious code, a key step in the design of next-generation dynamic anti-viral software.

1.3 Goals

The specific aim of this thesis is to present a method for improved V(D)J recombination analysis through a software package.

Chapter 2

Background

2.1 Biochemical Context

In order to analyze somatic hypermutation, we must first understand its context within the adaptive immune system.

2.1.1 Immunoglobulin Function

The human immune system protects against a wide array of potential antigens through the extensive use of immunoglobulins. Produced by plasma cells in response to an antigen, immunoglobulins are glycoprotein molecules that have two main functions as part of the adaptive immune system: (1) antigen binding, followed by (2) effector functions.

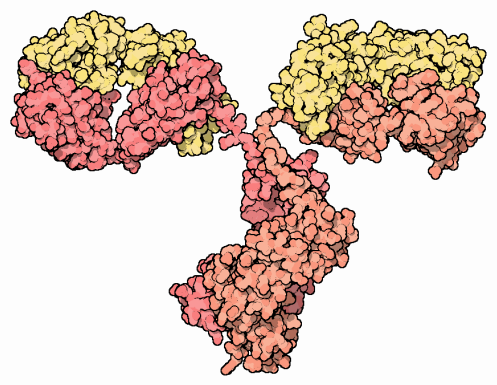

Figure 2.1: Immunoglobulin Representation. Reproduced from [1].

Antigen binding is highly specific, with immunoglobulins typically only binding to a few closely related antigens. This specificity is determined by immunoglobulin structure, as discussed below. However, once bound to the antigen, immunoglobulins trigger secondary effector functions through conserved patterns. These patterns include stimulation of the complement system (which lyses the antigen-presenting cell), and binding to phagocytic cells (which engulf and breakdown both the immunoglobulin and antigen-presenting cell).

2.1.2 Immunoglobulin Structure

The basic structure of an immunoglobulin is illustrated in Figure 2.2. While structural differences determine antigen specificity, all immunoglobulins have the same basic units:

- **Four Chain Structure**. Immunoglobulins are composed of two identical light chains and two identical heavy chains, bound together to form a "Y" shape.
- **Variable and Constant Regions**. Each chain can divided into two regions based on variability in the amino acid sequences.
- **Hinge Region**. The immunoglobulin molecule forms a "Y" shape due to the flexibility in the molecule at the hinge region.
- **Disulfide Bonds**. Inter-chain disulfide bonds hold the heavy and light chains together, while intra-chain disulfide bonds constrain each polypeptide chain.

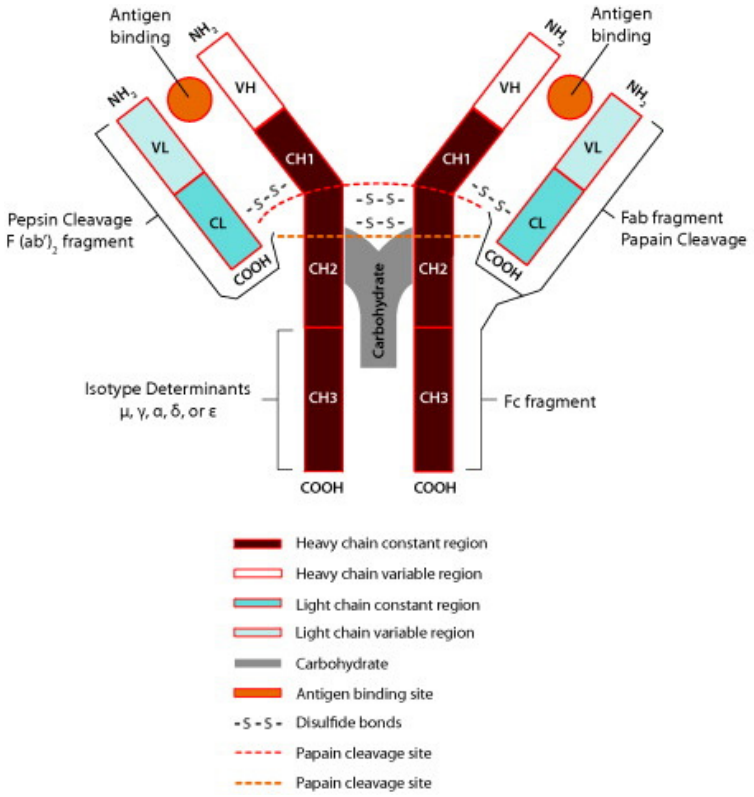

Figure 2.2: Basic Immunoglobulin Structure. Reproduced from [2].

2.1.2.1 Immunoglobulin Class Type

Human immunoglobulins can be divided into five classes, as depicted in Figure 2.3. This division is based on differences within the constant region of the heavy chain. Class type determines immunglobulin polymerization; IgM immunoglobulins form pentamers while IgA immunoglobulins form dimers.

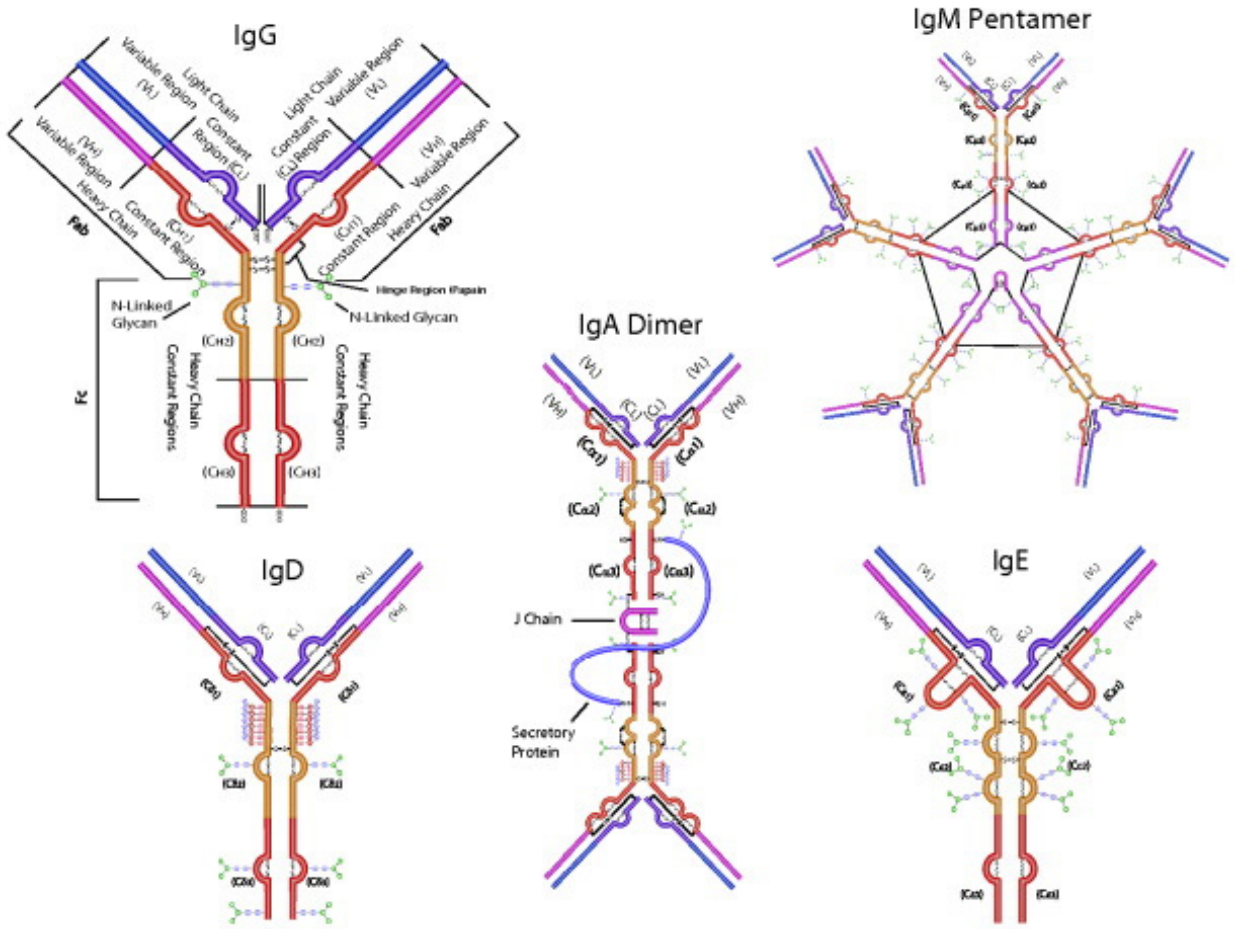

Figure 2.3: Immunoglobulin classes. Reproduced from [2].

2.1.2.2 Complementarity Determining Regions

Within the variable regions, three complementarity determining regions (CDRs) have been identified. The amino acid sequences in these regions produce the structure which interfaces with the antigen, determining the specificity of the immunoglobulin. For this reason, immunoglobulins with identical specificities have identical CDRs, while those with different specifities have different CDRs. Within the variable region, CDR1 and CDR2 are located in the V segment of a polypeptide chain, while CDR3 spans V, D, and J segments. CDR3 is the most variable, as it is encoded through a recombination event, as described below.

2.1.3 Diversity Generation

In order to ensure a matching immunoglobulin for every potential antigen, a human must encode more immunoglobulins than there are genes in the human genome. In fact, it has been estimated that the human preimmune antibody repertoire is capable of making more than 10^{12} different immunoglobulin molecules, while the entire human genome contains fewer than $40,000$ functional loci [3][4].

To achieve such diversity, antibodies use a two stage process: (1) V(D)J recombination followed by (2) somatic hypermutation. The combination of these processes obfuscates predecessor identification, making it very difficult to determine which germline sequences recombined to form an observed antibody.

2.1.3.1 V(D)J Recombination

As illustrated in Figure 2.4 (a), each heavy chain variable region is formed from the concatenation of a variable (V), a diversity (D), and a joining (J) segment. These segments are recombined from germline alleles, as illustrated in 2.4 (b). In humans, there are 281 V gene segments, 84 D gene segments, and 12 J gene segments, allowing for a potential 283,248 recombinations of one segment of each type[5].

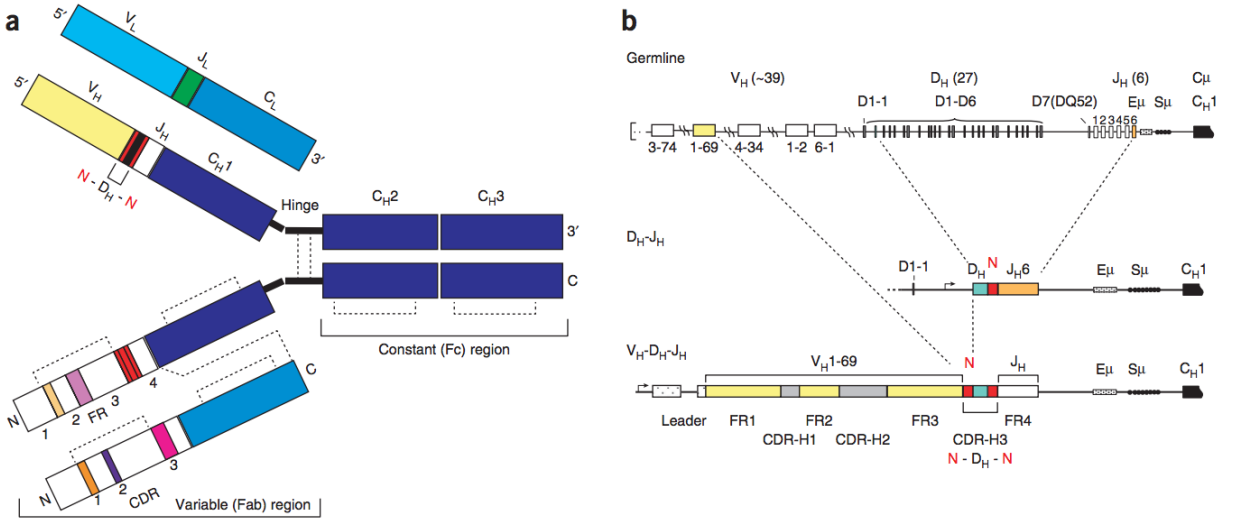

Figure 2.4: The Process of VDJ Recombination. Reproduced from [6].

Moreover, the joining between segments is imprecise, further increasing the variability of recombinations. RAG proteins introduce double-stranded breaks at flanking DNA sequences, resulting in random insertion or deletion of nucleotides at segment boundaries [3]. This process, is known as junctional diversification, greatly increases antibody diversity.

2.1.3.2 Somatic Hypermutation

While recombination of germline genes can generate a low-affinity antibody for every antigen, antibodies with higher affinity for the antigen will produce a more effective immune response.

To this end, point mutations in variable segment coding sequences accumulate through a process known as affinity maturation. This process occurs after recombination, helping to generate high-affinity memory cells. The mutations occur at a rate of about one mutation per variable region per cell generation [3]. As this is nearly a million times faster than spontaneous mutation rates in other genes, the process is called somatic hypermutation.

The mechanism of somatic hypermutation uses an error-prone DNA repair process. It is thought that activation-induced cytidine deaminase (AID), a 24 kDa enzyme, creates mutations through deamination of cytosine bases. By changing cytosine to uracil, AID produces a mismatched base pair that is repaired by an error-prone mechanism [7]. Thus, the mutations produced by somatic hypermutation are mainly single base substitutions, with occasional insertions and deletions.

2.2 Existing Methods of Predecessor Identification

The combination of these processes to generate antibody diversity makes it difficult to identify which germline genes produced an observed antibody. However, as this problem is important for modeling the immune system, several methods have been developed.

2.2.1 Initial Methods

Initial methods for germline predecessor identification relied on alignment of mature sequences to germline segments. IMGT/V-QUEST maps the DNA sequences of mature antibodies to an immunoglobulin and T-cell database, then uses this alignment to identify structurally important features of the mature antibody [8]. In contrast, the JOINSOLVER program determined conserved motifs that distinguished V, D, and J segments, and uses these motifs to identify the germline predecessor [9]. SoDA(2) uses a dynamic programming based implementation to perform a 3D alignment to identify germline sequences [10].

While these methods have had some success, they do not give any measure to meaningfully evaluate rearrangement confidence. Also, the large number of possible alignments makes alignment-based methods computationally expensive and prohibitively slow.

2.2.2 Probabilistic Methods

To solve these shortcomings, probabilistic methods have more recently been developed. Such methods have either been based on Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs).

2.2.2.1 HMM-based

Initial probabilistic methods were based on HMMs as they make simplifying assumptions to reduce computational complexity in training. These models are based on the assumption that nucleotide occurence rates follow a Markov Process, explained below.

2.2.2.1.1 Markov Process

Named after the Russian mathematician Andrey Markov, a **Markov process** is a finite-state, memory-less, nonlinear model of a sequence of states. That is, at any point in Markov process, the probability of the succeeding state depends only on the current state. Thus, in a Markov process with a known transition function, knowledge of the entire history of the process provides no predictive advantage over knowledge of just the present state.

More formally, given a finite set of states $S = \{s_1, ..., s_n\}$ and transition function $f : S \times S \rightarrow [0, 1]$, a Markov process is an ordered set of observations $\{x_1, ..., x_m \in S\}$ that satisfies the Markov property

$$P(X_m = x_m | X_{m-1} = x_{m-1}, ..., X_0 = x_0) = P(X_m = x_m | X_{m-1} = x_{m-1}) = f(x_{m-1}, x_m) \tag{2.1}$$

While such a process may appear to be limited, Markov processes allow powerful modeling of sequences with a few assumptions. As a simple example, consider the case of the degenerate gambler who wagers \$1 on each flip on an unending, fair coin flip. Let X_i be be the amount of money the gambler has after the ith flip, with $X_0 = \$10$. If the gambler continues to gamble indefinitely (or until losing all his or her money), then the sequence $\{x_n : n \in [0, \infty]\}$ is a Markov process.

2.2.2.1.2 Hidden Markov Models

A **Hidden Markov Model** (HMM) is a model that approximates the probabilities governing an unknown Markov Process. Given on an sequence of observed emissions each generated by an unobserved states, an HMM models the probabilities governing the Markov process.

This model makes two key assumptions. First, the Markov assumption states that the underlying process obeys the Markov property shown above (that is, the next state is dependent only on the current state). Second, the independence assumption states that the current observation is based only on the current state, and is independent of previous observations and states.

Formally, an HMM is defined as follows. Given a finite state alphabet $S = \{s_1, ..., s_n\}$, a finite observation alphabet $V = \{v_1, ..., v_n\}$, an observed sequence $O = \{o_1, ..., o_k : o_i \in V\}$ and an unobserved state sequence $Q = \{q_1, ..., q_k : q_i \in S\}$, an HMM λ is a 3-tuple

$$\lambda = (A, B, \pi) \tag{2.2}$$

where A is the transition function of the Markov process, B is the emission function, and π is the initial probability function [11].

$$A : S \times S \rightarrow [0, 1], A(s_i, s_j) = P(q_t = s_i | q_{t-1} = s_j) \tag{2.3}$$

$$B : V \times S \rightarrow [0, 1], B(v_i, s_j) = P(o_t = v_i | q_t = s_j) \tag{2.4}$$

$$\pi : S \rightarrow [0, 1], \pi(s_i) = P(q_1 = s_i) \tag{2.5}$$

Note that A and B do not vary with position in the sequence and demonstrate the Markov assumption that only knowledge of the previous state is required to make accurate predictions.

2.2.2.1.3 iHMMune-align

iHMMune-align is one such HMM-based probabilistic model [12]. The software first identifies the closest matching V segment, then builds an HMM which encodes transitions to all possible D and J segments based on the emitted nucleotide sequence. Another similar software package is the Soda2 statistical model [13]. While these methods are efficient, the simplifying assumptions of the HMM lower its accuracy.

2.2.2.2 CRF-based

While HMM-based probabilistic methods have enjoyed some success, the model makes stringent assumptions about the independence of the distributions of bases. To relax such assumptions, new methods based on conditional random fields have been developed.

2.2.2.2.1 Conditional Random Fields

A conditional random field (CRF) is a probabilistic model that relaxes some of the assumptions made in HMMs. Originally used for segmentation and sequential labeling in natural language processing, CRFs model the conditional probability of label sequences based on all labels within the sequence. Formally, for an input sequence $x = x_1x_2...x_n$, the linear-chain CRF calculates the conditional probability of a label sequence $y = y_1y_2...y_n$ as proportional to

$$\sum_i exp(\sum_j \lambda_j h_j(y, x, i)) \tag{2.6}$$

where $h_j(y, x, i)$ is a feature function defined on some subset of the input variables and output labels. Two common feature functions are the transition feature function $h_j(y_i, y_{i-1}; x, i)$ and the state feature function $h_j(y_i; x, i)$. The feature functions can be designed to capture dependencies throughout the entire sequence x, making CRFs more powerful than HMMs. Each feature function is given a weight λ_j. Thus the normalizing constant $Z(x)$ is defined as

$$Z(x) = \sum_y \sum_i exp(\sum_j \lambda_j h_j(y, x, i)) \tag{2.7}$$

and the probability of a label sequence y for an input sequence x is given by

$$P(y|x) = \frac{1}{Z(x)} \sum_i exp(\sum_j \lambda_j h_j(y, x, i)) \tag{2.8}$$

Thus, parameters $\Lambda = \{\lambda_j\}$ are trained to maximize the log-likelihood over all training samples

$$L(\Lambda) = \sum_{x,y} logP(y|x) \tag{2.9}$$

and predicted sequence labels $y*$ for a test sequence $x*$ are those given by

$$y* = argmax_y P(y|x*) \tag{2.10}$$

2.2.2.2.2 Raunaq's Method

The use of a linear-chain CRF has been used successfully to identify immunoglobulin sequence boundaries [5]. In the absence of knowledge about the grammar rules of VDJ recombination, feature functions were chosen to capture dependencies around a 5-base neighborhood. After identification of VDJ boundaries, and subsequent mode filtering, germline alleles were determined by mapping segments to the germline. This method set a new state of the art for germline identification, error rates are shown in Table 2.1.

	iHMMune Align		**CRF-based**	
Segment Type	**# Errors**	**Error %**	**# Errors**	**Error %**
V genes	707	5.3	136	1.0
D genes	1008	7.6	68	0.5
J genes	10	0.08%	18	0.13%

Table 2.1: Error Rates for Probabilistic VDJ identification. Reproduced from [5].

Chapter 3

Materials and Methods

3.1 Method Overview

The methods can broadly be broken into two investigations: somatic hypermutation analysis, and germline predecessor identification improvements. Hypermutation analysis required quantification of hypermutation rates, identification of hypervariable regions, and conserved motif discovery. Germline predecessor identification improvements came from immunoglobulin class clustering and combining multiple sources of segment identification data.

3.2 Datasets

Two datasets were used: a synthetic dataset for training the CRFs, and the Stanford S22 dataset for somatic hypermutation analysis and generalizability estimation.

3.2.1 Synthetic Dataset

The CRFs were trained on a synthetic dataset of 283,248 rearrangements of germline alleles. These segments were generated by synthetically recombining the Kabat database of germline V, D, and J alleles available on the JOINSOLVER website [9]. Statistics of the lengths of these alleles are shown in Table 3.1. The distribution of the lengths of the recombinants is shown in Figure 3.1, with statistics shown in Table 3.2.

Segment Type	Number	Mean Length	Max Length	Min Length
V	281	287	305	103
D	84	25	37	11
J	12	53	63	48

Table 3.1: Synthetic Dataset Alleles Length Distribution

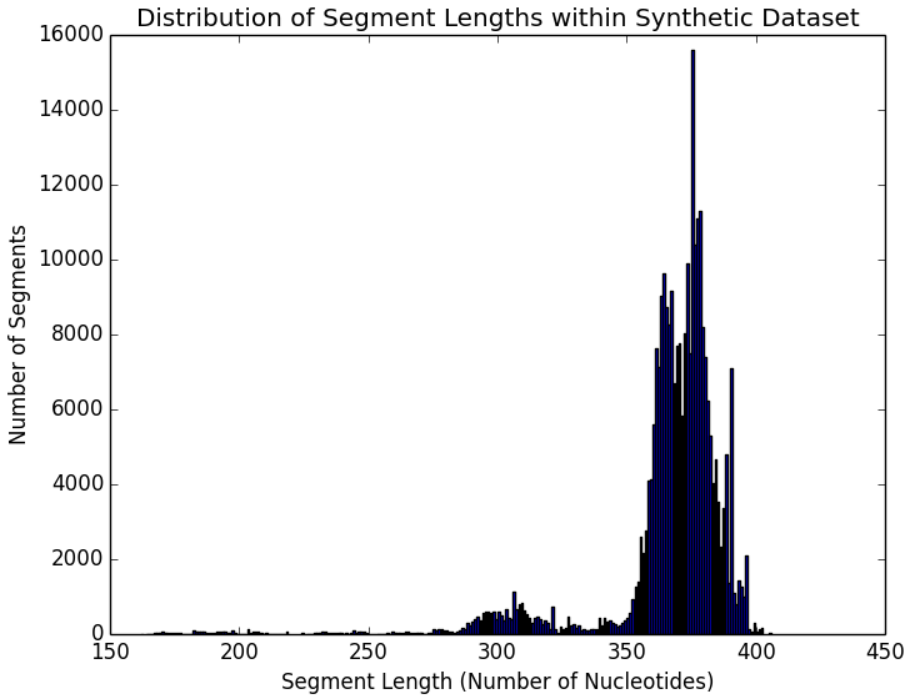

Figure 3.1: Distribution of Lengths of Recombinations from Synthetic Dataset

Number of Sequences	Mean Length	Max Length	Min Length
282402	365	405	162

Table 3.2: Distribution of Lengths of Recombinations within Synthetic Dataset

3.2.2 Stanford Dataset

Unfortunately, as the affinities of somatic hypermutation are not clear, the synthetic dataset cannot be designed to encode hypermutation processes. To this end, the Stanford_S22 dataset was used. The Stanford_S22 dataset consists of 13,153 reads from the recombined VDJ genes of an individual. The reads were produced via 454 pyrosequencing of genomic DNA from peripheral blood mononuclear cells, with chimeric sequences excluded [14]. The distribution of the lengths of the recombinants is shown in Figure 3.2, with statistics shown in Table 3.3.

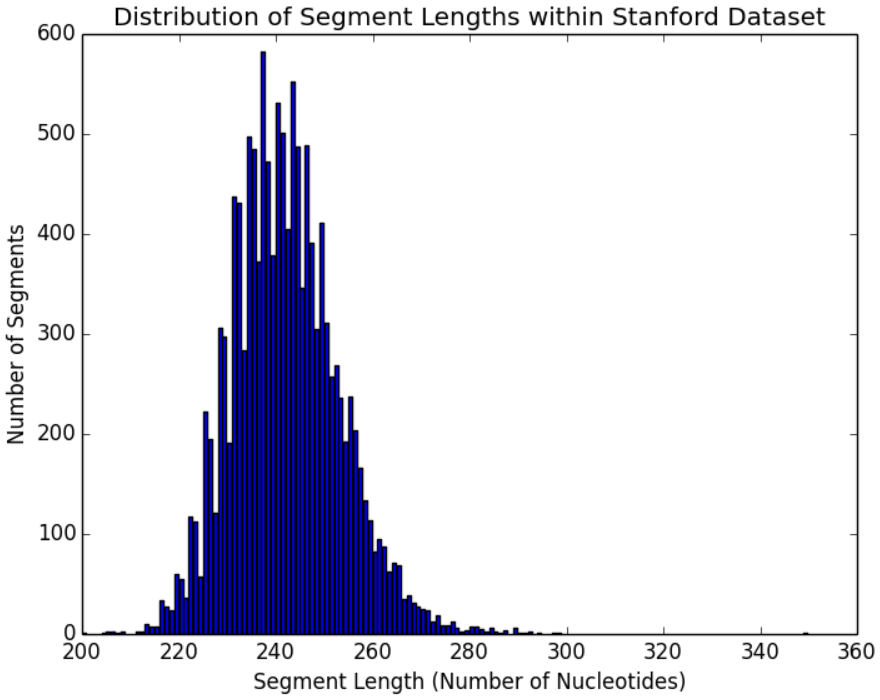

Figure 3.2: Distribution of Read Lengths within Stanford Dataset

Number of Sequences	Mean Length	Max Length	Min Length
13153	242	349	200

Table 3.3: Distribution of Read Lengths within Stanford Dataset

3.3 Clustering Immunoglobulin Classes

As the constant regions of immunoglobulins have significant differences by class (Section 2.1.2.1), it was hypothesized that the variable regions may also exhibit class-specific dependencies. Such dependencies had not been identified by inspection, so cluster analysis was performed to label classes and identify patterns.

3.3.1 k-Means Clustering

In order to cluster the sequences, the algorithm of k-means clustering was used. Introduced in 1967[15], the k-Means clustering algorithm is a method to produce an optimal partitioning of M points in N dimensions into K clusters. After randomly initializing K cluster centers, the algorithm follows an expectation-maximization paradigm:

1. Every data point is labeled as belonging to the cluster with the closest center.

2. Every cluster center is updated to be the mean location of its labeled data points.

The algorithm converges when there are no changes in the assignments of data points. While the k-means algorithm is simple and efficient, it has two major drawbacks. First, the number of clusters, K, must be known *a priori*. Second, as the algorithm relies on calculating $K * M$ pairwise distances as well as a new mean cluster location at every iteration, efficient distance calculations are prerequisite for scalability. While this is simple for Euclidean metrics of data in $\mathbb{R}^n$, it is not so clear for strings, especially of variable length.

To perform k-means clustering on a set of data points $X = \{x_1, ..., x_m\}$, a distance metric $D : X \times X \rightarrow \mathbb{R}$ must be defined. Furthermore, to ensure convergence to a local minimum, this distance metric must obey the triangle inequality $D(x + y) \leq D(x) + D(y)$.

3.3.1.1 Levenshtein Distance

One such metric that is frequently suggested is the Levenshtein edit distance metric [16]. This metric defines the distance between two strings to be the minimum number of single-character edits (insertions, deletions, or substitutions) that are needed to transform one string into the other. However, this distance metric does not behave well for strings of variable length (spaces are penalized as much as substitutions), so it is inappropriate for the Stanford S22 dataset.

3.3.1.2 Global alignment Scores with Needleman-Wunsch

To remedy this problem of variable length sequences, the global alignment scoring included in the BioPython module pairwise2 was used. This package uses the Needleman-Wunsch dynamic programming algorithm to efficiently align two sequences. As junctional diversification creates many insertions and deletions, no gap penalties were assessed.

3.4 Analysis of Hypervariable Regions

In order to identify patterns of and understand the mechanisms of somatic hypermutation, hypervariable regions were identified and analyzed.

3.4.1 Hypermutation Rate Quantification

To the author's knowledge, there were no publicly available datasets of hypermutation rates. Thus, hypermutation rates had to be quantified and compiled. By mapping each read of the Stanford_S22 dataset to find the closest sequence in the germline (using a trained CRF to identify segment boundaries), individual base pairs could be examined to determine whether they had been mutated. Transition mutations, transversion mutations, and insertion/deletion mutations were all recorded seperately. This quantification allowed extensive analysis (see Section 3.4.2).

3.4.2 Hypervariable Region Identification

Using the quantified hypermutation rates, both hypervariable and hypovariable regions were identified. Hypervariable regions were classified as regions in which the 10 nucleotide running average

of the hypermutation rate was elevated for at least 50 nucleotides. Conversely, hypovariable regions were defined as regions in which the 10 nucleotide running average of the hypermutation rate was decreased for at least 50 nucleotides.

3.4.3 Motif Identification

After identifying hypervariable regions and hypovariable regions, analysis could be performed (see 3.4.2). To the author's knowledge, little research had been performed on hypovariable regions, suggesting analysis of such regions may be fertile. To this end, nucleotide motifs were discovered using the DREME tool available online [17].

3.5 Germline Identification

As discussed in Section 2.2.2.2.2, the state-of-the-art method uses a linear-chain CRF to identify immunoglobulin sequence boundaries. After a rudimentary mode filtering algorithm fixes the boundaries, alignment methods find the corresponding germline V, D, and J genes.

To improve the mode filtering algorithm, a new method that combines mode filtering with the alignment step was developed. This algorithm seeks to maximize a score function calculated by multiplying the Needleman-Wunsch score by the product of the probabilities that each base is of the putative segment type. By finding the set of sequences that maximize such a function, the most likely sequences can be identified, combining both the mode filtering and alignment steps in one..

3.6 Implementation Details

Analysis code was written in Python and C++ and run using Amazon Web Services. Conditional Random Fields were implemented using CRF++[18].

Source code is available at http://blengerich.github.io/thesis.

Chapter 4

Results and Analysis

4.1 Immunoglobulin Class Clustering

Because of the biological significance of immunoglobulin classes, it was thought that clustering by class may provide a benefit to identification of germline sequences. As discussed in Section 3.3.1, the k-Means clustering algorithm was run with a distance metric from Needleman-Wunsch. Consensus centroid sequences are shown in Table 4.1.

4.1.1 Class-based Segment Prediction

After identifying clusters, a CRF was trained on 60% and tested on 40% of each cluster independently. Precision and recall rates are shown in Table 4.2. In Table 4.3, the precision and recall rates are shown for the combination of all class CRFs. When combined into a single method (using class labeling followed by the corresponding CRF for prediction), performance is much better than that of a single CRF.

Consensus Sequence	% Sequences
GAGGTGCAGCTGGTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGG-GGGTCCCTGAGA CTCTCCTGTGCAGCCTCTGGATTCACCTTCAGTAACTACTGCATGCACTGGGTCCGCCAG GCTCCAGGGAAGGGGCTGGAGTGGGTGGGATTTATTAAAAG——TGATGGTGGTAAA ACATACTACGCAGACTCTGTGA———AGGGCA—-GATT-CACCATCTCCA———GAG A———CAATTCCAAGAACACACT———GTATCTGCAAATGAACAGCC–TGAGAGCCG AGG—-ACACGGCCGTGTATTACTGTGCGAGA————————AGTAATAGCAGTAA CTAC—————ATTACTAC——-TACTACTACGGTATGGACGTCTGGGGCCAAGGGACC ACGGTCACCGTCTCCTCAG	29.25
—-GAGGTGCAG-CTGGTGGAG—-TCTGGGGGAGGCTTGGTACAGC————-CTGG—- ———G-GGGTCCCTGAGACTCTCCTGTGCAG-CCTCTGGATTC—-ACCTT-CAGTAGC ———TATGGTATGCACTGGGTCCGCCAGGCTCCAGGGAAGGGGCTGGAGTGGGTG GCAGGTATTAATAC——TAATGGTGG-TAACACAT—-ACTACGCAGACTCCGTGAAG GGC-CGATTCACCATCTC————-C–AGAGACAATTC—————CAAGAACACGCTG —-TATCTGCAA—-AT——-GAACAGCCTGAGAGCTGAGGACA-CGGCCG-T-GT—- —-AT———TACTGTGCGAG-A————————————GA———————————— ————————————-GGTACCATTACTACAACTACTA——————CTTT GACTTCTGGGGCCAAGGGACCCTGGTCACCGTCTCCTCAGTCTCCTCAG	11.95
CAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACCCTGTC CCTCACCTGCACTGTCTCTGGTGGCTCCATCAGCAGT-GTGGTTACTACTGGAGCTGG ATCCGGCAGCCCCCAGGGAAGGGCCTGGAGTGGATTGGGTACATCTATTACAGTGGG AGCACCAACTACAACCCGTCCCTCAAGAGTCGAGTCACCATATCAGTAGACACGTCCA AGAACCAGTTCTCCCTGAAGCTGAGCTCTGT-G——ACCGCCGCGGACACGGCCGT G—-TATTACTG–TGCGAGAG————————————AACTACTACTAT———AC—— —————————TTTGACTACTGGGGCCAAGGAACCCTGGTCACCGTCTCCTCAG	1.16
CAGGTGCAGCTGGTGCAGTCTGGGGCTGAGGTGAAGAAGCCTGGGGCCTCAGTGAAGG TCTCCT-GCAAGGCTTCTGGATAC—-ACCTTCACCAGCTACTGTATCCGCTGGGTGCGACAG GCCCCTGGAAAAGGGCTTGAGTGGATGGGAAGGATCAATCCTAGTGATGGTGATACAAAC TACGCACAGAAGTTCCAGGGCAGGGTCACCATCACCGCGGACAAGTCCATCAGCACAGC CTACATGGAGCTGAGCAGCCTGAGATCTGAGGACACGGCCATGTATTACTGTGCGAGAG- —————————G–A-AACCACTA-TACCAT–TACT———ACTACTACTAC———— TATGGACGTCTGGGGCCAAGGGACCACGGTCACCGTCTCCTCAG	23.32
CAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACCCTGTCCCT CACCTGCACTGTCTCTG——GTGGCTCCATCAGCAGTAGTGGTTACTACTGGAGCTGGATCC GCCAGCCCCCAGGGAAGGGCCTGGAGTGGATTGGGTACATCTATTA-T——-AGTGGGAG CACCTACTACAACCCGTCCCTCAAGAGTCGAGTCACCATATCAGTAGACACGTCCAAGAAC CAGTTCTCCCTGAAGCTGAGCTCTGTGACCGCCGCGGACAC–GGCCGTGTATTA——— -CTGTGCG-AGAGAGTTATAACAA————————————————TTACTACTACTACT- A–CGGTATGG—————-ACGTCTGGGGCCAAGGGACCACGGTCACCGTCTCCTCAG	34.32

Table 4.1: Consensus Immunoglobulin Class Sequences by Needleman-Wunsch Distance

Cluster	Segment Type	Precision	Recall
0	V	0.96762	0.97708
0	D	0.69680	0.51999
0	J	0.88699	0.90729
1	V	0.97677	0.97575
1	D	0.72790	0.61774
1	J	0.87847	0.92549
2	V	0.93128	0.95329
2	D	0.24731	0.07055
2	J	0.58266	0.63317
3	V	0.95308	0.98020
3	D	0.63596	0.47317
3	J	0.88904	0.83989
4	V	0.97817	0.97134
4	D	0.68325	0.51538
4	J	0.67	0.93408

Table 4.2: Precision and Recall of Class CRFs on Synthetic Dataset

	Precision		Recall	
Segment Type	Single CRF	Class CRF	Single CRF	Class CRF
V	0.82310	0.96852	0.95107	0.97535
D	0.55693	0.67646	0.60015	0.51396
J	0.84110	0.86647	0.50468	0.89976

Table 4.3: Weighted Average of Precision and Recall of Class CRFs on Synthetic Dataset

4.2 Hypermutation Rate Investigation

As knowledge of the distributions of hypermutation rates is not fully developed, hypermutation rates and their distributions were investigated. First, the hypermutation rate was quanitified by base position. This allowed analysis of hypermutation rates across segments, and visualization of hypermutation rate patterns specific to segments. Finally, hypervariable and non-hypervariable regions were identified and novel motifs were discovered for non-hypervariable regions.

4.2.1 Quanitification by Base Position

First, as no dataset of hypermutation rates by base position has been made publicly available, this dataset had to be constructed.

Preliminary attempts at quantifying these rates suggested that the distributions may be meaningful. Figure 4.1 shows the average hypervariability (Needleman-Wunsch distance two closest sequences) of 100 random sequences by base position. Three parametric curves can be identified, suggesting a potential pattern of hypermutation - elevated within a V,D, or J segment rather than spanning the boundary.

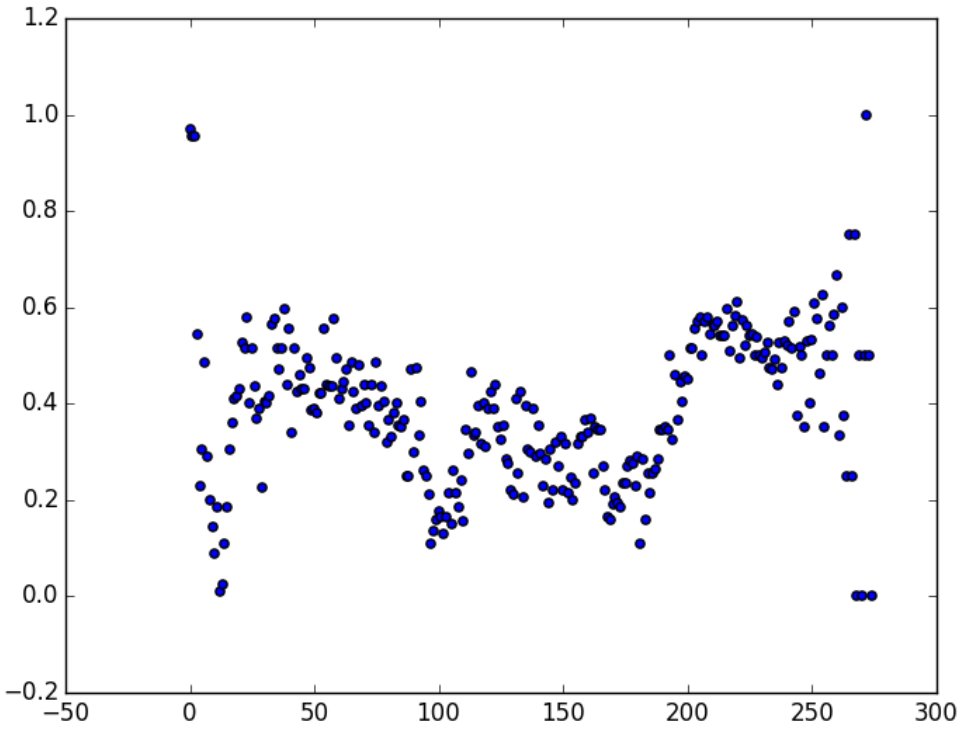

Figure 4.1: Hypervariability of Sequence by Base Position

Through significant analysis (Figure A1-Figure A9), patterns of hypermutation rates were clarified. By separating segment types and normalizing base position by the length of the segment, it was shown (see Figure 4.2) that hypermutation rates remain mostly constant within the D and J segments, while fluctuating repeatedly within V segments.

Furthermore, the distribution of hypermutation rates with V segments appear to have three or four peaks, potentially corresponding to the complementarity determining regions. There are three known complementarity determining regions: CDR1 is located in the beginning of the V segment, CDR2 is contained within the middle of the V segment, and CDR3 spans the end of the V segment as well as D and J segments. Thus, it is logical that the calculated hypermutation rates could

correspond to complementarity determining regions.

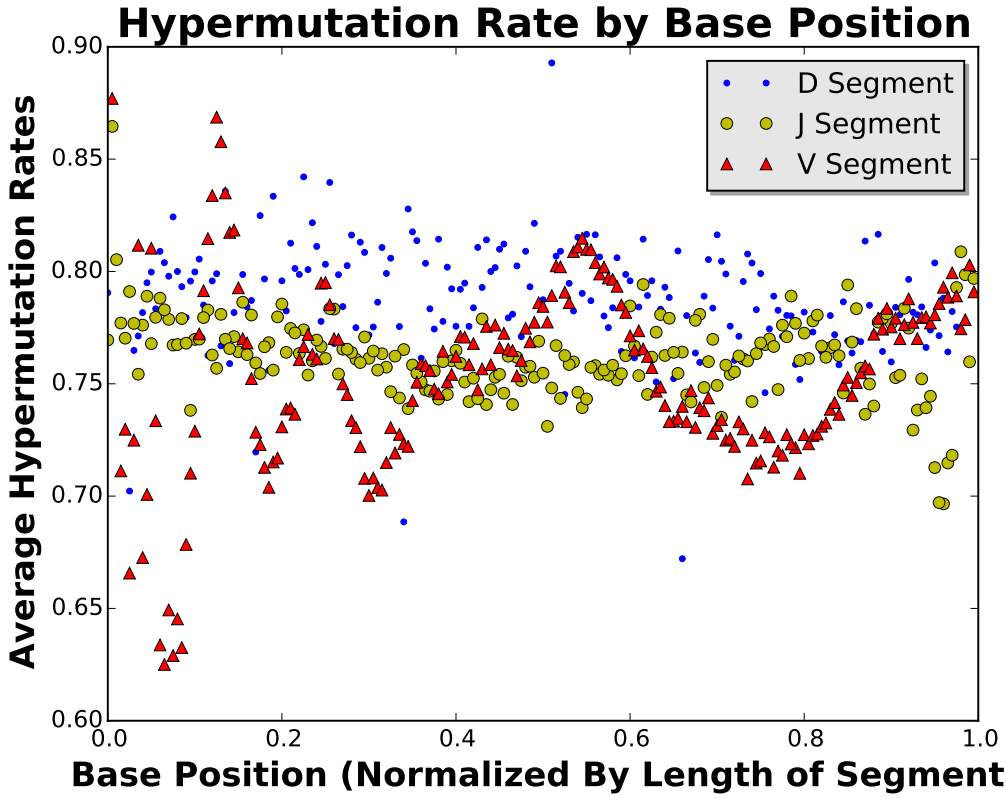

Figure 4.2: Mean Hypermutation Rate by Base Position for V, D, J segment types

As CDR3 spans the entire D and J segments, it would be expected that properly quanitified hypermutation rates would be elevated throughout the D and J segments. To this end, basic statistical analysis was performed on the running averages of hypermutation rates in each of the segment types. Selected results are presented in Table 4.4, full results are given in A1.

Segment Type	Mean	Std. Dev.
V	0.750959	0.149849
D	0.785855	0.075827
J	0.761050	0.132611

Table 4.4: Statistical Analysis of Running Averages of Length 9

4.2.2 Mutation Type

To further probe the distribution of hypermutation rates, mutations were separated by type (transition, transversion, insertion or deletion). This distribution is presented in Figure 4.3 for V segments, Figure A10 for D segments, and Figure A11 for J segments. While much of the distri-

butions seem stochastic, an interesting phenomenon occurs within V segments. Nearly all insertion/deletion mutations occur either at the midpoint or near the end of the segment. Junctional diversification increases mutation rates near segment boundaries, so it makes sense that the mutation rate would be increased near the end of the V segments. However, it is not clear why the midpoint of the V segment would also experience a similar increase in insertion/deletion mutations. This finding suggests further analysis, both computationally and experimentally, may be needed.

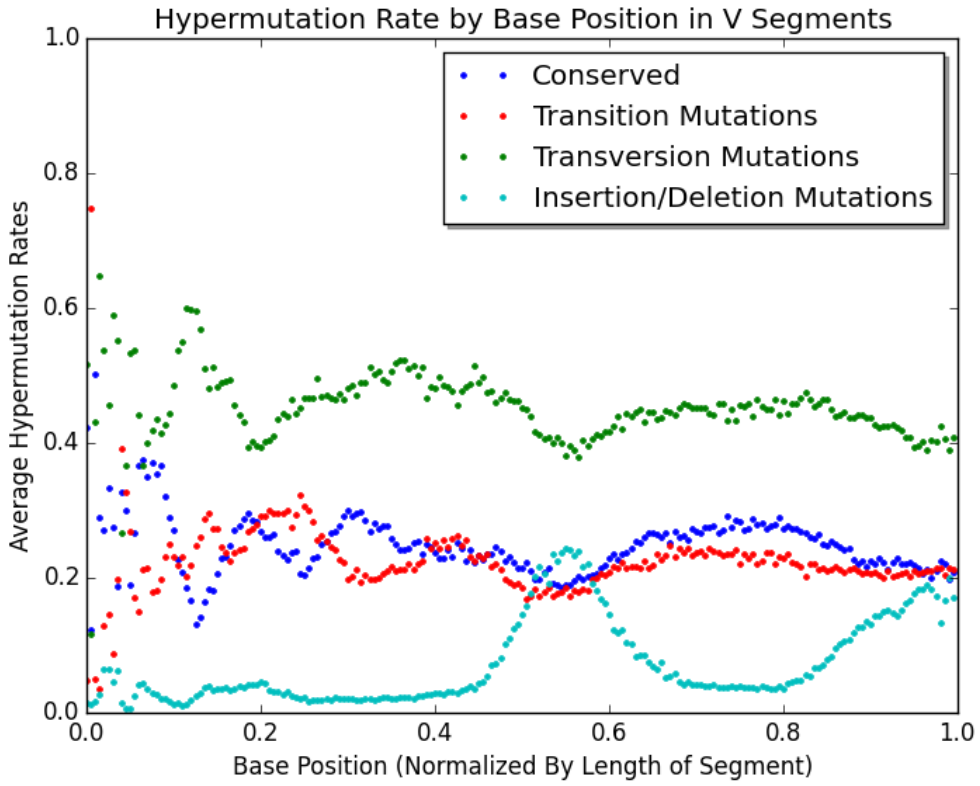

Figure 4.3: Hypermutation Rate by Base Position For each Type of Mutation (V Segments)

4.2.3 Motif Discovery

As discussed in Section 3.4.2, both hypervariable and hypovariable regions were identified. As hypovariable regions have been largely ignored for motif discovery, the author determined that motif discovery may be fruitful within hypovariable regions. Indeed, 19 motifs were found to be enriched in these regions, suggesting potential for experimental procedures. Such experimental procedures might investigate the possibility of factors binding to prevent hypermutation in hypovariable regions. Such findings would be a significant change to the current understanding that factors (such as AID) bind to promote hypermutation without competing protective factors.

Here the 5 motifs found to be most enriched in hypovariable regions are shown. The full table of 19 motifs is in Table A2.

Motif	e-Value
	4.2e-972
	2.7e-414
	7.5e-301
	2.1e-199
	4.4e-051

Table 4.5: Nucleotide Motifs Found to be Enriched in Non-Hypovariable Regions. An e-Value less than 1.0e-3 is generally considered significant.

4.3 Predecessor Identification

To improve identification of germline predecessors of mature antibodies, the above data was combined with a state-of-the-art CRF for segment prediction. Given a mature antibody, cluster algorithms assigned the read to a given cluster, which then used a trained CRF specific to that cluster. After the CRF identified base-wise probabilities for V, D, and J segment ownership, the alignment followed the novel algorithm discussed in section 3.5.

This method successfully improved identification of germline genes, scoring a 0% missassignment on a benchmark tool [14] for the Stanford_S22 dataset. However, the author believes this value is most likely due to a bug in the calculation of errors. Thus, this new technique, while promising, requires further evaluation.

Chapter 5

Discussion

5.1 Conclusions

In this thesis, the characteristics of immunoglobulin recombination were investigated. Immunoglobulins were clustered (presumably by immunoglobulin class), and it was shown that training a separate CRF for each class increases segment type identification as compared to the use of a single CRF. The rate of hypermutation was quantified by base position and a dataset of hypermutation rates has been made available. Preliminary analysis showed that hypermutation rates follow a pattern that can be explained by the biological structure of complementarity determining regions, suggesting further analysis could confirm this relationship. Furthermore, the pattern of insertion/deletion mutations within V segments is interesting, suggesting some biological phenomenon may cause an increase in diversification through mid-segment insertion/deletions. Finally, 19 motifs were found to be significantly enriched in hypovariable regions, suggesting potential for experimental investigation.

5.2 Future Work

This work suggests potential for future experiments, both experimental and computational. Experimentally, the nucleotide motifs strongly conserved in hypovariable regions should be investigated to identify protective factors. Also, the distribution of hypermutation rates, especially insertion/deletion mutations, should be investigated experimentally. Computationally, the new method of predecessor identification described should be investigated and the results studied rigorously to confirm performance advances.

Bibliography

[1] David Goodsell. Antibodies, September 2001.

[2] Sigma-Aldrich. *Antibody Basics*, 2015.

[3] B Alberts, A Johnson, and J Lewis. *Molecular Biology of the Cell*. Garland Science, New York, 4th edition, 2002.

[4] S. Ohno. An argument for the genetic simplicity of man and other mammals. *Journal of Human Evolution*, 1(6):651–662, November 1972.

[5] Raunaq Malhotra, Shruthi Prabhakara, and Raj Acharya. Predicting v(d)j recombination using conditional random fields. In Tetsuo Shibuya, Hisashi Kashima, Jun Sese, and Shandar Ahmad, editors, *Pattern Recognition in Bioinformatics*, volume 7632 of *Lecture Notes in Computer Science*, pages 210–221. Springer Berlin Heidelberg, 2012.

[6] George Georgiou, Gregory C Ippolito, John Beausang, Christian E Busse, Hedda Wardemann, and Stephen R Quake. The promise and challenge of high-throughput sequencing of the antibody repertoire. *Nat Biotech*, 32(2):158–168, 02 2014.

[7] Ziqiang Li, Caroline Woo, Maria Iglesias-Ussel, Diana Ronai, and Matthew Scharff. The generation of antibody diversity through somatic hypermutation and class switch recombination. *Genes and Development*, 18:1–11, 2004.

[8] V. Giudicelli, D. Chaume, and MP. Lefranc. Imgt/v-quest, an integrated software program for immunoglobulin and t cell receptor v-j and v-d-j rearrangement analysis. *Nucleic Acids Research*, 32(2):W435–W440, 2004.

[9] MM. Souto-Camerio, NS. Longo, DE. Russ, Hw. Sun, and PE. Lipsky. Characterization of the human ig heavy chain antigen binding complementarity determining region 3 using a newly developed software algorithm, joinsolver. *Journal of Immunology*, 172(11):6790–6802, June 2004.

[10] Joseph M. Volpe, Lindsay G. Cowell, and Thomas B. Kepler. Soda: implementation of a 3d alignment algorithm for inference of antigen receptor recombinations. *Bioinformatics*, 22(4):438–444, 2006.

[11] L. Rabiner and B.H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, Jan 1986.

[12] Bruno A. Gaëta, Harald R. Malming, Katherine J.L. Jackson, Michael E. Bain, Patrick Wilson, and Andrew M. Collins. ihmmune-align: hidden markov model-based alignment and identification of germline genes in rearranged immunoglobulin gene sequences. *Bioinformatics*, 23(13):1580–1587, 2007.

[13] S Munshaw and TB Kepler. Soda2: a hidden markov model approach for identification of immunoglobulin rearrangements. *Bioinformatics*, 26(7):867–872, 2010.

[14] Katherine J. L. Jackson, Scott Boyd, Bruno A. Gaëta, and Andrew M. Collins. Benchmarking the performance of human antibody gene alignment utilities using a 454 sequence dataset. *Bioinformatics*, 26(24):3129–3130, 2010.

[15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[16] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[17] Timothy L Bailey. Dreme: Motif discovery in transcription factor chip-seq data. *Bioinformatics*, 27(12):1653–1659, 2011.

[18] T Kudo. Crf++: Yet another crf toolkit. 2005.

Appendix

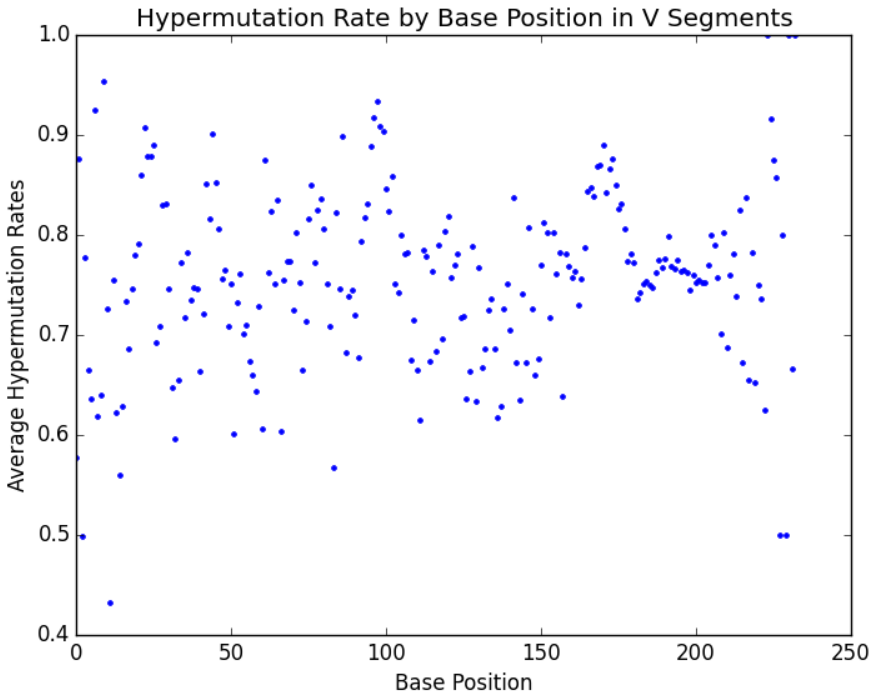

Figure A1

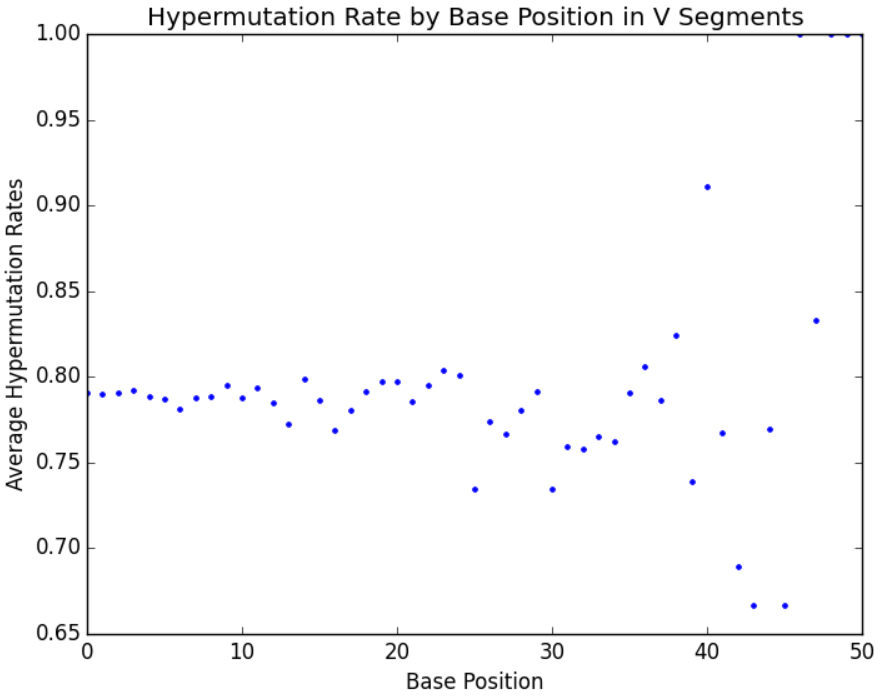

Figure A2

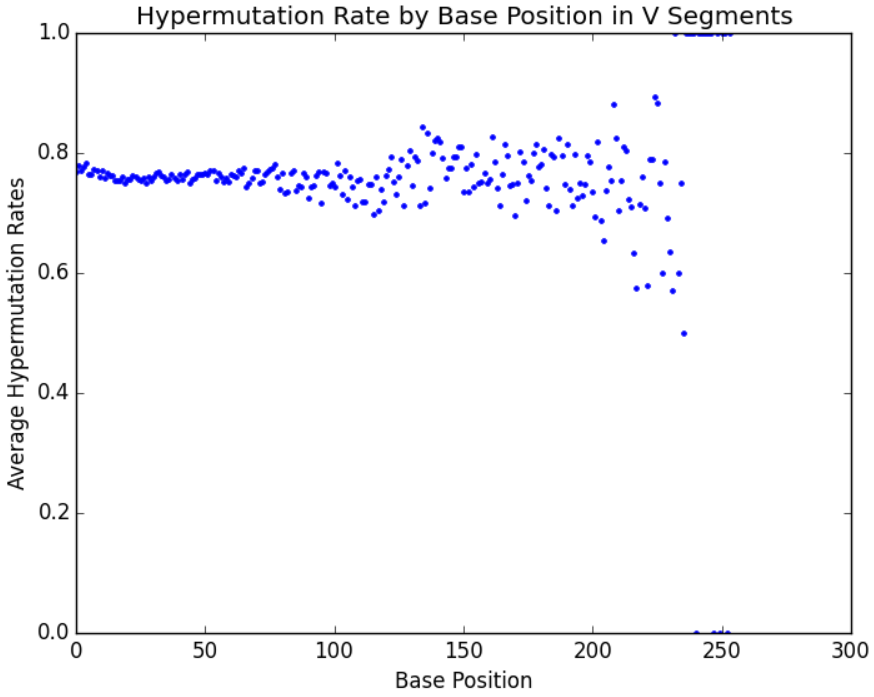

Figure A3

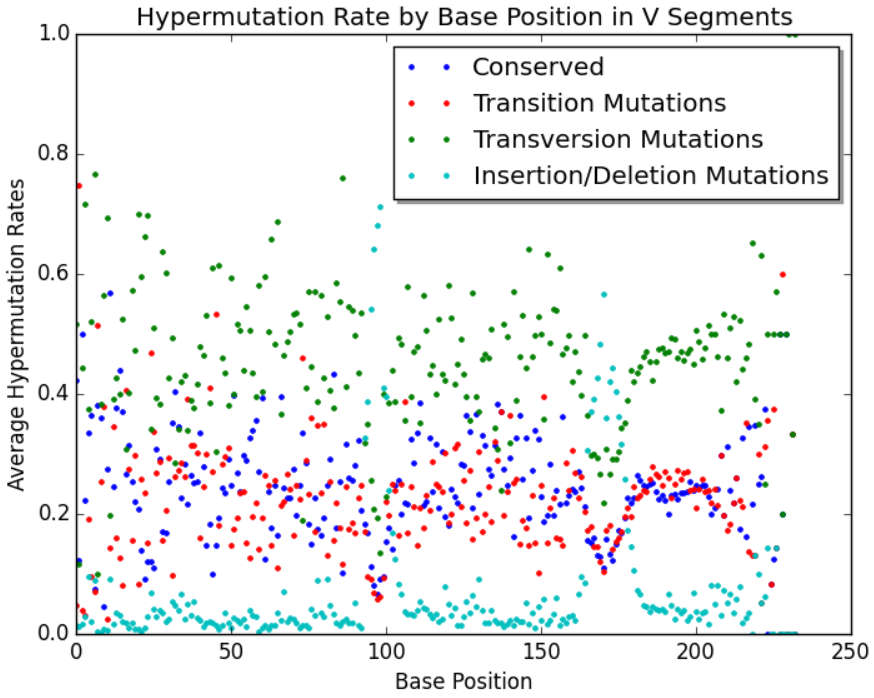

Figure A4

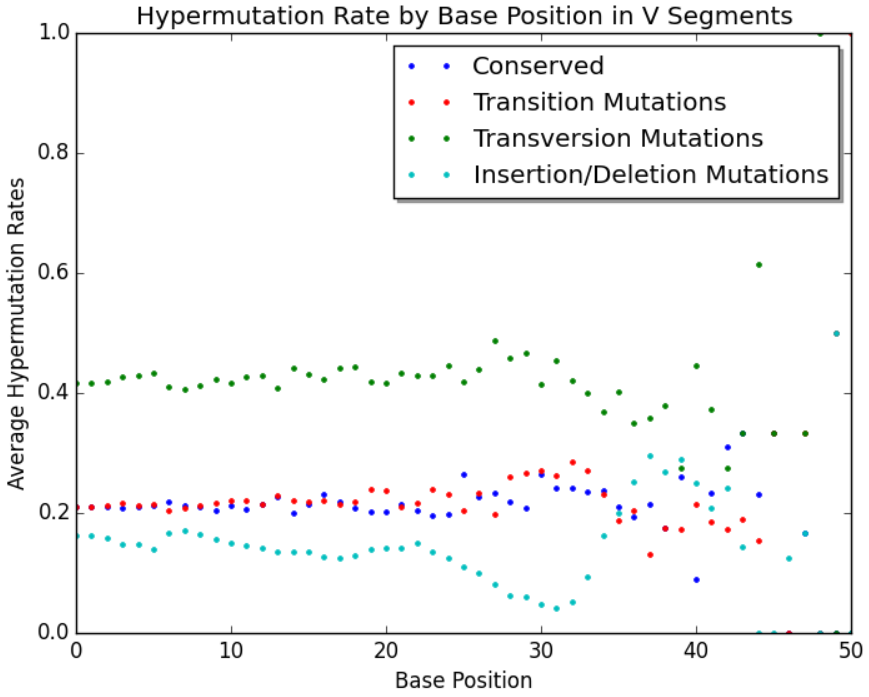

Figure A5

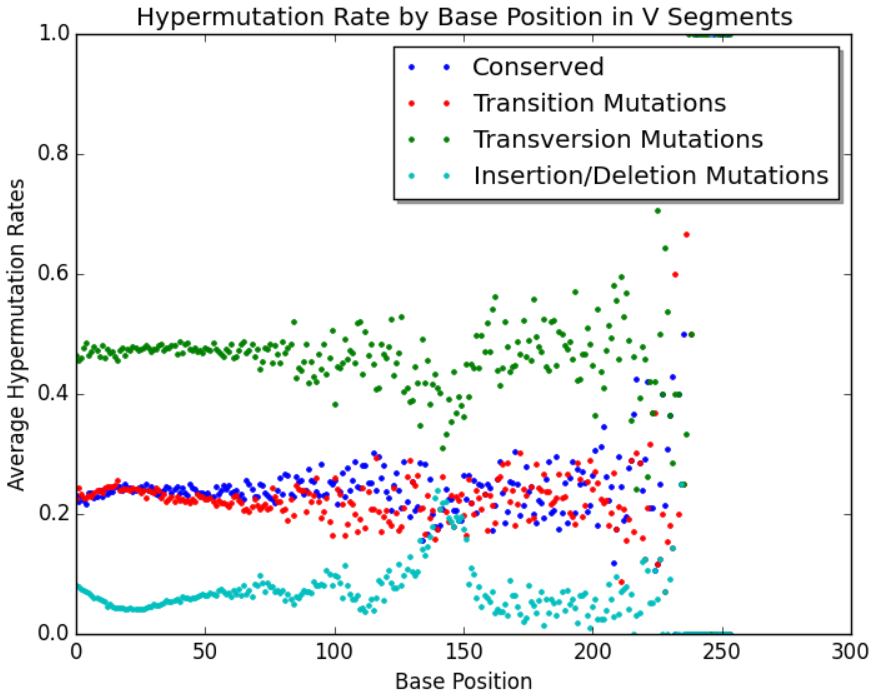

Figure A6

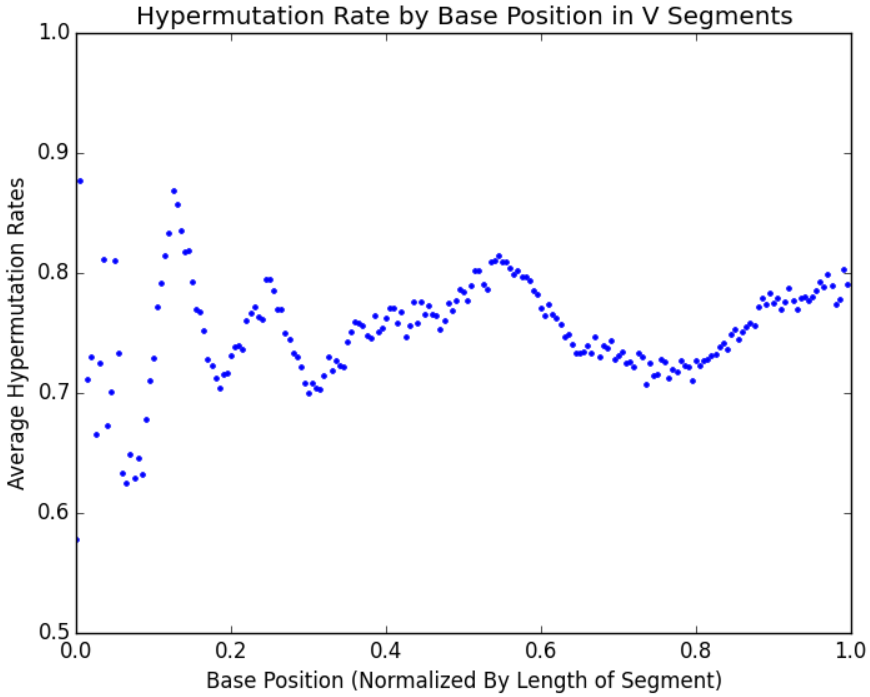

Figure A7: Hypermutation Rate by Base Position in V Segments

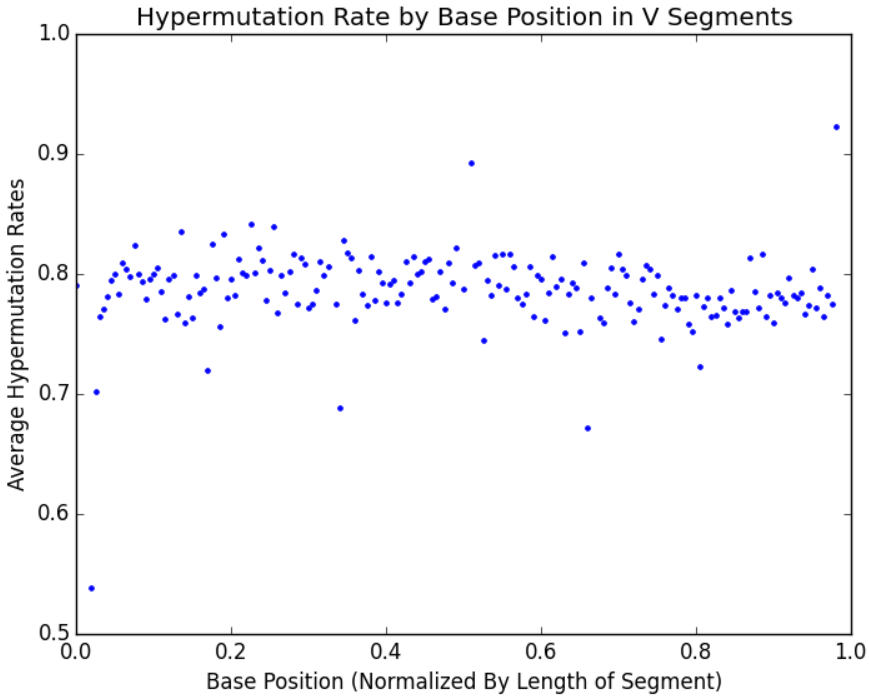

Figure A8: Hypermutation Rate by Base Position in D Segments

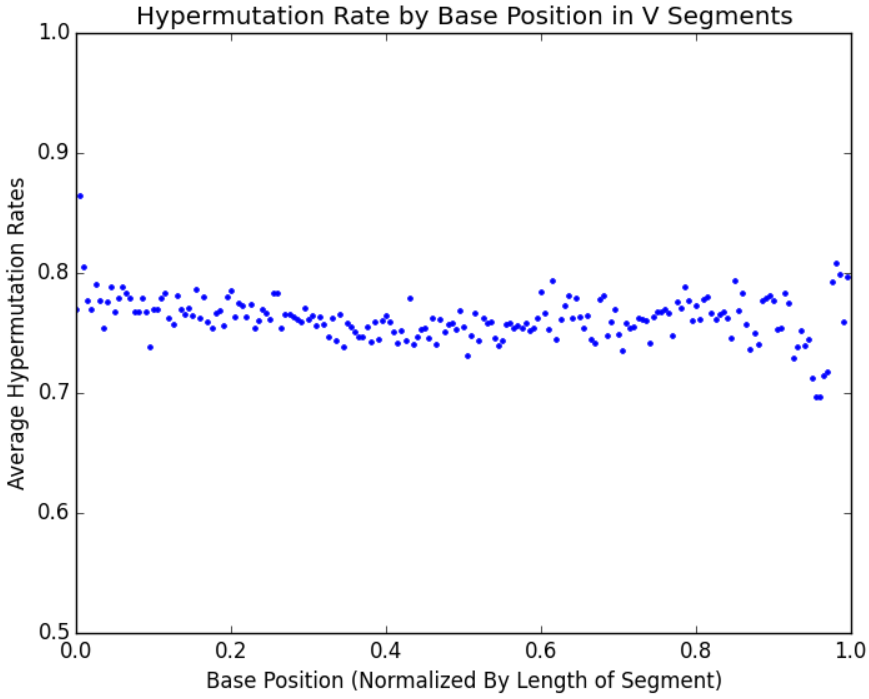

Figure A9: Hypermutation Rate by Base Position in J Segments

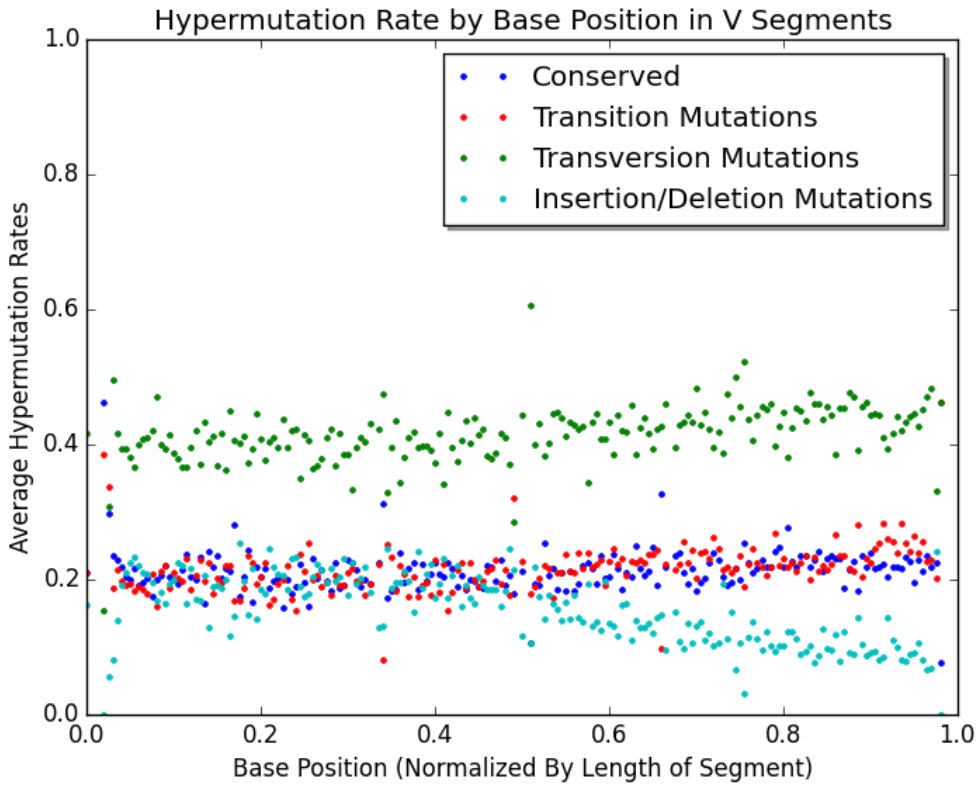

Figure A10: Hypermutation Rate by Base Position For each Type of Mutation (D Segments)

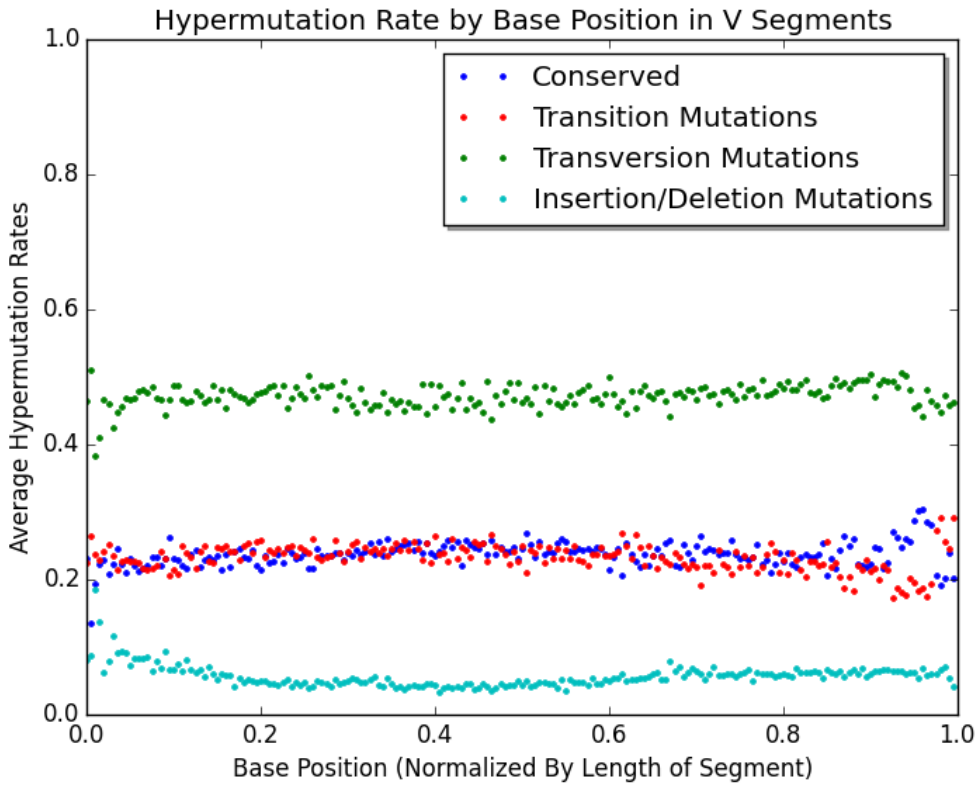

Figure A11: Hypermutation Rate by Base Position For each Type of Mutation (J Segments)

Length of Window	Segment Type	Mean	Std. Dev.
1	V	0.750837	0.430564
1	D	0.785665	0.357143
1	J	0.761116	0.418797
2	V	0.750174	0.311623
2	D	0.786614	0.248460
2	J	0.760662	0.298921
3	V	0.751143	0.254508
3	D	0.785772	0.192271
3	J	0.760675	0.241752
4	V	0.750762	0.221632
4	D	0.786414	0.160262
4	J	0.761069	0.207417
5	V	0.750814	0.198776
5	D	0.785617	0.135091
5	J	0.760882	0.184788
6	V	0.750466	0.182481
6	D	0.786130	0.116777
6	J	0.761359	0.166865
7	V	0.750957	0.169593
7	D	0.785709	0.100677
7	J	0.761178	0.153341
8	V	0.750631	0.159019
8	D	0.786300	0.088396
8	J	0.761246	0.141946
9	V	0.750959	0.149849
9	D	0.785855	0.075827
9	J	0.761050	0.132611

Table A1: Statistical Analysis of Running Averages

Motif	e-Value
	4.2e-972
	2.7e-414
	7.5e-301
	2.1e-199
	4.4e-051
	2.6e-043
	7.1e-015
	5.2e-016
	9.2e-015
	2.6e-013
	5.4e-009
	7.3e-008
	3.4e-007
	3.4e-003

Table A2: Nucleotide Motifs Found to be Enriched in Non-Hypervariable Regions

ACADEMIC VITA
BENJAMIN J. LENGERICH

EDUCATION

PENNSYLVANIA STATE UNIVERSITY, Schreyer Honors College
B.S., Computer Science and B.S., Mathematics 2015
Thesis: On the Origin of Sequences: Computational Analysis of Somatic Hypermutation for Probabilistic Immunoglobulin Predecessor Identification

SELECTED HONORS

SCHREYER ACADEMIC EXCELLENCE SCHOLARSHIP 2011-2015
EBERLY COLLEGE OF SCIENCE BRADDOCK SCHOLAR 2011-2014
UNDERGRADUATE SUMMER DISCOVERY GRANT 2012
PRESIDENT'S FRESHMAN AWARD 2012
BAUSCH AND LOMB HONORARY SCIENCE AWARD 2011

PUBLICATIONS

Experimental and Computational Mutagenesis To Investigate the Positioning of a General Base within an Enzyme Active Site. Jason P. Schwans, Philip Hanoian, Benjamin J. Lengerich, Fanny Sunden, Ana Gonzalez, Yingssu Tsai, Sharon Hammes-Schiffer, and Daniel Herschlag. *Biochemistry* **2014** 53 (15), 2541.

PRESENTATIONS

PENN STATE UNDERGRADUATE EXHIBITION 2015
PENN STATE UNDERGRADUATE EXHIBITION 2012
EBERLY COLLEGE OF SCIENCE RESEARCH SYMPOSIUM 2012

RESEARCH EXPERIENCE

RESEARCH ASSISTANT
Advanced Laboratory for Information System and Analysis 2014-present
- Applying machine learning to characterize somatic hypermutation and predict predecessor genes of mature antibodies.

RESEARCH ASSISTANT
Hammes-Schiffer Theoretical Chemistry Lab 2012-2014
- Implemented computational analysis of classical molecular dynamics simulations to investigate the effects of flexibility on an enzyme's catalytic rate.

INDUSTRY EXPERIENCE

SOFTWARE ENGINEERING INTERN
Google Summer 2014
- Designed and implemented a new targeting method, using machine learning to quantify the localization tendencies of websites.

SOFTWARE ENGINEERING INTERN
Synthetic Environment Applications Lab Summer 2013
- Parallelized graph connectivity algorithms for social network analysis, increasing the prohibitive network size for centrality calculations by a factor of 10^3.

SERVICE AND LEADERSHIP

PENN STATE DANCE MARATHON MORALE COMMITTEE 2012-2015
PNC LEADERSHIP ASSESSMENT CENTER 2014
PENN STATE SCHREYER HONORS COLLEGE STUDENT COUNCIL 2011-2013
PEDIATRIC CARDIOLOGY TRANSLATOR 2011