ABSTRACT

Title of Dissertation: HIGH-THROUGHPUT SEQUENCING CHARACTERIZATION OF DNA CYCLIZATION, WITH APPLICATIONS TO DNA LOOPING

Jason Matthew Hustedt, Doctor of Philosophy, 2019

Dissertation directed by: Associate Professor Jason D. Kahn, Department of Chemistry and Biochemistry

DNA flexibility is important both for fundamental biophysics and because DNA flexibility affects DNA packaging and regulation of gene expression through DNA looping. DNA flexibility has been studied with experiments ranging from biochemical ring closure or DNA looping experiments to AFM, crystallography, and

tethered particle microscopy. Even so, the flexibility of DNA *in vitro* and *in vivo* remains controversial.

In an attempt to resolve this controversy, we have developed a high-throughput, internally controlled, comparative ligation methodology using a library constructed of 1023 distinct DNA sequences ranging in length from 119 to 219 base pairs via ligation of pools of synthetic DNA of different lengths and PCR. The design incorporated barcoding for redundant identification of each molecule, allowing for a ligation reaction to be performed on the entire library in one reaction mixture. Two DNA concentrations were used in separate reactions to promote either unimolecular cyclization or bimolecular ligation and thereby explore a wide range of cyclization efficiencies (J factors). Half of each reaction mixture was treated with BAL-31 to destroy non-cyclized molecules. All products were linearized by restriction digestion and Illumina indices were added. The initial library and reaction mixtures were sequenced in a single Illumina MiSeq run.

From roughly 15 million assembled reads, over 13 million were identified using software written to identify and sort our sequence library. Each molecule was counted for each condition. From our analysis we see no evidence of extreme bendability at short DNA lengths. At higher DNA concentrations where bimolecular products are produced more rapidly, we see oscillatory behavior as a function of length. In contrast, at lower concentrations where unimolecular products dominate, we observe no helical variation due to the ability for all molecules to cyclize given enough time. In order to determine J factors through cyclization, bimolecular products must also be counted. Given the constraints of this experiment, not all

bimolecular products could be observed. Future experimentation can be performed to determine J factors across this size range, the results of which will improve coarse grain modeling of DNA. Extension of this methodology should be applicable to DNA loops anchored by proteins.

HIGH-THROUGHPUT SEQUENCING CHARACTERIATION OF DNA CYCLIZATION, WITH APPLICATIONS TO DNA LOOPING

by

Jason Matthew Hustedt

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctorate of Philosophy
2019

Advisory Committee:

Associate Professor Jason D. Kahn, Chair/Advisor

Distinguished University Professor Catherine Fenselau

Professor Najib M. El-Sayed

Associate Professor Douglas A. Julin

Associate Professor Jose Helim Aranda-Espinoza, Dean's Representative

Dedication

To my family; my partner, Connor; and my dog, Pike, for their unwavering support and patience that have made this possible.

Acknowledgements

For welcoming me into his lab and providing guidance through this process I would like to thank Dr. Jason Kahn, my thesis mentor and advisor. Without his continual guidance this process would not have been possible.

I am grateful to Dr. Ashton Trey Belew for generous advice on sequence analysis and to Dr. Najib El-Sayed for advice on planning of experiments and for access to computing resources at the University of Maryland Institute for Advanced Computer Studies.

I would like to thank Dr. Catherine Fenselau for her advice and guidance throughout my graduate career and for access to mass spectrometer resources in her laboratory when we wished to verify the presence of our proteins after gels proved inadequate.

I am grateful to Dr. Philip DeShong for helping to provide focus to both my research and my graduate career.

I would like to thank Dr. Robert Manning and Dr. John Maddocks for their insight into coarse grain modeling of DNA.

I am grateful to Dr. Claire Fraser, Luke Tallon, and Lisa Sadzewicz at the Institute for Genome Sciences at the University of Maryland School of Medicine for their help in answering questions and working through our non-traditional sequencing sample preparation.

I am grateful to Dr. A. James Mixson at the University of Maryland, Baltimore for funding and discussions on work with the histidine-leucine peptide project.

I am grateful to Raymond Peterson and Celadon Labs for financial support to myself and materials.

All of the members of the Kahn research group past and present who have offered help with experimentation and companionship over the years. I would particularly like to thank Brandon D. Ng who worked directly with me in the development of this project and Dr. Daniel B. Gowetski who taught me many of the skills necessary for my graduate career after I joined the Dr. Kahn's lab.

Finally, I would like to thank my friends and family for their continual support through this process. I would especially like to thank my partner Connor without whom I would not have been able to complete this process.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AFM	atomic force microscopy
bp	base pair(s)
dATP	deoxyadenosine tri-phosphate
DNA	deoxyribonucleic acid
dsDNA	double stranded DNA
EDTA	ethylenediaminetetraacetic acid
HTS	high-throughput sequencing
IDT	Integrated DNA Technologies
LacI	lactose repressor
Lk	linking number
LZD	leucine zipper dual-binding protein
MALDI	matrix assisted laser desorption/ionization
MCS	multiple cloning site
NEB	New England Biolabs
nt	nucleotide
ORF	open reading frame
PAGE	polyacrylamide gel electrophoresis
PCR	polymerase chain reaction
RNA	ribonucleic acid
siRNA	small interfering RNA
SC	size class
SV	sequence variant

SV-C	SV cyclization side oligonucleotide
SV-S	SV synthesis side oligonucleotide
TBE	Tris, borate, and EDTA (buffer)
TE	Tris and EDTA (buffer)
TEV	Tobacco Etch Virus nuclear-inclusion-a endopeptidase
TPM	tethered particle microscopy
Tw	twist
Wr	writhe

Chapter 1: Introduction

Section 1.1: DNA: The Basis of Genetics

One universal constant of all known life is the ability to pass on information from generation to generation allowing for beneficial traits to be kept and non-beneficial traits to be left behind. The discovery of DNA as the means of passing this information on gave birth to modern molecular biology. In order for life to perpetuate itself, the information stored in the DNA must be accessible, easily replicated, and stored in a manner that makes it available in every cell that needs it. The vital role of replication is made possible by the nature of DNA having two complementary strands of nucleotides. These two strands run in antiparallel order and are frequently twisted around one another with a relatively short helical repeat of 10.5 bp[1]. In the same year that Watson and Crick published their structure of DNA they also discussed the difficulties that such a structure imposed on accessing the information held within[2]. In order to access an individual strand the two strands must be separated, causing localized over-twisting of the DNA immediately surrounding the strand separation. The ability to alleviate this mechanical stress is handled by DNA gyrases and helicases.

In addition to structural changes to the DNA when accessed, there are local structural changes that are undergone in order to store DNA in cells. In bacteria the chromosomal DNA is circular and is compacted both through supercoiling of the DNA where it is wrapped further upon itself and through prokaryote specific packaging proteins such as the histone-like protein HU[3]. In some viruses DNA is

packaged in circular form. In eukaryotes chromatin is used to package genetic material in a more compact fashion than linear DNA itself allows. The human genome is just over 3 billion base pairs, if no compaction were used and the genome was stretched out each chromosome would be roughly two centimeters in length all together just exceed a meter in combined length.

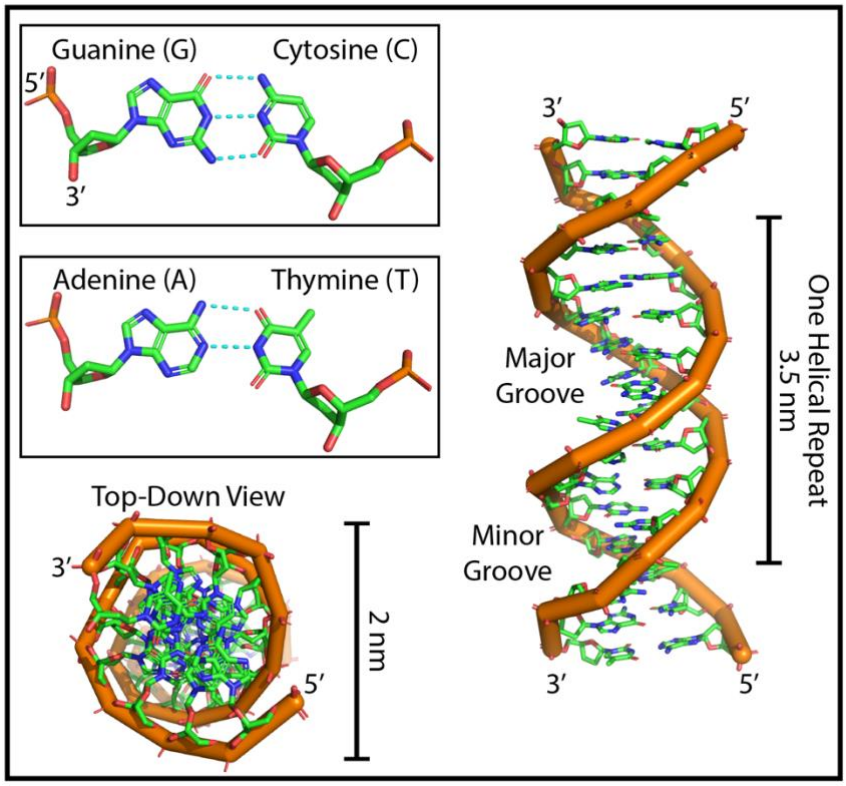

Figure 1.1 B-form DNA: The two canonical Watson-Crick base pairs and one and a half helical repeats of ideal B-form DNA double helix, with major and minor grooves labelled. Molecules rendered in PyMol v2.3.1 from the crystal structure 3BSE[4].

Section 1.2: Intrinsic Curvature and Protein Induced Bending of DNA

DNA has a high degree of conformational variability that is necessary for its function. This variability includes multiple higher order structures such as the G-quadruplex, DNA cruciform, and the direct bendability of the macromolecule itself.

Macromolecular bending can be classified into two main groups: intrinsic curvature of DNA and bending induced by proteins interacting with DNA. Within any particular sequence of DNA there is always a small degree of sequence dependent bending, however specific cases of dramatic sequence dependent bending have been observed to occur as well. The most well-known case of a DNA sequence preferring a particular bend is known as an A-tract. When spaced in phase with one another, a set of A-tracts ($A_6N_5A_6N_4$ etc.) can induce a piece of DNA that is bent much more than the maximum extent of bending expected from thermal fluctuations. In nature phased A-tracts have been seen to cause small circular regions of roughly 700 bp in length in kinetoplast DNA in trypanosomes[5].

DNA may also be deformed into a bent conformation through binding of a protein. Through the binding of DNA to nucleosomes it can be packaged neatly into the nucleus. The wrapping of DNA around the nucleosome relies upon the intrinsic flexibility of DNA but would not form the tightly wound form found around the nucleosome if it was not present. Both cAMP receptor protein and TATA-binding protein activate transcription of the genome based upon recognition of a specific DNA sequence[6–8]. This recognition can rely upon the flexibility of a short DNA fragment adjacent to or often within the sequence of the gene of interest. The lac operon on the other hand is repressed by the DNA binding protein LacI. While the binding site for LacI itself is not substantially bent, DNA looping of two lac operators induces a bend in the DNA between the two operators[9]. The bacteriophage 434 repressor protein also relies on changes to DNA conformation to repress transcription. In the 434 repressor protein binding to the 14 bp operator has been demonstrated to

not have any base interactions, instead the protein relies upon the twisting flexibility of a dATAT sequence in the center of the operator that allows the adjacent regions to properly fit into the binding pockets of the 434 repressor[10]. In all of these protein-DNA interactions, the intrinsic flexibility of the DNA allows for the protein to perform the necessary function. By better understanding DNA flexibility, better understanding of the protein-DNA complex is also achieved.

Section 1.3: Understanding DNA Long-Range Structure and Topology

Since the elucidation of the basic structure of the double helix, much work has been performed to better understand the shape of DNA as a macromolecule. At larger scales than the short linear DNA that was initially crystallized many concepts are necessary to define the overall topology of a given molecule. The most common structural parameter mentioned in reference to DNA by biochemists at large is its persistence length. The persistence length of DNA is a sequenced average estimation of how "rigid" a given molecule of DNA is. There is a consensus in the literature that the persistence length of DNA is roughly 150 bp or 50 nm[11]. At lengths shorter than the persistence length, DNA behaves like a rigid rod, at lengths greater than the persistence length DNA behaves like a random coil. One additional important concept for describing DNA is that of linking number (Lk). When a piece of DNA is closed upon itself into a circle and the respective strands are ligated to themselves, the twisting nature of DNA will have caused the backbones to have crossed over themselves a number of times. This wrapping of the DNA around itself is called twist. If the length of the DNA is a multiple of 10.5 bp, the helical repeat, the number of crossover nodes corresponds directly to the length of the DNA divided by the helical

repeat. A second type of crossover event that will occur is when the double helix crosses over itself. This type of crossover is referred to as writhe. Collectively the twist and writhe of a closed molecule are the linking number for that particular molecule (Eq. 1).

$$Lk = Tw + Wr \ (Eq. 1)$$

A second important concept for understanding closed circular DNA is the J factor. In 1950 Jacobson and Stockmayer developed the mathematical theory behind the ability for a linear polymer of a particular length to cyclize upon itself[12]. Their work established what is now known as the J factor, which is a ratio of the equilibrium constant for formation of a circular DNA molecule (k_c) to the equilibrium constant for bimolecular reaction (k_b). Ultimately this ratio represents the local relative concentration of one end of a DNA molecule to its other end when in an alignment capable of ligation (Eq. 2).

$$J = \frac{k_c}{k_b} \ (Eq. 2)$$

The most well-known application of the J factor came in 1981 and 1983 when David Shore and Robert Baldwin showed the impact of helical phasing on DNA cyclization. Using twelve molecules of varying lengths spread from 236-254 base pairs, an increase in J factor can be seen as the number of base pairs nears a multiple of the helical repeat (10.5 bp) yielding an oscillatory curve where the dependence of J factor is tied not only to the length of the DNA but to its helical phasing[13,14]. In the same year, Shimada and Yamakawa extended the work of Jacobson and Stockmayer to theoretically show the upper and lower bounds for J factor taking the helical repeat

of the DNA into account[15]. Simulated J factors from Shimada and Yamakawa's work across a range of lengths is illustrated in figure 1.2.

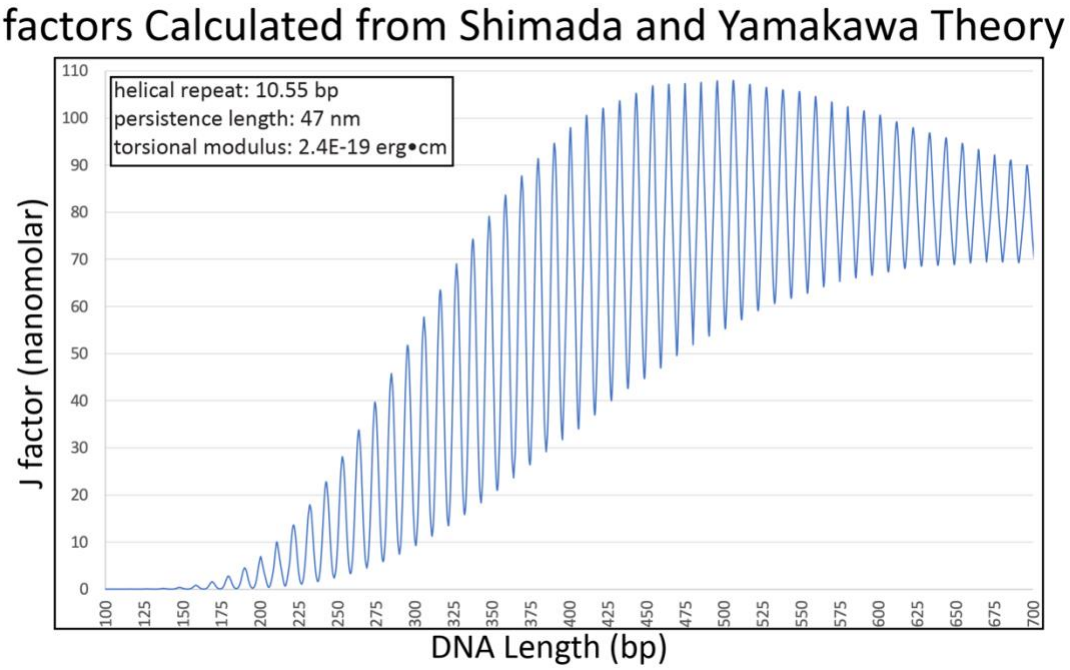

Figure 1.2 Shimada and Yamakawa Theory: J factors for DNA lengths ranging from 100 base pairs to 650 base pairs. To form a circular molecule the ends of the DNA must be in proper alignment for ligation, as the length in base pairs approaches a multiple of the helical repeat the J factor increases, indicating that the local relative concentration of the two ends to one another in a form that can ligate has increased. As lengths increase the upper and lower bounds for J factor increase until around 500 base pairs, after which the chances of cyclization start to decrease as a whole as length continues to increase due to entropy for a random chain.

From the mathematical work of Shimada and Yamakawa combined with the experimental work of Shore and Baldwin, the oscillatory nature of the J factor can be explained by the helical repeat of the DNA itself. When the length of the DNA is a multiple of the helical repeat, the two ends of the DNA align for each strand to be ligated back upon itself. As the length of the DNA diverges from a full helical repeat, simply bending the DNA into a curved surface will not in fact bring the appropriate

ends of the two strands together. In order for the DNA to cyclize, it must change conformation either undertwisting or overtwisting to line the appropriate backbones up to ligate, as seen in figure 1.3.

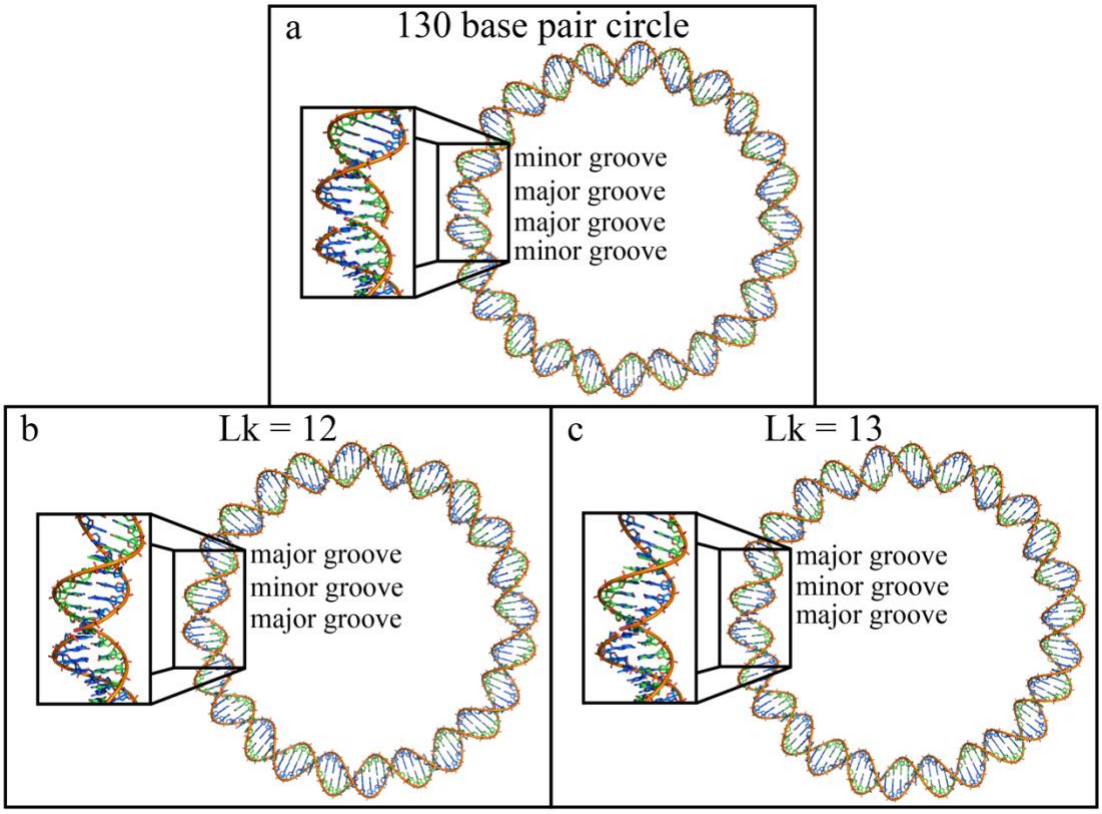

Figure 1.3 Representative cyclization of 130 bp DNA: A 130 base pair molecule bent to bring the two ends of the DNA molecule together. The appropriate backbones are mis-aligned in (a) preventing ligation in this conformation. In (b) and (c) the DNA is undertwisted or overtwisted, respectively, to bring the backbones into alignment to allow for cyclization. The overall change needed to the molecule to be in a position for this ligation event to occur is what causes the decrease in J factor relative to molecules where the DNA length is a multiple of the helical repeat. For this figure, ideal B-form DNA was modeled using the 3DNA open source software suite[16]. The DNA was constrained to coordinates generated to have the appropriate helical rise and twist using VMD[17].

Section 1.4: Experimentation to Determine DNA Flexibility

In order to better understand and define how DNA behaves, a number of different methods have been employed. One approach that has seen widespread use is DNA ring closure. In ring closure experiments linear DNA either participates in a cyclization reaction where cohesive ends come together to form a circular molecule. Initial ring closure assays visualized products through absorbance measurements during DNA sedimentation after shearing[18–20]. In more recent ring closure experiments, T4 DNA Ligase is used to capture the DNA in any form where cohesive ends have come together and polyacrylamide gel electrophoresis is used to analyze the captured molecules[13,21]. With the utilization of ligase, products are covalently captured, meaning more products can be obtained as linear multimers and dimer circles can form. The competition between intramolecular and intermolecular reactions results in an accurate means of determining the J factor as discussed above[15]. The appropriate concentrations of T4 DNA Ligase, DNA, and length of reaction are needed in order for a useful experiment to be performed. The most widely known experiments to determine J factors by Shore and Baldwin were performed using ligase catalyzed ring closure [13,14]. Modern fluorescence experiments have been used in bulk and single molecule studies to look at ring closure without the need for ligase[22,23]. There are many labs that have utilized ring closure experiments to identify J factors, notably the Crothers Lab, the Vologodskii Lab, the Widom Lab, and the Ha lab[23–28]. A further discussion on modern controversies in J factors is presented in chapter 2.

An application of DNA ring closure is DNA looping experiments, where cyclization methods are applied to protein-DNA complexes. In these methods the DNA is cyclized with and without the relevant protein, the impact of the protein on the DNA can be observed as a change in overall linking number[29]. Work performed on looped complexes has looked at the tied topological characteristics for many different protein-DNA complexes including the Lac repressor[9,30], transcription factor IIB in yeast[31], the human oncoprotein p53[32], and many other naturally occurring proteins[33]. Looping work has also been performed by our own lab using rationally designed dual DNA binding proteins in an attempt to separate the contribution of protein flexibility from long range looping interactions[34].

A relatively recent technique that is being used to probe DNA flexibility on DNA alone and in protein bound environments is that of tethered particle microscopy. In tethered particle microscopy (TPM) the DNA is tethered on one end to a surface and on the other end to a reporter particle. By observing the location of the reporter molecule the conformation of the DNA can be extrapolated. These experiments have been done on free DNA of varying length showing Brownian motion of the reporter particle as the DNA changes conformation on its own and on protein-DNA complexes to view how the binding of different transcription factors changes the conformation specific DNA sequences[35,36]. By looking at the differences in location of the reporter particle, TPM can observe the changes in DNA flexibility rapidly as the experimental conditions are changed via introduction of a DNA binding protein.

Chapter 2: Design, Creation, and Verification of a DNA Length Library

Section 2.1: Overview

For over twenty years following the publication of Shimada and Yamakawa's theory in 1984 there was a consensus that DNA on its own behaved in a manner that agreed with the published theory[15,27]. Many experiments were performed at various lengths and for each of these experiments, the theory held. Figure 2.1 shows the Shimada and Yamakawa theoretical J factors plotted on both linear and log scales. At DNA lengths greater than 165 bp the theoretical J factor is in excess of 1 nM when the two ends of the molecule are in phase with one another. Below this length the theory shows a rapid decrease in J factor. As length decreases, the effect of helical phasing also becomes more exaggerated as can be seen when the same results are graphed in log scale (figure 2.1 c & d), where it can be seen that the maxima and minima in J factor across a helical repeat differ by more than of two orders of magnitude.

In 2005 Cloutier and Widom published results showing that DNA fragments around 100 base pairs in length had cyclization efficiencies approximately three orders of magnitude above the expectations from Shimada and Yamakawa[15,27]. Following the work of Cloutier and Widom, the Vologodskii lab attempted to replicate Widom's work and was unable to do so, finding that their own 100 base pair fragments fell in line with existing theory[37]. Vologodskii identified one main problem with the Cloutier and Widom approach. In order for a ring closure experiment to

measure J factor, the rate of ligase binding a molecule which can serve as a substrate to ligase much be much less than the rate of substrate dissociation into its component parts. If the substrate is a monomer circle, this is the dissociation into linear monomers. If the substrate is a bimolecular junction, this is the dissociation into two linear precursors. Vologodskii found that the overhang used by Widom (GGCC) had a melting temperature about 20 °C higher than the overhang used in trying to replicate the work (AGCT), with this less stable overhang at the same ligase concentration J factor was seen to return to the theorized value[37].

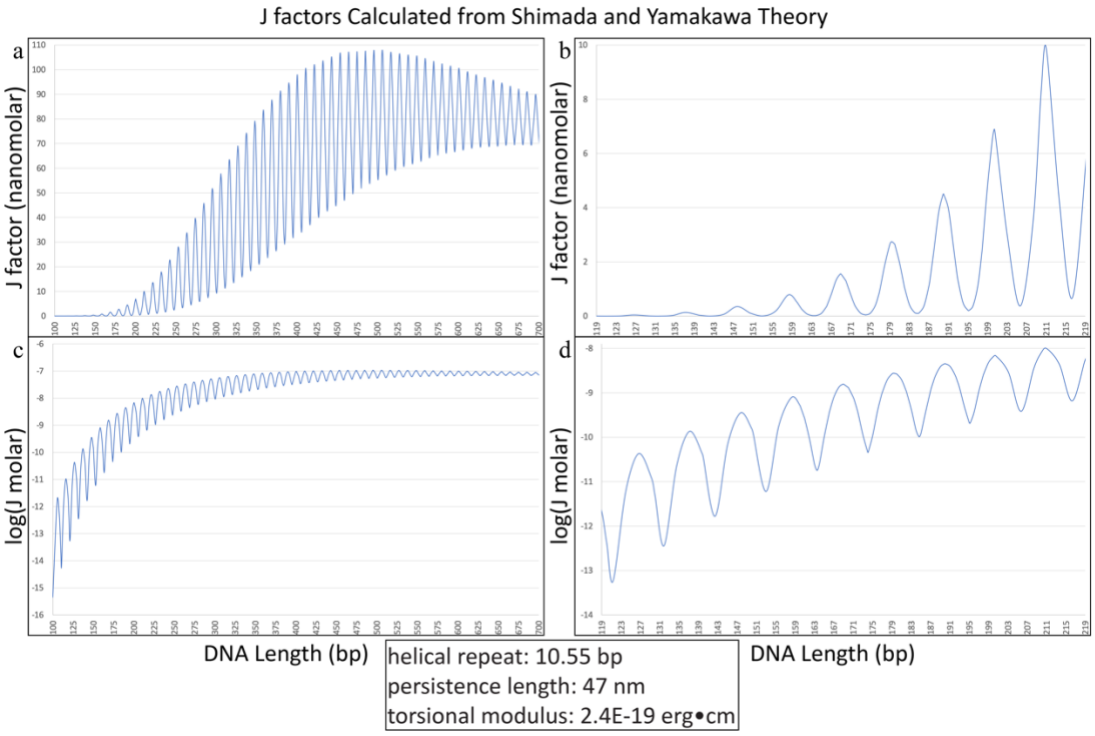

Figure 2.1 Linear vs Log Scale J factors from Shimada and Yamakawa Theory: a) Simulated J factors from the Shimada and Yamakawa theory for DNA, from 100 to 650 base pairs in length. The J factor as a function of length oscillates across each helical repeat; b) a focused range of (a) looking at 115-215 base pairs, the J factor can be seen dropping below 1 nanomolar for lengths less than 165 base pairs; c) the J factors from (a) presented in a log scale: as the length of a fragment decreases the difference between maxima and minima in J factor across a helical repeat become more pronounced; d)

a focused section of (b) looking at 115-215 base pairs. In (b) and (d) it is apparent that the modulation in J factor is a sum of Gaussian curves for each linking number with the maxima separated by the helical repeat length.

Following Widom's work, Taekjip Ha has observed extreme bendability in DNA ranging from 67-106 bp. Using fluorescence-based assays Ha is able to observe the rate of DNA looping without the presence of ligase[23]. In this work J factors are calculated roughly three orders of magnitude greater than those theorized by Shimada and Yamakawa. This is the same range of J factors observed by Widom in ~100 bp circles. Work performed by Vologodskii has demonstrated that if these J factors in the 100 bp region are accurate that an increase should be seen in the 200-300 bp range as well where DNA bending dominates the J factor. However, numerous experiments have shown alignment with Shimada and Yamakawa theory at these lengths[21,24,34,37].

Section 2.2: Experimental Aims

Given the controversy that exists surrounding the validity of the Shimada and Yamakawa theory at low DNA lengths and the differences in available experimental data, we set out to generate a DNA length library spanning from below the size range where we expect to see cyclization into the length ranges where in there remains a consensus on the applicability of established theory. To achieve this, we decided to look at every length from 119 base pairs to 219 base pairs. In addition to generating a length library, we set out to generate a sequence library at every possible length to prevent any sequence dependent intrinsic curvature from biasing apparent cyclization efficiency for that length. To obviate difficulties inherent to previous work with the absolute measurement of J factors, the goal in this experiment was to perform ligation

on the entire library in one tube and sort out the product distribution via high-throughput sequencing.

Section 2.3: Sequence Design

The combined DNA length and sequence library was designed to incorporate three regions of variable length within each molecule such that the library could be constructed in a manner that simplified downstream analysis via high throughput sequencing, by indexing each molecule. A full discussion of the computational separation of indices is presented in chapter 3. The individual sequences within each of these three regions was carefully considered prior to construction of any molecules. Each member of the library was generated as two sides, the sequence variant (SV) and size class (SC) sides, then combined into the final library. These two sides include three overall size class variants and 341 length/sequence variants. An overall schematic for the generation of the sequence variants via mutually primed extension is shown in figure 2.2. After generation of individual components all molecules were cleaved at both the XhoI and BstEII (cyclization junction and synthesis junction) sites and gel purified as shown in figure 2.3.

The names from the two classes to identify each size are in the order of size class, SV0-30 insert, SV0-10 insert (e.g. 0470309 would be a molecule from the 47 size class with 3 and 9 base pair insertions in the 0-30 and 0-10 insertion regions respectively). Pre-assembled sequence variant molecules are named in the same order without the size class (e.g. SV1203 is a sequence variant molecule containing a 12 base insert in the 0-30 insert region and a 3 base insert in the 0-10 insert region).

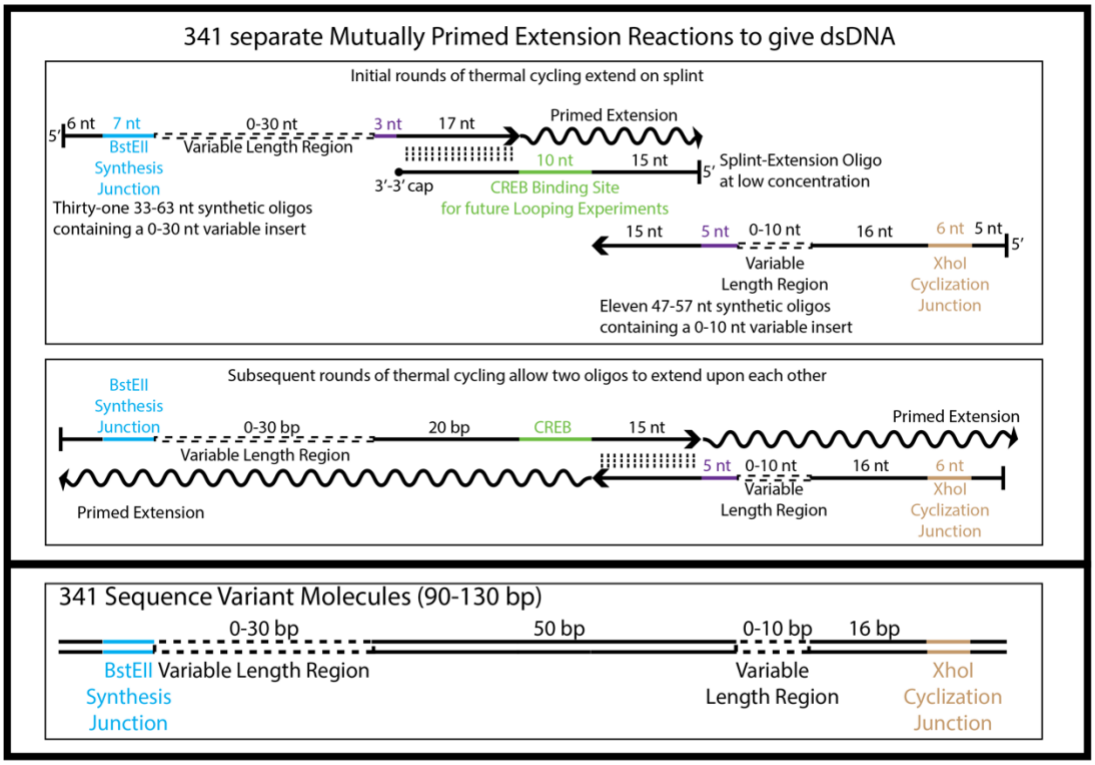

Figure 2.2 Sequence variant generation: A schematic of the generation of the sequence variant (SV) side of the length library. Three oligos were combined in each of 31x11=341 individual tubes. During the first round of thermal cycling a primed extension occurs where the synthesis side variable oligo (at the left in the figure) extends upon the splint molecule. In subsequent rounds of extension, the cyclization side oligo (on the right in the figure) and synthesis side oligo will extend upon one another, resulting in a single dsDNA in each tube that includes one of each of the variable length regions. Individual SV molecules are named in the direction shown here with synthesis side insert number followed by cyclization side insert number (e.g. SV1507 has a 15 bp insert in the 0-30 insert region and a 7 bp insert in the 0-10 insert region).

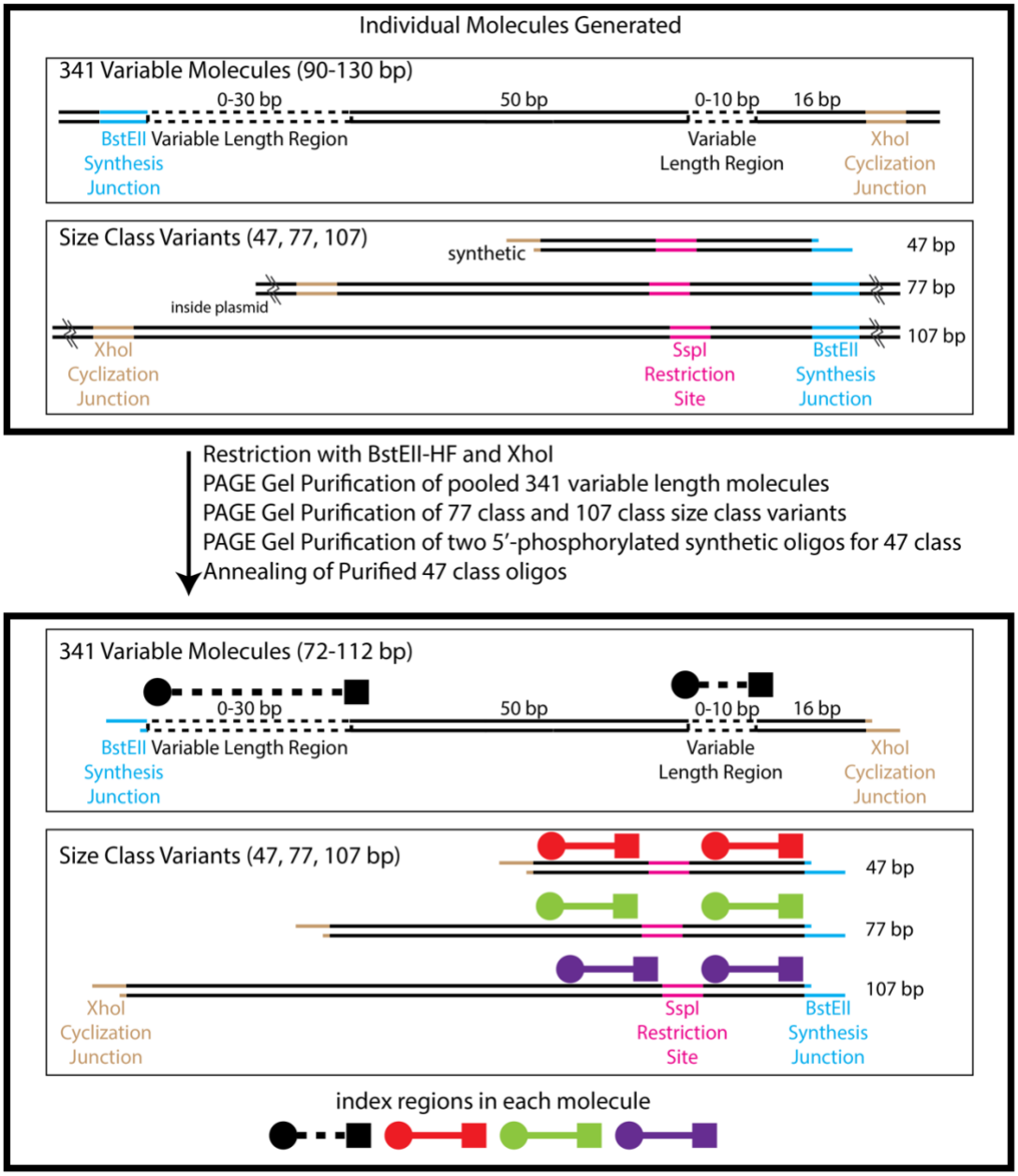

Figure 2.3 Isolation of library sides: A schematic showing all individual library molecules after construction, cleavage, and purification. Following construction of each library molecule purification was performed utilizing the appropriate Qiagen kit. Molecules synthesized via mutually primed extension were prepared for restriction digestion using a Qiagen PCR clean-up kit and plasmids containing size class variants were purified using a Qiagen Midiprep kit.

Section 2.3.1: Sequence Variant Design

The sequence variants were constructed in 341 separate mutually primed extension reactions. In each of these reactions three separate oligonucleotides were included: a synthesis side oligo including a variable length insert of 0-30 base pairs in length, a cyclization side oligo including a variable length insert of 0-10 base pairs, and a splint oligo. On either end of each library molecule is a BstEII or XhoI restriction site. The BstEII restriction site was chosen as an enzyme that has no native cut sites in pUC19 that after restriction leaves an asymmetric five base overhang, allowing for assembly of sequence variant side and size class sides while minimizing self-ligation between a given side. The BstEII overhang is interchangeably referred to as the synthesis junction. The XhoI restriction site was chosen as and is referred to interchangeably as the cyclization junction. The choice of XhoI as the cyclization junction is due to the overhang present after restriction (AGCT). This junction has been used in previous work and is selected to allow for the rate of ligatable substrate disappearance to be greater than the rate of ligase binding in assayed conditions[34,37]. The cyclization side oligos also contain a priming site incorporated such that if a large library insert is present on the cyclization side a second read starting at this location could read just the variable insert data. This priming site was not necessary in this analysis but is present to make the library extensible to a larger number of applications.

The synthesis side oligos were composed of the following segments, from left to right 5'-3': a 6 nucleotide leading sequence present to assure that the BstEII cut site was not at the end of the final double stranded product; the 7 nucleotide BstEII

recognition sequence (GGTCACC); a variable insert of 0-30 base pairs in length, which allows for coverage of every length in the desired range (detailed in section 3.2); a 3 nucleotide sequence inserted to prevent any differences in annealing efficiency from causing different concentrations to result from the mutually primed extension reaction; and a 17 nucleotide splint complement region.

The cyclization side oligos were composed of the following segments, from left to right 5'-3' (shown right to left in figure 2.2): a 5 nucleotide leading sequence present to assure that the XhoI cut site was not at the end of the final double stranded product; the 6 nucleotide XhoI recognition sequence (CTCGAG); a cyclization independent priming site; a variable insert of 0-10 base pairs in length, which allows for multiple sequence combinations to exist at every given length (detailed in section 3.2); a 5 nt sequence intended to prevent differences in annealing efficiency during the mutually primed extension from effecting the experiment; and a splint 15 nt complement region.

With the synthesis side and cyclization side oligos above the total length of each side was quite large (33-63 nt and 47-57 nt respectively). Due to the decreased efficiency of oligonucleotide synthesis at lengths much greater than this, we decided to rely upon a splint oligo to allow us to incorporate the region in between these two oligos. This splint molecule is the same in all 341 sequence variant reactions, yielding a common sequence region in all 341 products. This common sequence contains a CREB binding site, included to make the project extensible to potential future DNA looping work.

The splint molecule was composed of the following segments, from left to right 5'-3' (shown right to left in figure 2.2): a direct sequence match to the cyclization side splint complement region; the CREB binding site; the reverse complement of the synthesis side splint complement region; and a 3'-3' cap to prevent the splint molecule from being itself extended, serving as a template for the synthesis side oligo.

Section 2.3.2: Sequence Variant Indexing

In designing the sequence variant library, we wished to index the inserts in a manner that would allow us to know which insert was present even if the entire insert was not sequenced. To this end, within both variable inserts the same nucleotide barcoding schematic was implemented. Each of the four nucleotides was assigned a two bit numerical value: A-00, C-01, G-10, and T-11. The appropriate set of nucleotides was incorporated to identify the total length of the insert in six bit binary. To prevent the three variable bits from being immediately adjacent they were separated by two non-index common nucleotides, as in NnnNnnN, where N is a length coding nucleotide and n is a filler non-index nucleotide. For any variable sequence shorter than 7 nucleotides, leading zeros were excluded. For example, while the six bit binary for the number one is 000001, only the final two bits: 01 are coded into the sequence with the incorporation of a C. Sample barcoding shown in table 2.1.

SV 0-30 Insert Length	SV 0-30 Insert Sequence
2	Ga
5	CatCa
8	AatGagA
11	AatGagT
16	CatAagA
23	CatCagT
30	CatTagG

Table 2.1 Example sequence variant barcoding: A list of select sequence variant insert lengths with the corresponding barcoded insert.

Section 2.3.3: Size Class Variant Design

As the synthesis side oligos of the SV class include a 0-30 base pair insert, to span the desired range of sizes, the three size class (SC) variants 30 base pairs apart in length were required. These three variants were 47 bp, 77 bp, and 107 bp in length. hereas the sequence variants were generated via mutually primed extension from synthetic oligos, the size class variants were designed to be generated using a gBlock (IDT). Upon cloning this gBlock into pUC19, each of the three size class variants can be excised from a single plasmid in a single reaction, ensuring equivalent yields and presumably similar dead fractions of non-ligatable DNA ends. For each of the three size classes, an SspI restriction site was incorporated near the synthesis junction side of the molecule allowing enough sequence to be present on either side of the SspI cut side for the full identity of the pre-cleaved molecule to be determined after cleavage.

The 47 class fragment could not be purified as discussed later in this chapter, instead the 47 class fragment was generated as two 5’ phosphorylated oligonucleotides from IDT. These two oligos contained the appropriate BstEII or XhoI overhang at the start of their sequence such that when annealed the dsDNA product would not require subsequent restriction and could be used as is.

Section 2.4: Library Fragment Construction

All enzymatic reagents were purchased from New England Biolabs, PCR Purification and Nucleotide cleanup kits were purchased from Qiagen, DNA oligonucleotides and the gBlock were synthesized by IDT, and all other reagents were purchased from Fisher Scientific. An MJ Research PTC-200 Gradient thermocycler was used for PCR reactions.

Section 2.4.1: Sequence Variant Construction Methodology

Each of the synthesis side oligos was synthesized by IDT using the sequence design in section 2.3. Initial trials on the synthesis side oligos allowed only for extension upon the splint oligo. In each trial PCR reactions were performed in 50 µL reactions as follows: 1 Unit NEB Phusion polymerase, 4 µM synthesis side oligo, and 1 mM each dNTP. Both 1x NEB Phusion HF and 1x NEB Phusion GC buffers were used across a range of annealing temperatures from 50.3 to 59.8 °C (figure 2.4). The thermal cycler protocol was as follows:

Initial denaturation: 95 °C

Cycle (35 repeats):
1. 95 °C for 1 minutes
2. Anneal for 25 seconds (tested temperatures: 50.3, 52.7, 56, 59.8 °C)
3. 72 °C for 30 seconds

From this work it was determined that NEB Phusion HF buffer and an annealing temperature near 60 °C were optimal for the initial extension. Selected samples from this gel were run again and are shown in figure 2.5. After determining the appropriate buffer and annealing temperature, trials were performed +/-Mg^{2+} (2 mM) and with varying splint concentrations from 0.032 to 0.08 µM splint (figure 2.6).

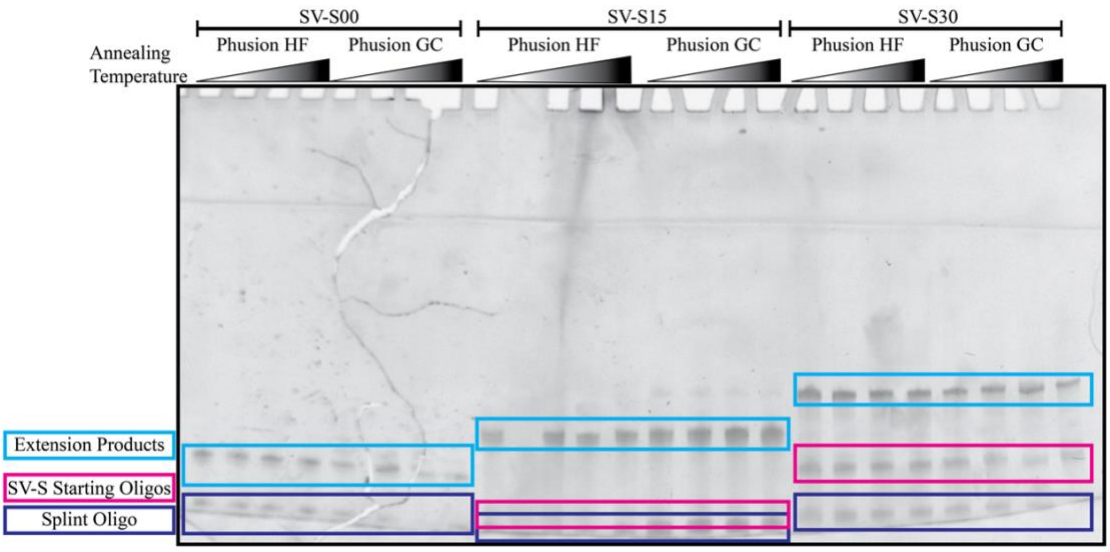

Figure 2.4 SV primed extension buffer test: Denaturing 8% poly-acrylamide gel (40:1) of extension products from selected Synthesis Side (SS) variant molecules on the Splint Oligo. SS00 is not visible in this gel and SS15 is overlapping with the splint oligo. No considerable difference is seen between HF and GC Phusion buffers. All tested annealing temperatures (50.3, 52.7, 56, and 59.8 °C) resulted in similar quantities of product from the extension reaction.

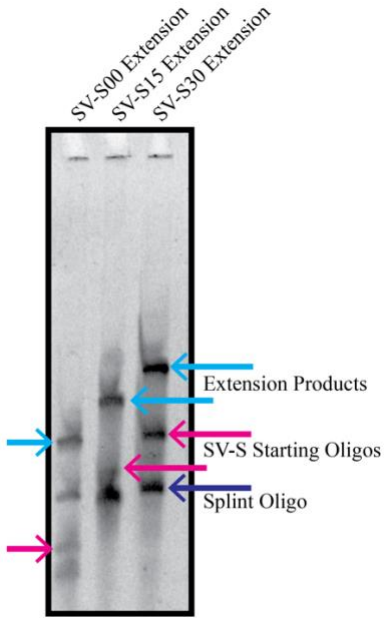

Figure 2.5 SV primed extension annealing temperature test: Extension products from sequence variant synthesis side oligos: SV-S00, SV-S15, and SV-S30 on the splint oligo using NEB Phusion HF buffer

and 59.8 °C annealing temperature. The starting product can be seen in pink, with the splint oligo in purple and final extension products in blue.

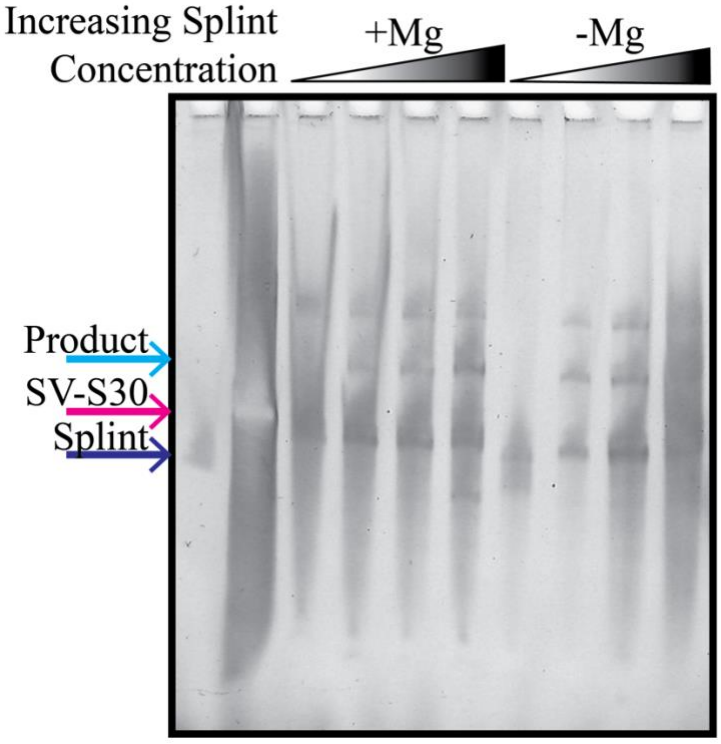

Figure 2.6 SV primed extension Mg^{2+} test: Extension products from sequencing variant synthesis side oligo SV-S30 on the splint oligo at 0, 0.0032, 0.016, and 0.08 µM splint. Each reaction was performed in Phusion HF buffer with an annealing temperature of 59.8 °C with and without 2 mM Mg^{2+}. The SV-S30 oligo was present in large excess while determining the minimum amount of splint needed to yield products. In this excess a second self-annealed product can be seen above the desired product.

After conditions were determined for optimal extension the extension was verified on all 31 individual extension reactions and analyzed by polyacrylamide gel electrophoresis. By design, during the first phase of thermal cycling the synthesis side oligo and splint oligo anneal and the synthesis side oligo extends upon the splint oligo, generating the reverse complement of the cyclization side oligo. As such, during subsequent cycles of thermal cycling the extended synthesis side molecule and

cyclization side oligos will both extend upon one another generating full length double stranded DNA. The final thermal cycling protocol was as follows:

1. Initial denaturation: 95 °C 5 minutes
2. 95 °C 1 minute
3. 63 °C 1 minute
4. 72 °C 1 minute
5. Goto step 2 (4 times)
6. 72 °C 3 minutes
7. 95 °C 1 minute
8. 43 °C 1 minute
9. 72 °C 2 minutes
10. Goto step 7 (4 times)
11. 72 °C 3 minutes
12. 95 °C 30 seconds
13. 55 °C 1 minute
14. 72 °C 2 minutes
15. Goto step 12 (29 times)
16. 72 °C 5 minutes

This protocol was performed in 341 individual tubes with each of the possible combinations of the 31 synthesis side oligos and 11 cyclization side oligos. Each reaction also contained the splint oligo and was individually purified after thermocycling using a Qiagen PCR Cleanup Kit. After purification a portion of each reaction was pooled into eleven pools for each cyclization side oligo length and the pooled library was analyzed via native 8% (40:1) polyacrylamide gel electrophoresis (figure 2.7).

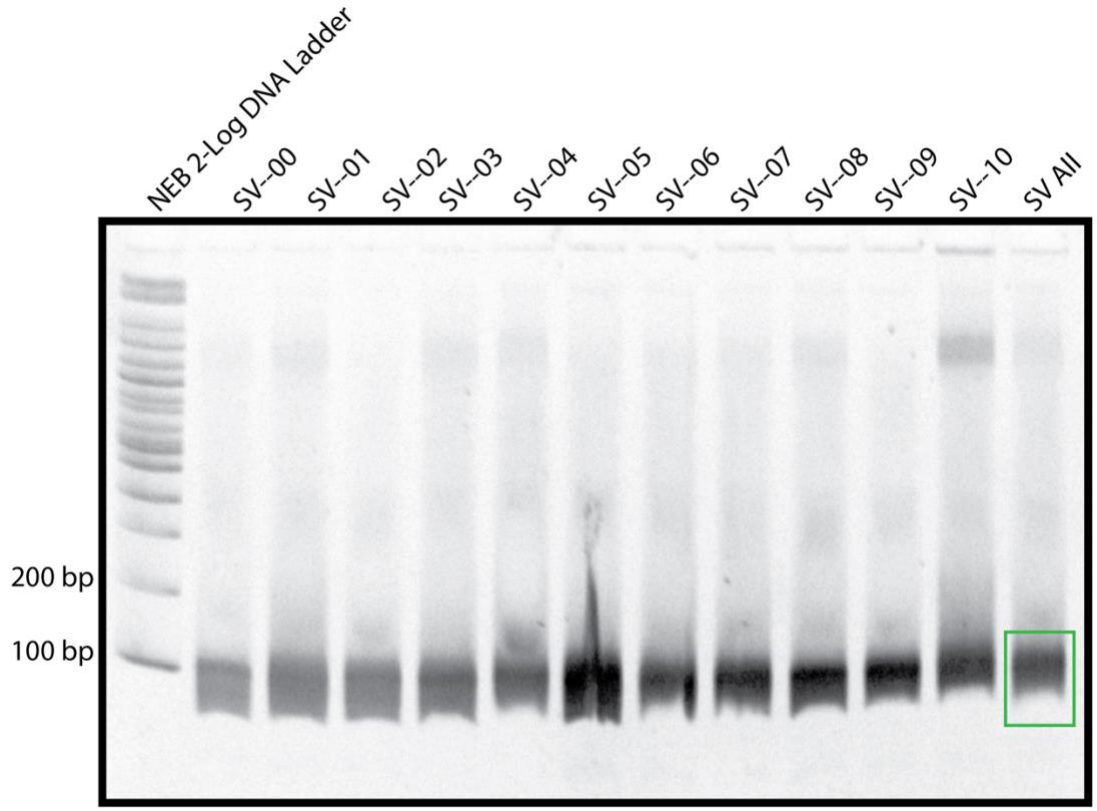

Figure 2.7 Pooled SV mutually primed extension products: Pooled analysis of fully constructed sequence variant portion of the library. The section outlined in green was extracted from a separate, unstained, gel to move forward in full library construction. Each lane is a smear of 31 separate products formed by mutually primed extension with each of the individual cyclization side SV oligos (00-10). Expected sizes range from 90 bp to 130 bp.

Section 2.4.2: Size Class Variant Construction Methodology

The three size class variant molecules were designed as part of a single gBlock. The appropriate overhangs for insertion into a pUC19 vector via Gibson Cloning were determined using the NEBuilder assembly tool and the gBlock was ordered from IDT (figure 2.8). After insertion of the gBlock into a HindIII-EcoRI cleaved pUC19 the cloned plasmid sequence was verified and named pJ4. pJ4 was

grown and a Qiagen Midiprep kit was used to isolate plasmid from transformed DH5-α cells.

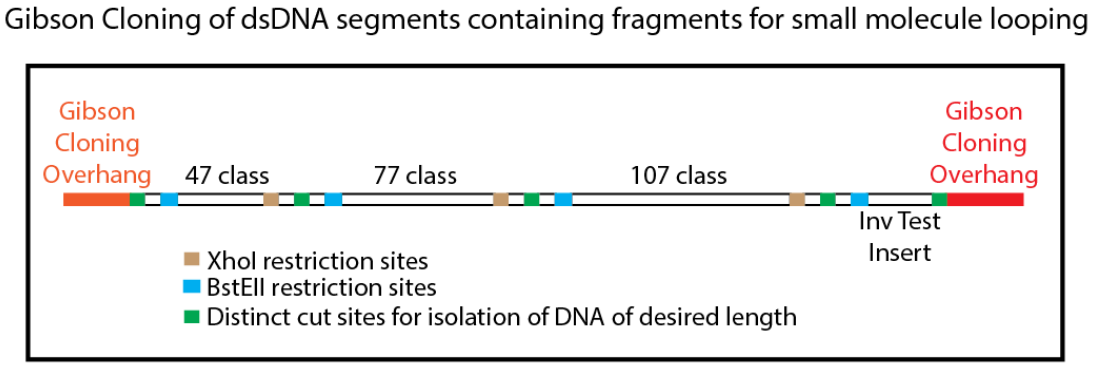

Figure 2.8 Design for gBlock used to generate size class variants.

After plasmid DNA was isolated, a triple digest was performed with BstEII-HF, EcoRV-HF, and XhoI in NEB CutSmart buffer at 37 °C. In order to fully digest 15 µg of plasmid DNA, 200 Units of BstEII-HF, 50 Units of EcoRV-HF, and 200 Units of XhoI were required to complete the digestion in two hours in a 500 µL reaction. The digestion mixture was ethanol precipitated and loaded onto an 8% (40:1) polyacrylamide gel for separation. Bands in this gel were identified via UV Shadowing. No photographs were taken of the initial extraction gel in order to later use the DNA in ring closure assays without having to worry about excess DNA intercalation agents, DNA groove binding agents, or DNA damage from UV light interfering with the results. The shadowed DNA was extracted overnight into 50 mM sodium acetate 1 mM EDTA pH 7. Both the 107 size class and 77 size class were successfully isolated in this manner. However, the 47 size class could not be obtained in reasonable yields through gel extraction. In order to move forward we purchased 5'-phosphorylated oligonucleotides from IDT that would result in the desired BstEII

and XhoI overhangs after annealing to one another. The synthetic 47 size class oligonucleotides were purified in a 10% (40:1) denaturing polyacrylamide gel and annealed. Verification of the three size class variants via 8% (40:1) native polyacrylamide gel electrophoresis can be seen in figure 2.9.

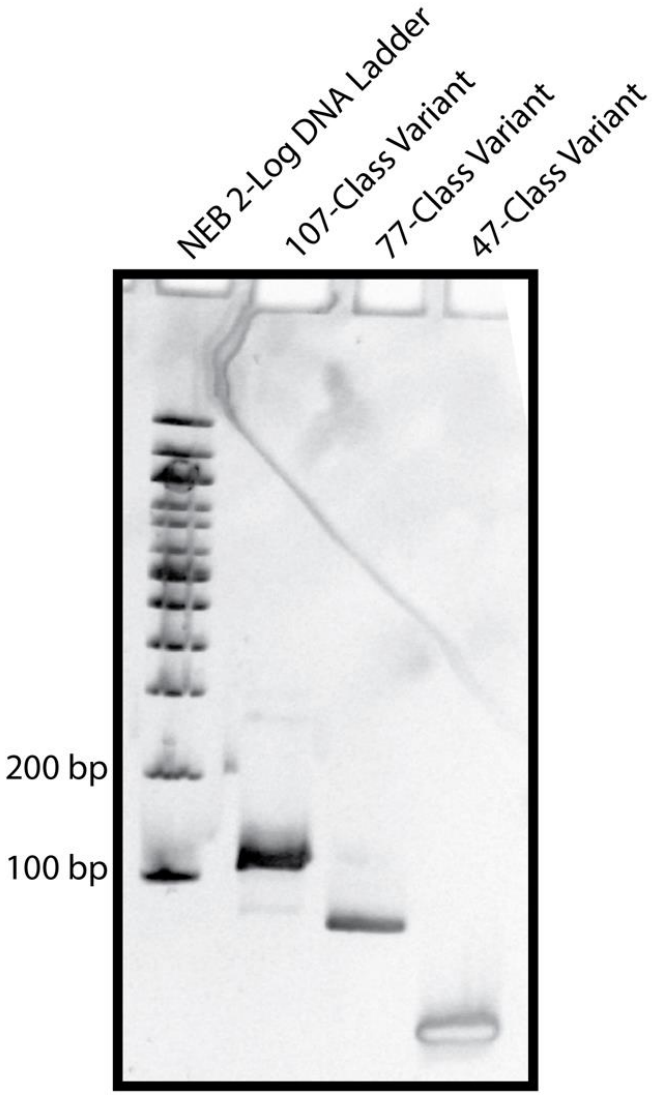

Figure 2.9 Purified SC products: Native polyacrylamide gel electrophoresis showing restricted and gel purified 107 size class and 77 size class variants next to the synthetically generated and annealed 47 size class variant, which was present in much higher concentrations and self-shadowed. Stained with SYBR Gold purchased from Fischer Scientific.

Section 2.5: Library Verification

All enzymatic reagents were purchased from New England Biolabs, [α-^{32}P]-dATP was purchased from Perkin Elmer, PCR Purification and Nucleotide cleanup kits were purchased from Qiagen, and all other reagents were purchased from Fisher Scientific. An MJ Research PTC-200 Gradient thermocycler was used for PCR reactions.

Section 2.5.1: Generation of a Test Library Using Radiolabeled DNA

In order to determine the appropriate concentrations of T4 DNA Ligase and of DNA to use for the bulk experiment, test ring closure experiments were performed. As these experiments are performed using very small amounts of DNA, in order to visualize the results radiolabeled DNA was used. In order to make each of the radiolabeled molecules, nested primers were designed for amplification of the library molecules generated in section 2.2. These sequence variant nesting primers overlapped with the 5' leading sequence designed to allow efficient cleavage with the appropriate restriction enzyme and the restriction enzyme recognition sequence, as this allowed for the same two nesting primers to be used for all sequence variants. The nesting primers used for size variants amplified the entire Gibson insert. Using these primers and [α-^{32}P]-ATP, PCR was performed to generate radioactive versions of portions of the library components.

Sequence variant PCR program:

1. 95 °C 5 minutes
2. 95 °C 1 minute
3. 64.6 °C 30 seconds
4. 72 °C 30 seconds

5. Goto step 2 (35 times)
6. 72 ℃ 5 minutes

pJ4 size class variant PCR program:

1. 95 ℃ 2 minutes
2. 95 ℃ 30 seconds
3. 72 ℃ 3 minutes
4. Goto step 2 (30 times)
5. 72 ℃ 5 minutes

All labeling reactions used a master mix containing 1x NEB Phusion HF buffer, 0.1 mM each dNTP, 1 µL purified template (concentrations varied), 1 µM each primer, 1 Unit NEB Phusion polymerase, and 0.13 µM labelled dATP at 3000 Ci/mmol. The pJ4 size class variant PCR has combined annealing and extension steps.

The products of nested PCR on sequence variants SV1004 and SV1707 were purified via Qiagen PCR Clean-up and digested with either XhoI, BstEII-HF, or both, in 1x NEB CutSmart buffer, and run on an 8% (40:1) native gel. The gel was dried and imaged via image phosphor screen (Storm Imager). Figure 2.10 shows the results of these restrictions and verifies that the length of the leading sequence on both ends was sufficient to allow cleavage with XhoI and BstEII.

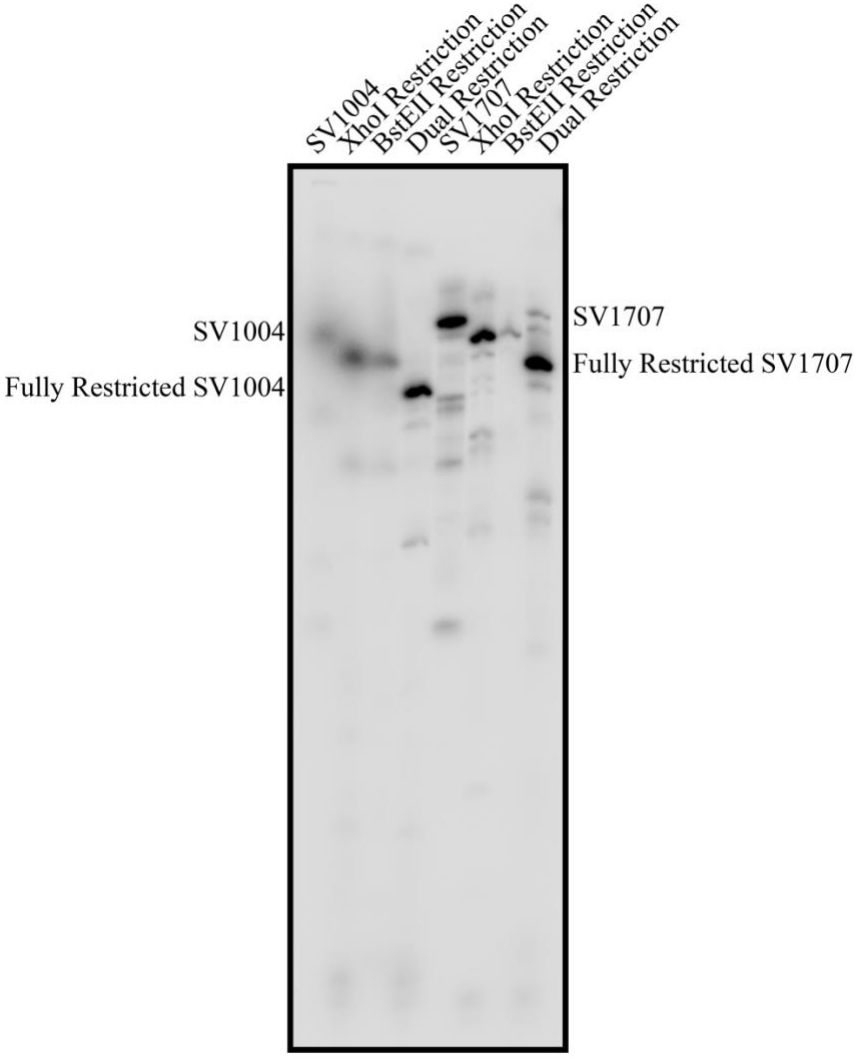

Figure 2.10 SV and SC restriction enzyme test: Sequence variant molecules 1004 and 1707 were restricted using XhoI, BstEII-HF, or dual restricted with XhoI and BstEII-HF. The final dual restricted molecules are 86 bp (SV1004) and 96 bp (SV1707). This test restriction demonstrates that both ends of the molecules are capable of undergoing restriction. This was necessary to determine that the sequence present on the outside of the BstEII and XhoI was long enough as designed. Due to the short overlap of nested primers (11 nt) used to make the radiolabeled samples, off target products were also generated. As these primers were not used to generate the final library and were only used to make select test molecules, we were not worried about these additional products.

The pJ4 PCR reaction was purified via Qiagen PCR Clean-up and all of the reaction product was triple digested with XhoI, BstEII-HF, and EcoRV-HF in 1x

NEB CutSmart buffer. Without the EcoRV-HF two products near 47 base pairs exist from this restriction as the test tnsert in the gBlock design is this size (figure 2.8), EcoRV-HF was added to cut the test insert portion in half allowing purification of the 47 class independently. After ethanol precipitation this material was loaded onto a 10% (40:1) native PAGE for purification (figure 2.11). The appropriate bands for the 107 size class, 77 size class, and 47 size class were cut from the gel matrix and allowed to extract in 50 mM sodium acetate and 1 mM EDTA (pH 7) overnight with agitation.

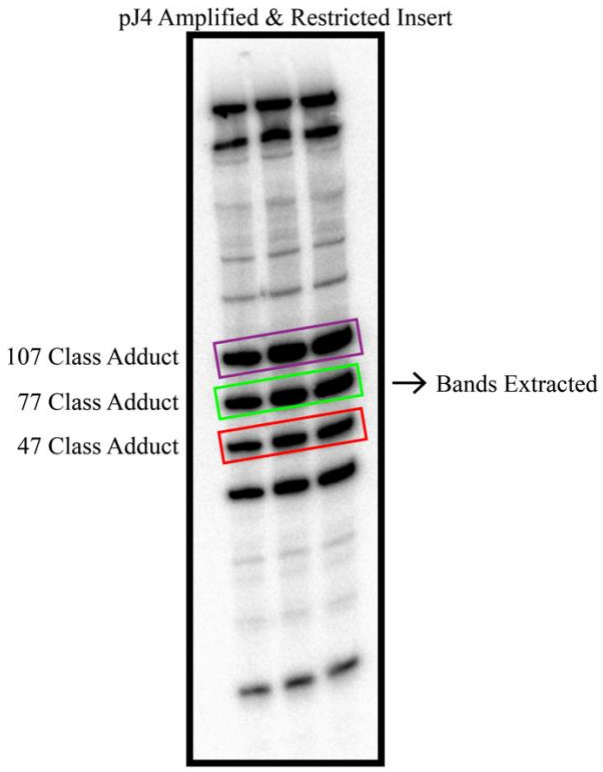

Figure 2.11 Radiolabeled SC purification: A 10% (40:1) native gel of the radiolabeled PCR amplification products of the insert region of the pJ4 plasmid. This gel shows the three desired bands loaded across three lanes to purify all of the material from the PCR reaction which generated it. Multiple off target PCR products are also visible. These off target products are not present in the final library as its preparation does not rely upon PCR to obtain this material, it is cut directly from the

plasmid. The three size class bands were identified and extracted from the gel matrix in 50 mM sodium acetate and 1 mM EDTA (pH 7).

Assembly of the two sides was demonstrated with the purified radiolabeled size class variants and with non-purified sequence variants (figure 2.12). Ligation of the two sides of the library to one another was performed at room temperature overnight with 100 nM each DNA component, 1x NEB CutSmart Buffer, 8 U/µL T4 DNA Ligase, and 1 mM ATP. After assembly the ligase was heat killed (80 ºC, 30 min) and the cyclization junction was re-cleaved with the addition of XhoI to the same solution. These assembled molecules did not show complete ligation at the BstEII site, and subsequent work showed that the BstEII used for the molecules in question was not fully active. The test assembly does show molecules of the appropriate size for both 107 class and 77 class molecules. In this test and subsequent non-radioactive tests, the 47 class adduct did not effectively gel purify for use out of the pJ4 plasmid. It is unknown why the 47 class material was not functional after gel extraction, however in none of our experiments did the 47 class variant yield sufficient material to move forward in library assembly. For this reason synthetic oligos were utilized for the 47 class, as mentioned above. In the 077 class and 107 class assemblies the desired product can be seen but an excess of starting material was visible as well. Further experimentation demonstrated that the BstEII end annealed best when first heated to 55 °C and allowed to cool to room temperature prior to the addition of ligase. In subsequent assemblies this procedure was followed.

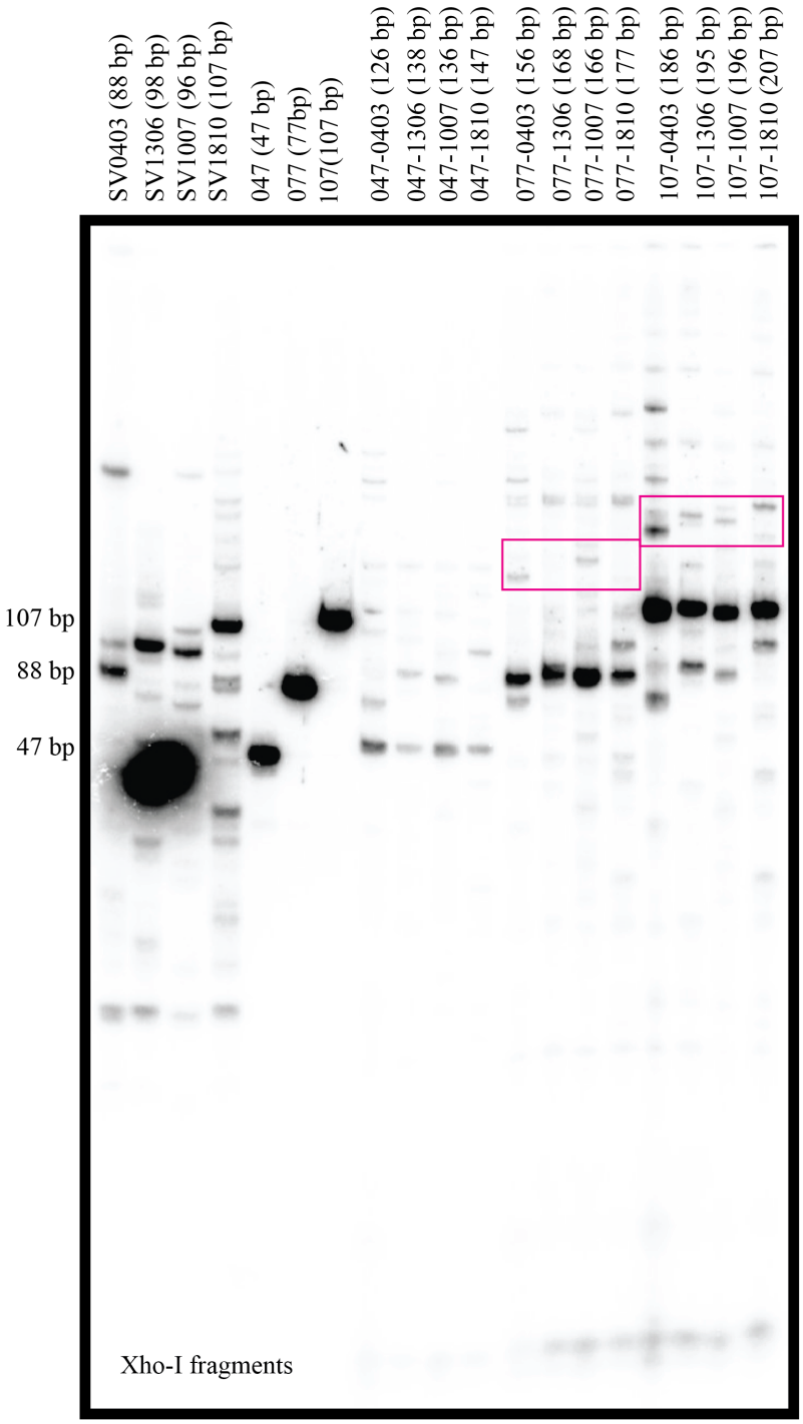

Figure 2.12 Radiolabeled library assembly: Radioactive samples from both sequence variants and size class variants showing assembly to give the desired material in pink boxes. No observable product is seen from the 47 class assemblies. In the 77 class and 107 class assemblies the desired product can be seen but an excess of starting material is visible as well.

Section 2.5.2: Radioactive Test Cyclization

Both DNA concentrations and T4 DNA Ligase concentrations desired in the ring closure assay were decided upon based on previous work in the lab[34,38]. In those ring closure experiments the size and presence of a DNA binding enzyme could have had an impact on the concentrations used. As such, we decided to verify that cyclization occurred to an appropriate extent under the chosen DNA and ligase conditions.

Sample DNA library molecules used for these test ligations utilized the sequence variant molecules prepared with larger size class molecules than the final library used. This allowed for a single set of ligation tests to span our current size class and future size classes of interest (discussed in chapter 5). Library molecules were prepared in the same manner as in section 2.5.1. The test cyclization was performed with 1 nM DNA in 1x NEB CutSmart Buffer supplemented with 1 mM ATP. The final concentration of T4 DNA Ligase in the reaction was 1000 U/mL. After incubation at room temperature for 30 minutes the reaction was heat killed, ethanol precipitated, and resuspended in 1x TE. Half of the resuspended DNA for each molecule was digested with BAL-31 Nuclease for one hour. Both pre and post BAL-31 digested samples were run on an 8% (40:1) polyacrylamide gel with 7.5 µg/mL chloroquine (figure 2.13).

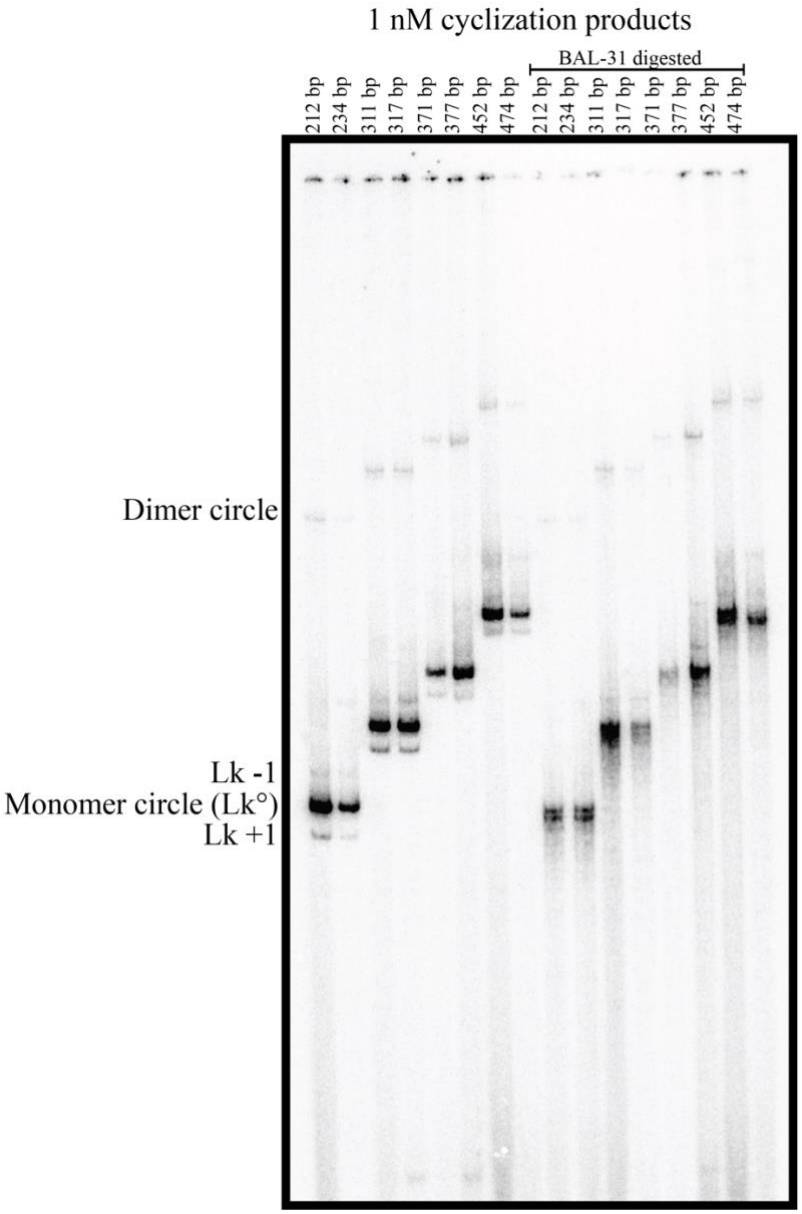

Figure 2.13 Radiolabeled test cyclization gel: Products of a ring closure assay performed on multiple different lengths of DNA under the same ligase conditions and DNA concentration. Monomer circles with $\Delta Lk = \pm 1$ topoisomers can be seen as well as the formation of double circles. Different topoisomers can be seen on this gel due to the presence of chloroquine (7.5 μg/mL) in the gel.

From figure 2.13 the ligase concentration decided upon for subsequent ring closure assays was 1000 U/mL. While this assay was performed at 1 nM, given the

decrease in J factors at shorter lengths we decided to test 0.5 nM and 0.05 nM DNA concentrations in ring closure experiments intended for sequencing analysis.

Section 2.6: Assembly of Complete Library

In total, 341 separate sequence variant library molecules were generated and three size class variant molecules were generated. These components were each combined into the final full sequence and length library in three separate reactions, one for each size class.

For each mixture the individual size class molecule was combined with all 341 sequence variants in a single tube with each component at 500 nM with NEB T4 DNA Ligase (400 U/µL), CutSmart Buffer (1x), and ATP (100 mM). Reactions were allowed to proceed overnight at room temperature. The following morning T4 DNA Ligase was heat killed at 80 °C for 30 minutes before 160 Units of XhoI was added to each mixture and the mixtures were digested for one and a half hours at 37 °C. At this point the individual amount of each molecule was low enough that in order to visualize the complete library via gel electrophoresis would have consumed too much of the sample library. Based on gel electrophoresis of the material that went into the library formation reactions and the test work done on individual molecules we decided to move forward with the length library cyclization experiment. The quality of the library was verified by DNA sequencing as described in chapter 3.

Section 2.7: Ring Closure Experiment

From the radioactive test experimentation performed (section 2.5), we decided upon a final T4 DNA Ligase concentration of 1000 U/mL and DNA concentrations of

0.5 nM and 0.05 nM. A schematic of the ring closure experiment is provided in figures 2.14 and 2.15. In order to end with equivalent amounts of DNA from the 0.5 nM and 0.05 nM reactions, they were performed at different volumes (150 µL and 1,500 µL respectively). All ligation reactions were carried out for a total of 30 minutes at room temperature in 1x NEB CutSmart Buffer with 1 mM ATP.

The ligation reactions were heat killed prior to ethanol precipitation. Precipitated DNA was resuspended in TE and each sample was split in half. Half of each sample was digested with BAL-31 nuclease with the addition of an equivalent volume of 2x BAL-31 nuclease buffer and 1 unit (NEB) of BAL-31 per sample. The BAL-31 digestions were heat killed prior to an additional ethanol precipitation. Resuspended BAL-31 digested samples and original non-BAL-31 digested samples were digested with SspI-HF in 1x NEB CutSmart Buffer. Following SspI digestion all samples were again ethanol precipitated and resuspended in TE to be submitted for sequencing.

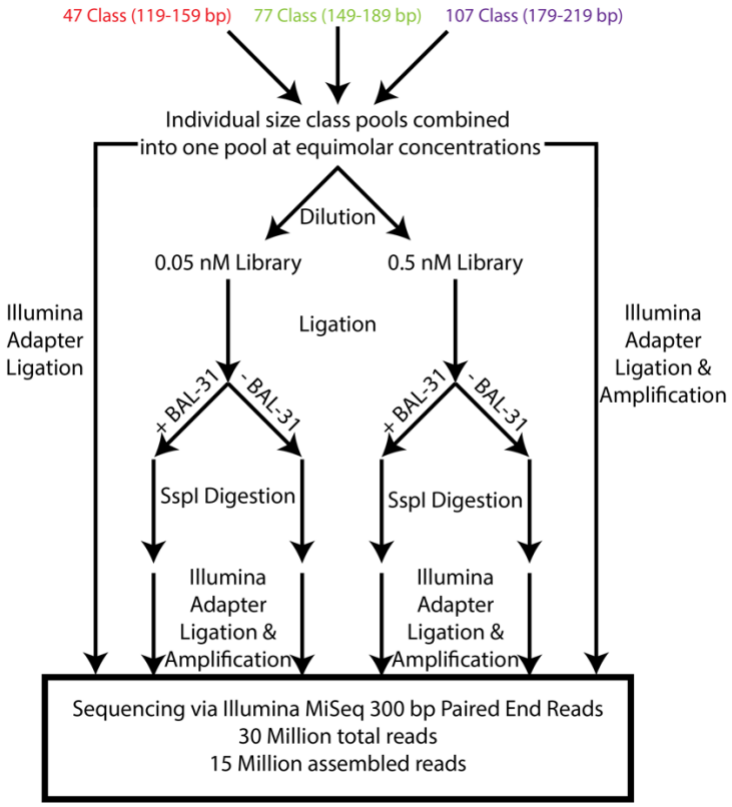

Figure 2.14 A schematic showing the ring closure experiment workflow.

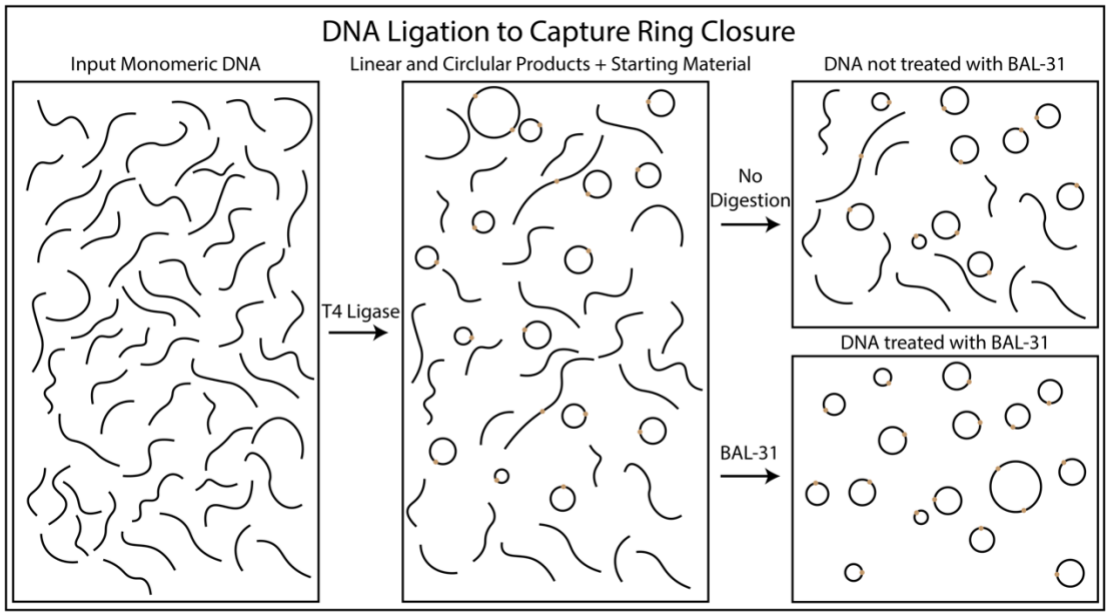

Figure 2.15 Schematic of DNA ring closure as captured by T4 DNA Ligase.

Chapter 3: Length Library Ring Closure Experiment Sequencing and Sequencing Analysis Programming

Section 3.1: Overview

The field of DNA sequencing has changed drastically in the last fifty years. From the advent of Sanger sequencing in 1977 where a single sample would require four separate reactions to sequence via gel electrophoresis to the advent of the first capillary electrophoresis sequencer by Applied Biosystems in 1987, the cost of sequencing drastically decreased as the rate of sequencing increased[39,40]. At the time of the NIH and Celera human genome projects in the late 1990s the predominant DNA sequencing method was capillary electrophoresis using Applied Biosystems 3730xl instrumentation. While this instrumentation made sequencing the human genome easier than Sanger could have imagined in 1977, it pales in comparison to the advances seen in high throughput sequencing over the last twenty years. In modern sequencing multiple short reads that are reassembled into a full sequence using a known reference sequence are favored over individual long reads. This multiplexing has made it possible for benchtop instruments like the Illumina HiSeq X Ten to have a throughput capable of sequencing 45 complete human genomes in a single day at a cost of less than $1000 per genome sequenced[41,42]. With new multiplexing methods that are capable of multiple reads in a single run, genomic studies have drastically changed and diverse applications like the one presented here have become possible[43].

Section 3.2: Library Sequencing

Six samples were submitted for sequencing. Four samples were the two ligation reactions performed at different concentrations ±BAL-31 digestion, the other two samples were the linear assembled library as it went into the ring closure experiment (as seen in figures 2.14 and 2.15). In deciding upon the volume of reaction necessary after choosing ligation concentrations, we settled on 150 μL and 1.5 mL reactions for the 0.5 nM and 0.05 nM ligations respectively. In total this is roughly 75 femtomoles of DNA in each ligation reaction. After being split for BAL-31 digestion a maximum of 37.5 femtomoles (4 nanograms) should have been present per sample. In the BAL-31 samples some portion of the DNA was digested, resulting in even less DNA than the maximum 37.5 femtomoles possible. In researching the number of reads obtained in a typical MiSeq run, we felt this would be enough as 37.5 femtomoles is roughly 30 billion molecules per sample (29 million molecules per individual sequence). However, as we found out, the Illumina workup generally requires much more DNA than this at roughly 50 ng worth of DNA for a low input sample.

Sequencing was performed on an Illumina MiSeq at the University of Maryland School of Medicine Institute for Genome Sciences (IGS). The IGS standard operating procedure requests 50 ng worth of DNA in a low input sample. Using a Promega QuantiFluor system it was determined that our samples contained anywhere from 0.64 to 5 ng. With this in mind, over discussions with Luke Tallon and Lisa Sadzewicz at IGS we decided that library amplification was in our best interests. In order to verify whether sample amplification bias issues were present we also

submitted a high concentration sample of the library that did not undergo amplification.

Library preparation (adapter ligation and amplification via Kapa Hyper Prep kit) was performed by IGS, with different adapter sets being applied to all submitted samples for multiplexing. All six samples were sequenced on the same flow cell in a single run of the Illumina MiSeq with 300 base pair paired end reads.

In total, 31,553,784 reads were sequenced containing 5,291,218,417 base pairs. Sequence analysis was performed on the computational cluster maintained by the University of Maryland Institute for Advanced Computer Studies at the University of Maryland College Park. Prior to writing our own software to analyze the sequencing results, FastQC was used to assess overall read quality[44]. There were no apparent issues with any of the reads present in FastQC analysis. The average sequence quality began to drop off after 100 base pairs, but this was to be expected given our library length as reads of short length appear to have added semi-random nucleotides after the desired sequence. After assessing quality the paired end reads were assembled using the Fast Length Adjustment of Short reads (FLASh) software suite[45]. Given the short length of some of the sequences in the library, 300 base pair reads extended beyond the start of the sequence on the other strand, and in order to overcome this FLASh had to be set to allow for sequences to extend beyond the start of the other molecule, but all other settings remained at the default. After assembly there was a total of 15,077,132 double stranded reads in the library.

Section 3.3: Sequencing Analysis Programming

As sequencing has become more commonplace the field of bioinformatics and sequence analysis has also developed rapidly to try to fill the needs of molecular biologists and biochemists. While many programs have been written focused on genome assembly, no existing program existed that would satisfy our specific needs. Through discussions with Drs. Najib El-Sayed and Ashton Trey Belew, we decided the best route to analyze our data would be to write a software package from scratch to perform all of the necessary analysis. In order to determine cyclization efficiency, three classes of molecules must be identified: unreacted library, unimolecular circles, and bimolecular/multimerization products. These classes must be distinguished based not only on which sequence elements are present within a given read, but the ordering and directionality of each of these elements in the read. In order to do this, a program was written in Perl and appropriate indices were derived to allow for separation of each sequence. In order to try to see as many possible molecules as possible the sequence analysis program allows for substitutions, insertions, and deletions as variable inputs when running the program. Given how the index sequences were barcoded (as discussed in chapter 2), a single base substitution is often enough to identify multiple index hits for each actual index. To overcome this, the index sequences were lengthened to encompass the entire insert length and nucleotides on either end of the insert region. By doing so the program could be run allowing for a single substitution and proper index identity could still be achieved. With the ability to look for index matches with substitutions, insertions, and/or deletions, we found that very few additional matches could be found with a single substitution within any

index sequence. With the overall error rate for base calling during sequencing being low, we moved forward only with direct index matches for the rest of our analysis. The Perl program used for the following analysis can be found in appendix 5.1, and the indices used can be found in appendix 5.2. An electronic current working version of all applicable sequencing analysis files can be found at https://github.com/jhustedt/kahn-ngs

Section 3.3.1: Library Indices

In order to identify all of the components within an individual read there are four index areas in each library molecule (figure 3.1). While the two sequence variant variable regions have their size encoded in a fraction of the total length of the read, in order to allow for potential single base substitution during sequencing the indices for these regions were designed to span the entire length of the variable insert including constant nucleotides on either end of the variable sections. Size class indices are present on either side of the SspI restriction site such that after SspI cleavage individual sides from the original molecule can be identified. While the variable sequence region indices are not a part of the class they are inside, they are colored along with their parent molecule throughout this section for ease of visual identification of intra-class and inter-class reactions.

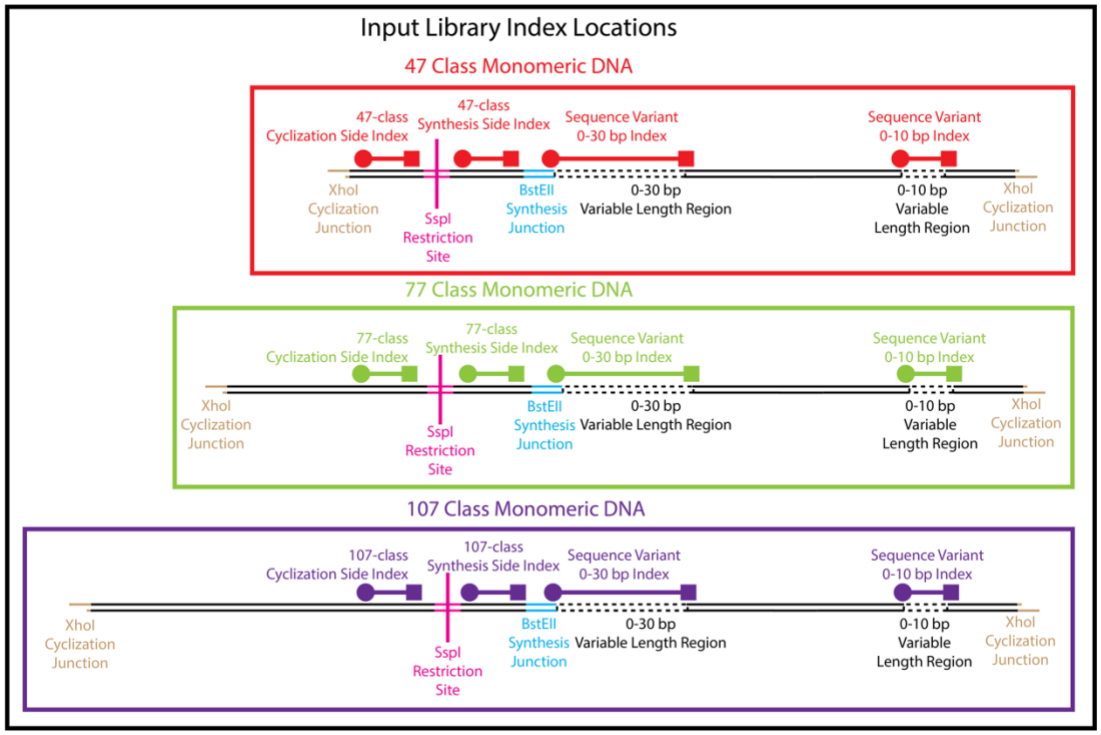

Figure 3.1 Input library index locations highlighted for each size class: Size class to sequence variant class ligation occurred at the light blue region. Each molecule of the library contains two indices from the sequence variant class that will differ according to molecule as well as two indices from the size class side of the molecule.

Section 3.3.2: Classification of Identified Molecules

Based on the input library there are four classes of reactions that needed to be identified and separated. These include unreacted input library, monomer circles, dimer circles, linear multimers, and circular multimers. Across each of these classes there are multiple sub-classes that must further be separated. In dimer circles for example there are both inter-class (e.g. 47+77) and intra-class (e.g. 47+47) dimer circles that both can occur either as cyclization from an AB-AB linear bimolecular molecule or as cyclization from an AB-BA or BA-AB linear bimolecular molecule, where A and B are the different XhoI ends from the SV and SC sides respectively.

The number of possible combinations is large and separating everything out is not immediately straightforward. Given the nature of the sequencing workflow, with only three size class variants present (47, 77, and 107), there are abundant intra-class bimolecular and linear products that unfortunately cannot be separated from unimolecular products. A general schematic showing separation of all possible identified molecules can be seen in figure 3.2, with connecting lines shown between indistinguishable sequenced DNAs in rounded boxes. This schematic is presented as a general overview – like a photo of the metabolism chart. Individual schematics of all elements within figure 3.2 are broken out below: a representative unimolecular circle (figure 3.3), a representative intra-class bimolecular circle (figure 3.4), representative inter-class bimolecular circles (figure 3.5), and inter-class bimolecular and multimer linear products (figure 3.6).

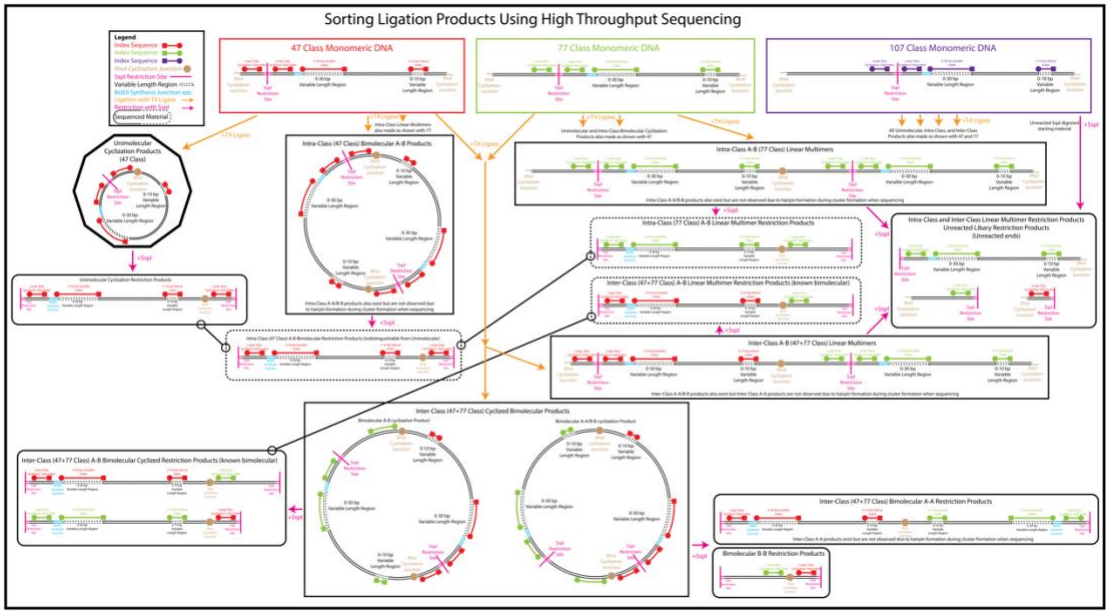

Figure 3.2 Overview schematic of possible ligation products: An overview schematic of possible unimolecular, intra-class, and inter-class ligation reactions that are expected to happen in our ligation experiment. The three size classes are shown at the top as monomeric DNA with red (47 class), green

(77 class), or purple (107 class) indices. Orange arrows show ligation events within these classes leading to the intermediate products. The SspI restriction site is highlighted in pink and post restriction linear products are shown after the pink arrows. This chart is disassembled in the sub-views below.

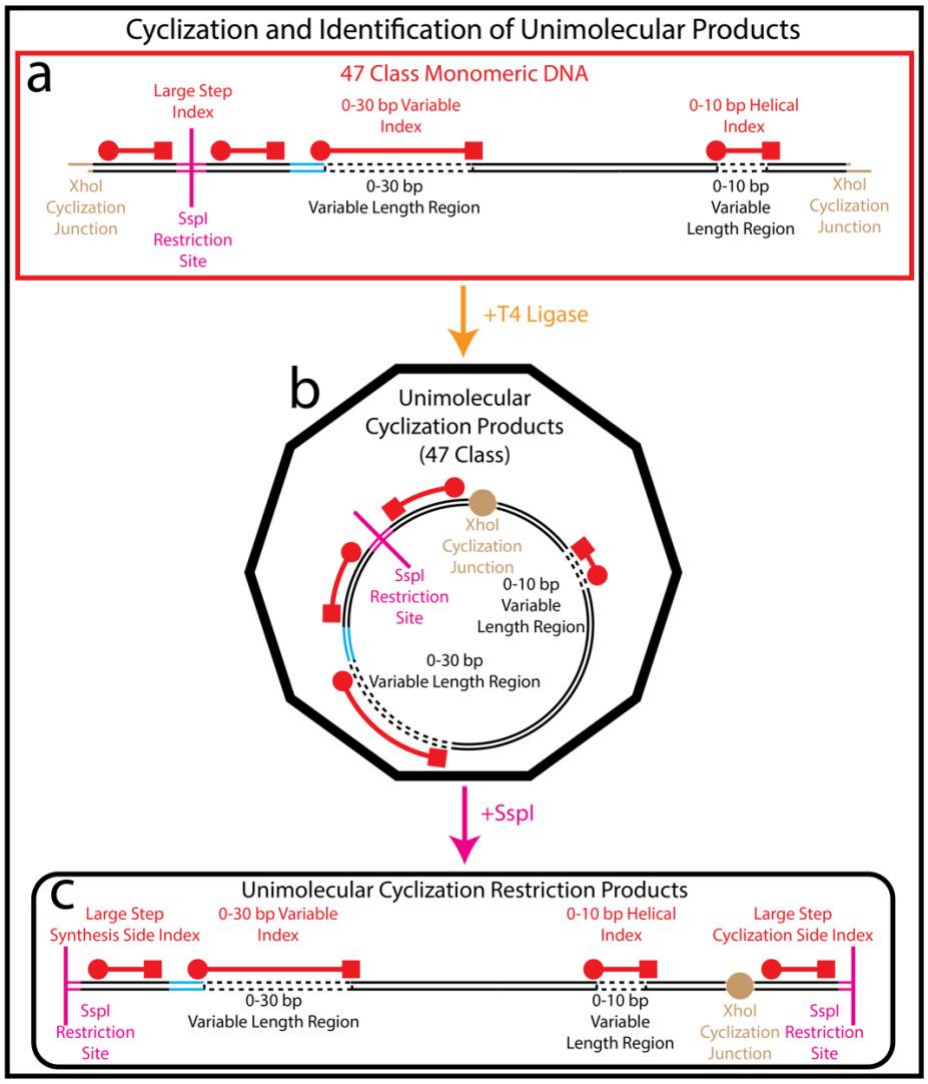

Figure 3.3 Unimolecular cyclization: A schematic showing the unimolecular cyclization of a sample 47 class molecule: The input material (a) once ligated becomes a unimolecular circle (b). After restriction with SspI the resultant molecule (c) will have the two large step indices on flanking ends of the molecule with the two sequence variant indices in between. The four indices seen in (c) in order are the 47 class synthesis side index, the SV 0-30 bp insert index, the SV 0-10 bp insert index, and the 47 class cyclization side index. The large step indices differ for the 77 class and 107 class molecule allowing differentiation from 47 class molecules.

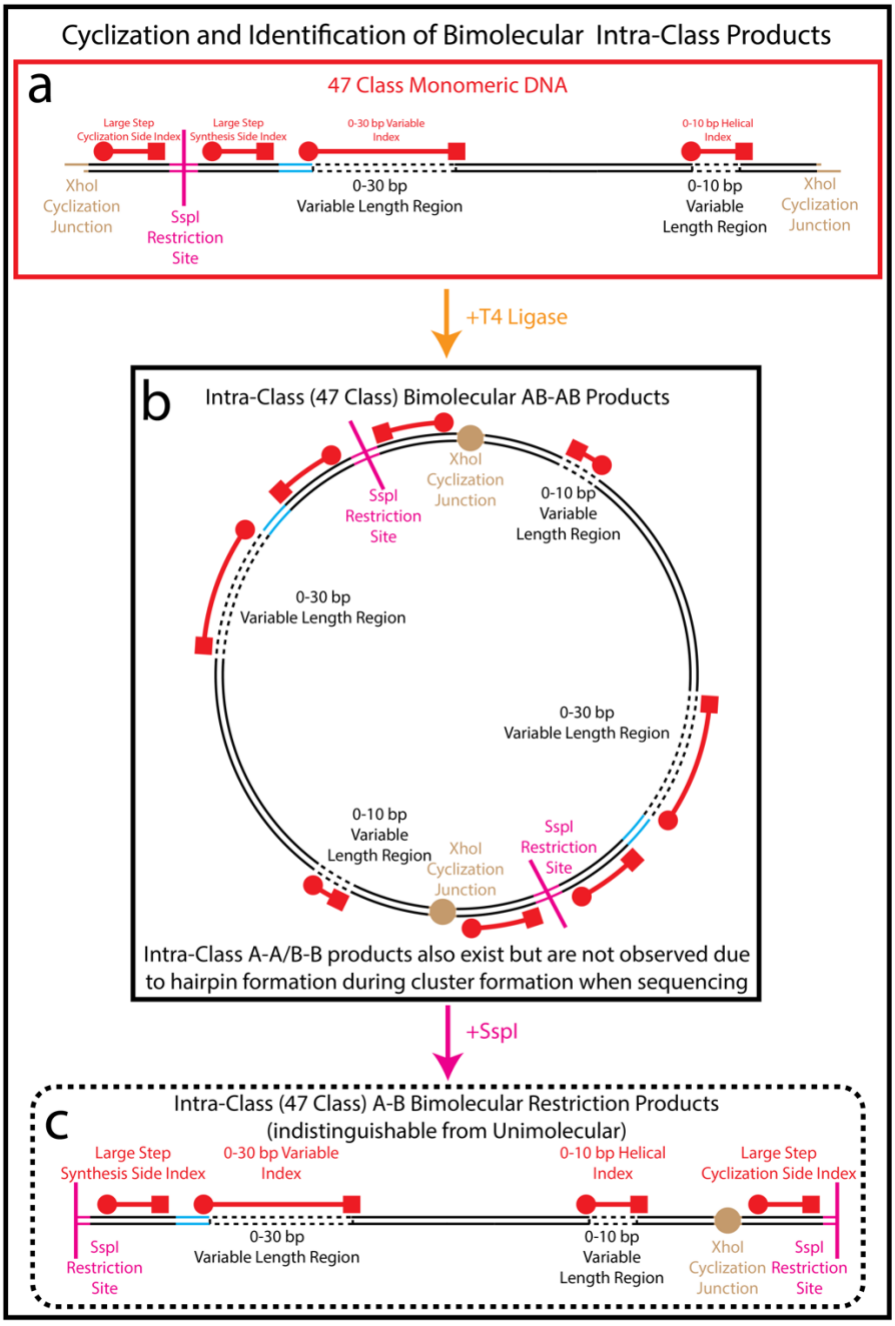

Figure 3.4 Bimolecular intra-class cyclization: A schematic showing a bimolecular intra-class ligation followed by cyclization of two 47 class molecules in an AB-AB orientation. The input material (a) once ligated to a second linear molecule cyclizes to become a bimolecular circle (b). After restriction with SspI, two identical molecules will result (c). Each will have the two large step indices on flanking ends of the molecule with the two sequence variant indices in between. In an AB-AB ligation these molecules are indistinguishable from a unimolecular cyclization event.

If the two large step indices (see top left figure 3.3) are from the same size class molecule we can conclude that the ligation event that occurred to give this molecule came from a ligation to another 47 class molecule, however, as the size class index on the cyclization side does not have any coding other than size class alone, we are unable to determine whether the sequenced molecule came from a unimolecular cyclization (as in figure 3.3) or from a bimolecular intra-class AB-AB cyclization (as in figure 3.4).

Bimolecular intra-class BA-AB molecules and bimolecular inter-class BA-AB molecules are not seen during sequencing. Regardless of variants present, we believe there is enough overlapping sequence in all A-A ligation molecules to prevent proper cluster formation due to hairpin formation as this overlapping sequence is an extended inverted repeat[46]. B-B ligation events from intra-class ligations likewise can form hairpins, however B-B ligation events from inter-class ligations should form distinguishable products as seen in figure 3.5. This figure also shows identification of bimolecular inter-class AB-AB products. These products yield an A-B linear fragment after restriction with SspI that is distinguishable as an inter-class product (as seen in figure 3.5c).

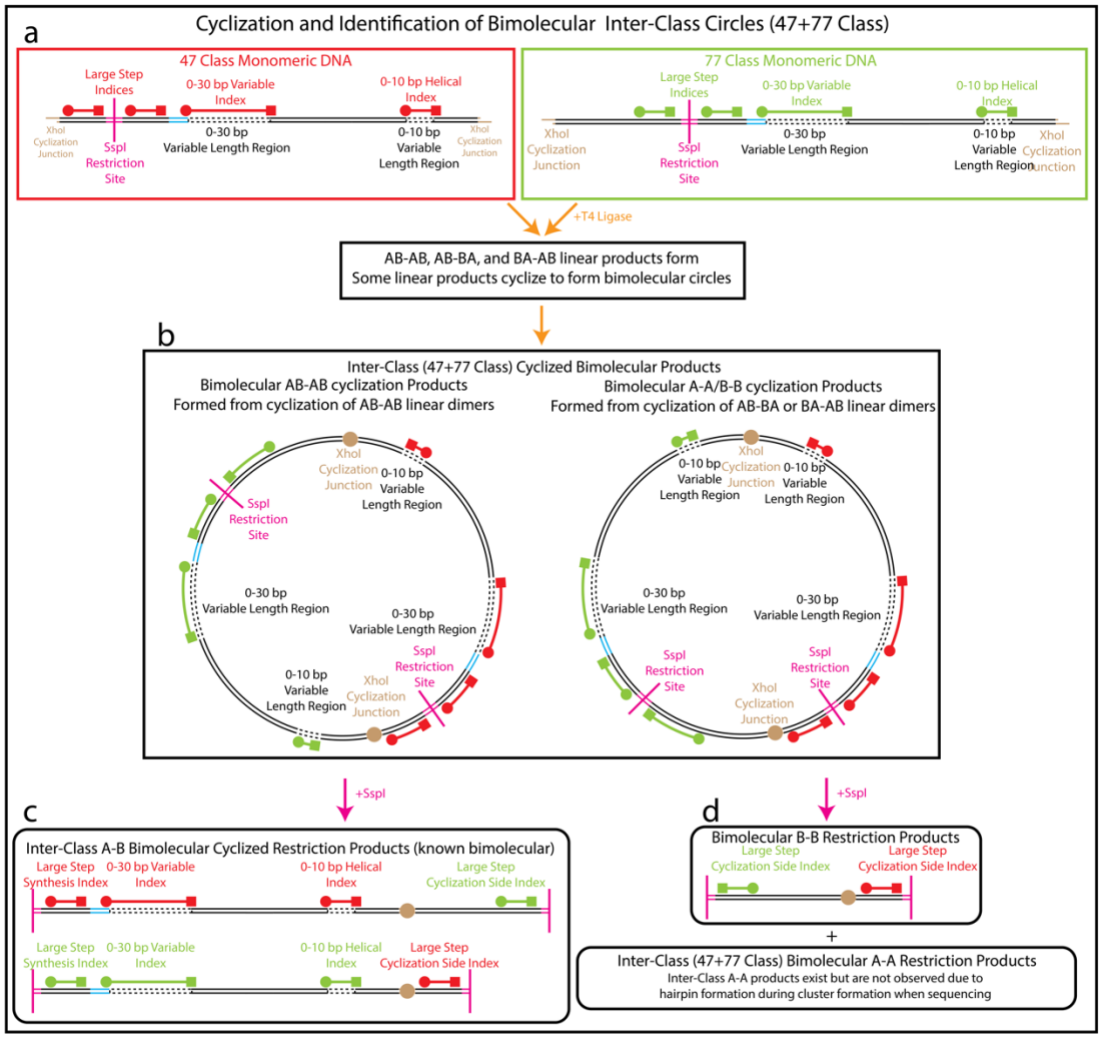

Figure 3.5 Bimolecular inter-class cyclization: A schematic showing a bimolecular inter-class ligation and cyclization of a 47 class molecule with a 77 class molecule. The input material (a) once ligated to a second linear molecule cyclizes to become one of two different bimolecular circles (b), AB-AB and A-A/B-B cyclization products, named after their junctions. After restriction with SspI two molecules will result from each bimolecular circle. The bimolecular AB-AB circle will yield two distinct A-B linear molecules of differing size (c). These two molecules are known to be bimolecular molecules as the two large step indices on flanking ends of the molecule differ from one another. The bimolecular A-A/B-B circle will yield an A-A molecule that will not sequence and a B-B molecule which can be sequenced, but does not provide information on the lengths of the SV 0-30 and SV 0-10 regions of participating molecules.

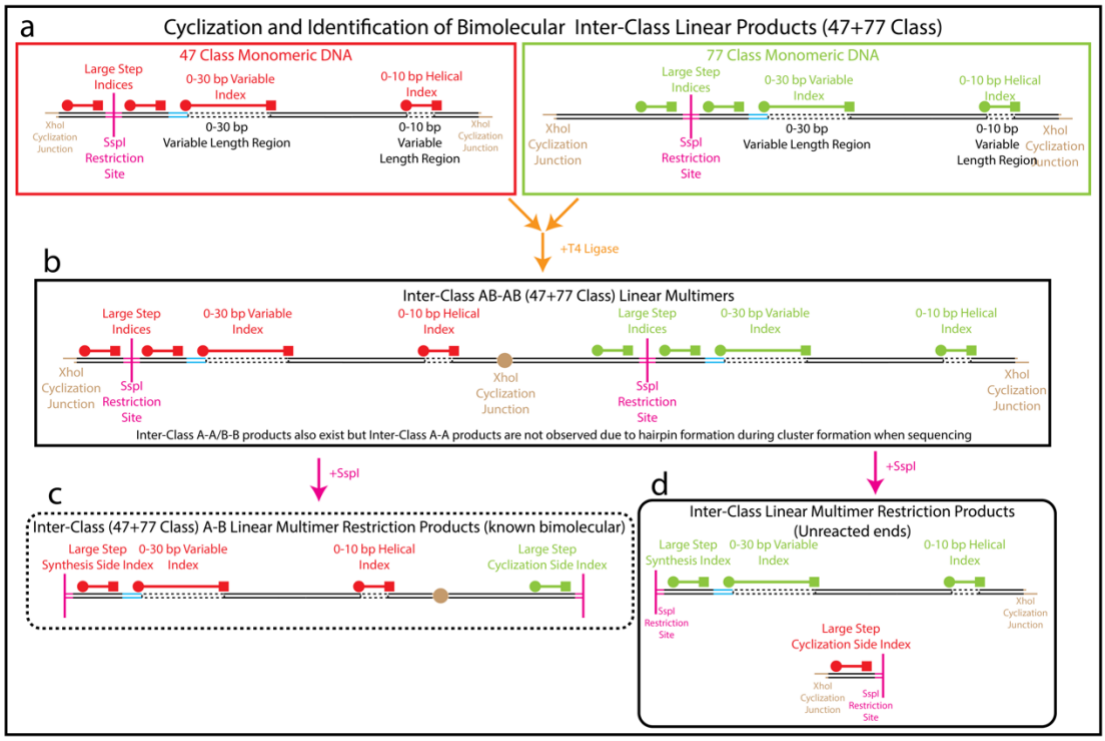

Figure 3.6 Bimolecular inter-class linear ligation: A schematic showing a bimolecular inter-class ligation of a 47 class molecule with a 77 class molecule ending with a linear product. The input material (a) once ligated to a second linear molecule becomes a linear dimer (or multimer) (b). In the event that this multimer does not cyclize and remains a linear molecule, after restriction with SspI three molecules will result: a molecule that is known as bimolecular from the middle of the dimer or multimer (c) and two unreacted fragments from the ends of the dimer or multimer (d). The molecule in (c) is the same as the A-B product from cyclization of the AB-AB dimer circle in figure 3.5. These two molecules are identical, but the products from figure 3.6 are susceptible to digestion with BAL-31 and will not appear in samples that have been treated with BAL-31.

The contribution of inter-class bimolecular AB-AB ligations can only be distinguished between dimer circles and linear dimers/multimers when looking at them as a fraction of the entire library in samples that have been digested with BAL-

31 and those that have not been digested with BAL-31. Dimer circles will survive BAL-31 digestion, linear dimers and linear multimers will not.

Section 3.4: Ring Closure Sequencing Library Results

Section 3.4.1: Summary of Results

The logic shown in the above schematics was used to separate and identify as much of the library as possible. A total of 6,360,976 of the 15,077,132 reads, roughly 42%, were identified as coming from our own library. Another 6,691,252 reads were identifiable contamination. In all, 13,052,228 reads across all six samples, roughly 87% of our total reads, were identified.

Identified Indices and Contamination within the Sequencing Results				
Sample	Assembled Reads	Library Reads	Identified Contamination/ Spike-in Control	Total Identified Percentage
0.5 nM Ligation	1,978,879	1,866,481	93,129	99.03%
0.5 nM Ligation BAL-31 Treated	4,487,293	487,489	2,828,312	73.89%
0.05 nM Ligation	2,040,329	875,430	1,130,571	98.32%
0.05 nM Ligation BAL-31 Treated	3,952,974	698,730	2,466,299	80.07%
Amplified Starting Library	1,852,028	1,775,870	66,894	99.50%
No Amplification Starting Library	765,629	656,976	106,047	99.66%
Total	15,077,132	6,360,976	6,691,252	86.57%

Table 3.1 Identified indices and contamination: A comparison of read identity within each assembled sample. Samples with a higher input quantity of DNA had a higher number of total reads coming from our own desired material. Colors here representative of the colors assigned in figures analyzing the library throughout the rest of the chapter, 0.5 nM ligation reaction products in orange, 0.05 nM ligation reaction products in blue, and library in green. The total identified percentage consists of all library reads as well as reads coming from contamination and spike-in control.

Section 3.4.2: Analysis of Sequencing Contamination and Amplification Bias

While the majority of all assembled reads could be identified, a larger amount was attributable to contamination than was initially expected by us. In fact, slightly more reads came from contamination than came from actual library molecules. Given the low input quantity of our samples the library prep procedure amplified background contamination and the spike-in control DNA added to each sample by the IGS sequencing center more than in a normal sequencing run. Much of this contamination came from spiked in control DNA in the form of ϕX174. The other contributing factors of contamination included sections of the ampicillin resistance gene, the ampicillin resistance promotor from pBR322, the F1 origin of replication, the pUC origin of replication, NEBNext adapter sequences, and Illumina TruSeq adapter sequences. Contamination analysis occurred after reads with our own indices had been identified. Reads containing no index hits were searched in NCBI's VecScreen tool. If a match was found the full sequence of the match was broken into smaller chunks of 20-25 nucleotides and added to the index file for subsequent contamination screening. After each new source of contamination was added to the index file a subsequent search of reads with no hits was performed on VecScreen until all remaining zero hit reads yielded no results on VecScreen. A breakdown of all sequenced spike in control and contamination as a proportion of each sample can be found in figure 3.7 and a breakdown of raw counts for each sample by identity can be found in figure 3.8.

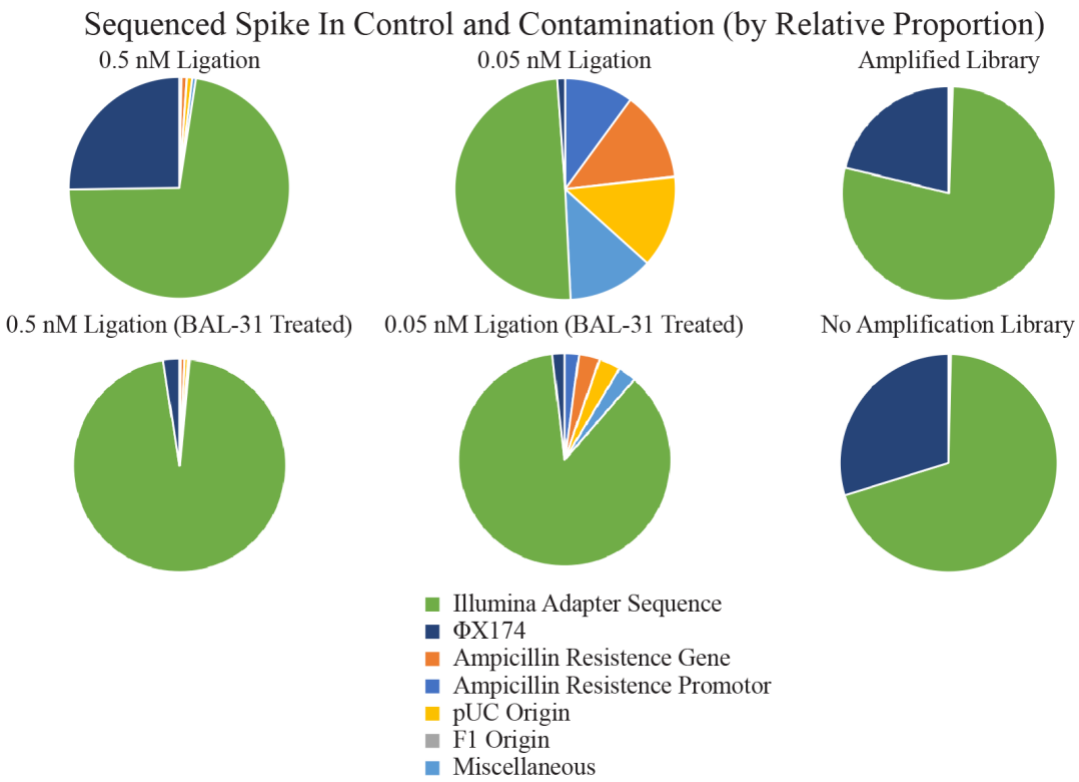

Figure 3.7 Sequenced spike-in control and contamination by relative proportion: The majority of all contamination in any given sample was adapter-adapter dimerization (green). The second most frequent contaminant was ϕX174 which is used by IGS as a spike in control for each sample (dark blue). Additional contaminants were found in each sample but in low populations relative to the adapter and spike in numbers for a given sample.

For the samples submitted with the highest quantity of DNA the total number of reads identified was 98% or higher. The two samples with the least DNA were the two BAL-31 treated samples. With these two samples 74% and 80% of the total reads were identified. It is believed that the remainder of the non-identified reads in these samples are random PCR artifacts from the library preparation step at IGS. While as a percentage of the total reads our own sample was less than half, we did obtain enough total reads for further analysis.

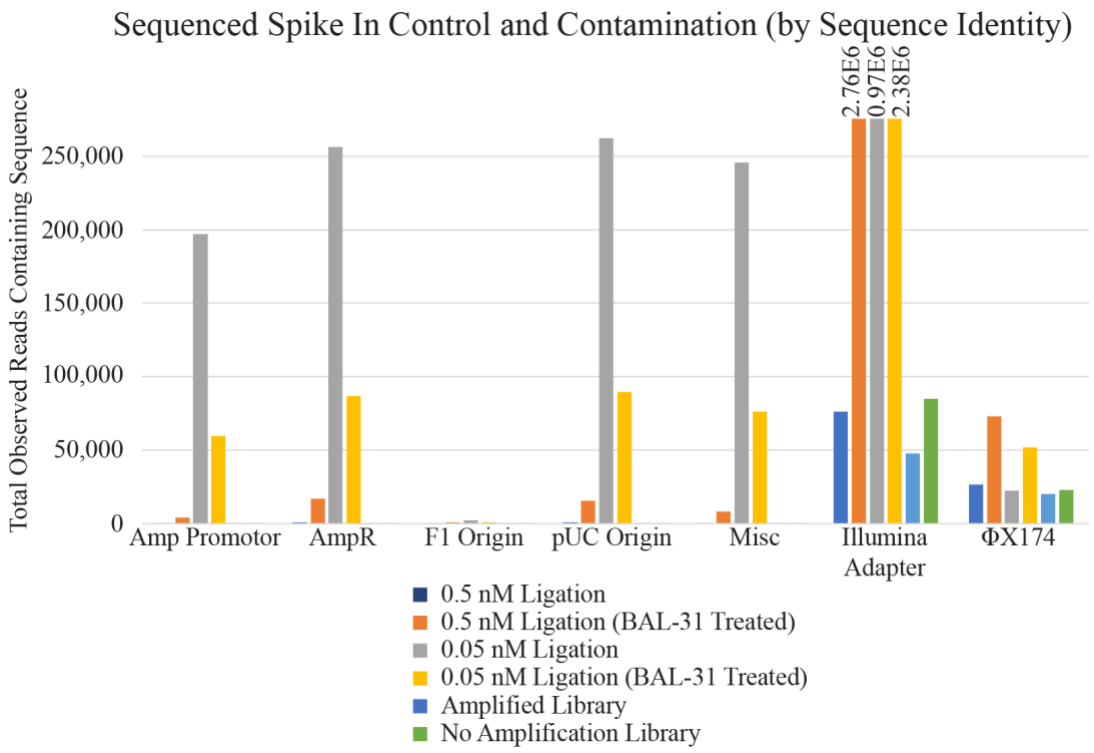

Figure 3.8 Sequenced spike-in control and contamination by sequence identity: When the source of contamination is broken down by sequence identity it can be seen that the high quantity DNA samples (0.5 nM Ligation and the two Library samples) have very little contamination, but in the low DNA samples, all contamination is amplified. In these bar graphs the Illumina adapter peaks can be seen exceeding the range of the bar graph by an order of magnitude.

In addition to contamination issues, the overall impact of the pre-sequencing amplification was checked to verify that the counts obtained from amplified samples could be trusted. The impact of amplification on the library can be seen in figure 3.11. In both the amplified and non-amplified libraries a discrepancy between each size class can be seen. While the concentrations of each class was believed to be identical from ultraviolet absorbance measurements taken of the purified samples at 260 nm it is clear that the concentrations were not exact and some variation was introduced during library prep and workup. The relative abundance of each individual molecule

however does not show any amplification bias from the pre-sequencing amplification. This comparison does not rule out potential bias that could exist in cluster formation.

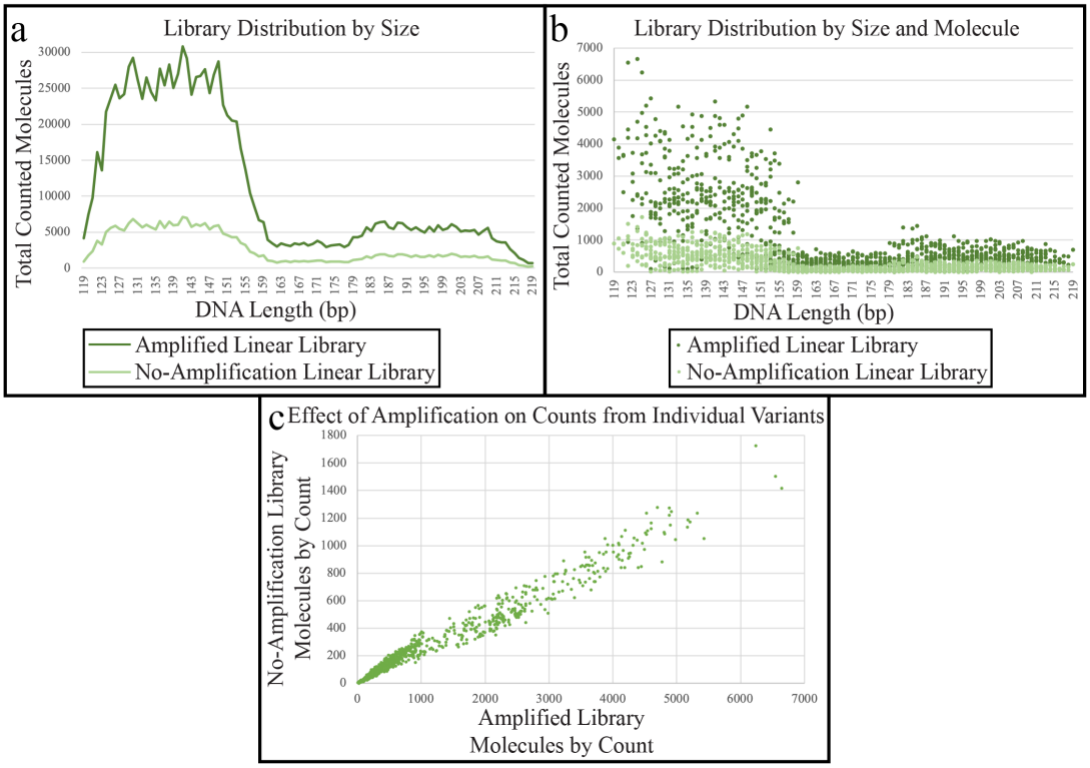

Figure 3.9 Comparison of PCR amplified and non-amplified library sequencing results: The three size classes have uneven concentrations in both the non-amplified and amplified library (a). When each individual DNA sequence variant is graphed no trends appear to make us believe that sequence specific bias during amplification would be an issue (b). When raw counts of non-amplified molecules are graphed against the total counts of the same molecule after amplification a relatively linear increase is apparent without any outliers (c) again indicating that the amplification process did not apply a bias to the total number of counts observed by a given molecule.

Section 3.5: Cyclization Dependence on DNA Length

Section 3.5.1: Identifiable Bimolecular Reactions

Given the nature of the library there are three types of bimolecular molecules that can be identified as bimolecular. These three types of reactions come from inter-class AB-AB ligations (figure 3.5 c), BA-AB ligations, and AB-BA ligations (figure 3.5 d). As discussed in section 3.3, B-B ligations are only seen in high numbers when they are inter-class B-B ligations (table 3.2), this is due to the likelihood that intra-class B-B molecules will form hairpins during cluster formation on the Illumina flow cell. While it can be argued that the largest molecules in the 107 class will have higher rates of cyclization which will decrease the overall number of events observed from bimolecular ligations in this class relative to the 47 class, the fact that even across identically sized class-class interactions we see a discrepancy means that a reason outside of total size must be to blame. After the initial ligation forming a bimolecular linear molecule between a 47 class and 107 class molecule the total length is in the same size range as when two 77 class molecules ligate forming a bimolecular linear molecule. As such, the rates of dimer circle cyclization for these two instances should be roughly equivalent. Yet, while there are a total of 6,179 B-B ligation events between 47 class and 107 class molecules, there are only 3 observed intra-class 77 class ligation events.

Inter-class and Intra-class B-B reactions observed							
	Inter-class			Intra-class			
Sample	47-77	47-107	77-107	47-47	77-77	107-107	Total
0.5 nM Ligation	1650	4379	2913	24	1	4	8971
0.5 nM Ligation BAL-31 Treated	345	1687	2653	60	0	0	4745
0.05 nM Ligation	58	49	603	2	1	3	716
0.05 Ligation BAL-31 Treated	43	64	1992	29	1	2	2131
Total	2096	6179	8161	115	3	9	16563

Table 3.2 A comparison of inter-class and intra-class B-B reactions observed. In total 16,436 of the 16,563 observed bimolecular B-B reads came from inter-class ligation events. We believe that the formation A-A/B-B dimers should be random and only dependent upon collision of sticky ends between two linear molecules. As such, the number of intra-class bimolecular events should be higher than is observed but cluster formation limitations prevent detection of these molecules (as discussed above).

In BAL-31 treated samples only cyclized DNA should remain, which means in the BAL-31 treated samples the formation of a B-B junction also requires making an A-A junction. Likewise, in linear multimerization the two XhoI ends should not have any preference for ligation to one or the other XhoI end of another molecule. As such, in the event that the overall dead fraction of each end is similar, A-A molecules and B-B molecules should be roughly similar in quantity in non-BAL-31 treated samples and identical in BAL-31 treated samples. However, A-A ligations are very rarely seen in the sequenced library due to shared sequence between each of the sequence variation portions on each molecule. In total only 387 A-A ligation events were observed. Given the low incidence of A-A molecules, no conclusion can be gained from their presence/absence for a particular molecule or length.

The final class of bimolecular molecules that can be distinguished come from A-B inter-class ligation events. When two monomers ligate to become a linear dimer

there are four possible orientations: AB-A'B', A'B'-AB, AB-B'A', and BA-A'B' (illustrated in figure 3.5). The dimerization rate constant for these four ligation reactions should be equivalent across different dimers. While the rate of formation of dimer circle will depend upon total dimer length, the expected relative proportion of A-B ligation events should be equivalent to the proportion of A-A and B-B events. The discrepancy between observed A-B bimolecular population and the anticipated population can at least partially be explained by the fact that intra-class A-B dimers are indistinguishable from unimolecular cyclization events (see figure 3.4). Inter-class A-B bimolecular products do not favor any particular length. The total number of observed molecules from each size class scales with the input library concentrations, with 47 class being the most abundant followed by 107 class and 77 class as the least abundant (figure 3.10). Figure 3.11 explains the length classification for these molecules.

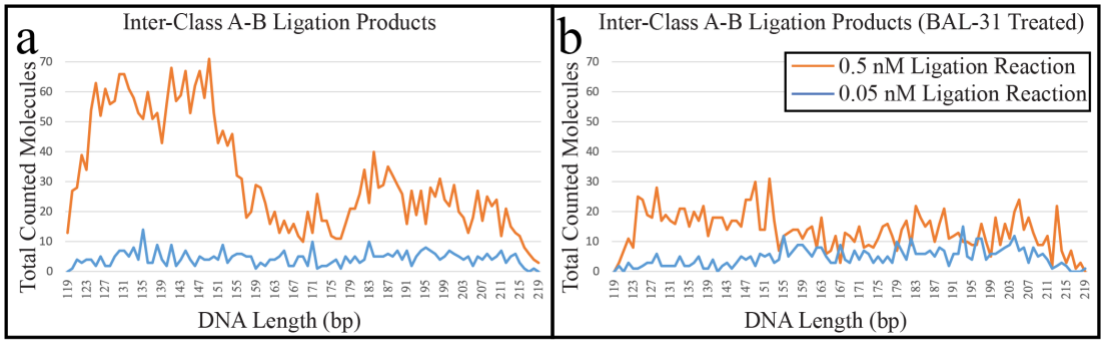

Figure 3.10 Inter-Class products by size: A comparison of inter-class A-B ligation products across all lengths before (a) and after (b) treatment with BAL-31 nuclease. The population impacted the most by BAL-31 nuclease treatment is that of 47 class molecules, suggesting that they are predominantly linear bimolecular or multimer products. 0.5 nM ligation reactions are shown in orange, 0.05 nM ligation reactions are shown in blue. For each length, all molecules composed of different sequences that resulted in that particular length are summed.

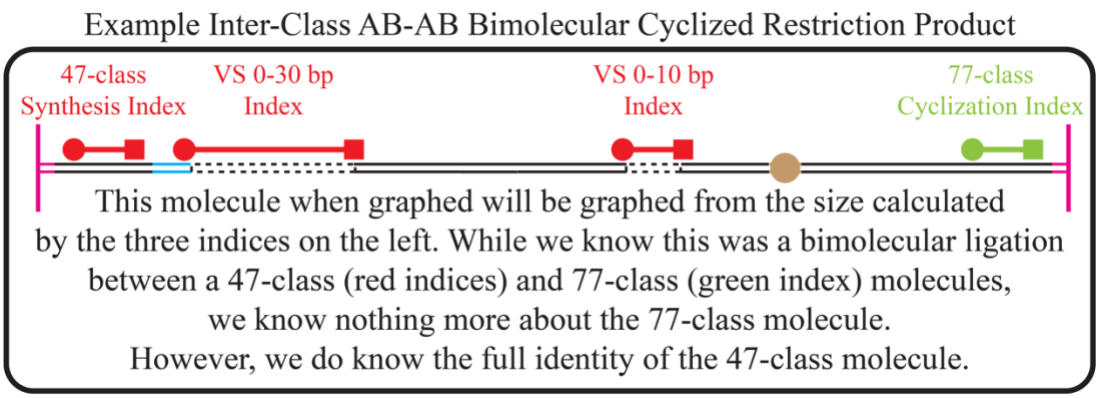

Figure 3.11 Inter-class A-B product naming: A schematic showing how naming was chosen for inter-class A-B products. In the above molecule the three indices in red are known to have come from the same original 47 class molecule that ligated to a 77 class molecule. As the full original size of the 47 class molecule can be determined the inter-class A-B length given in figure 3.10 is based solely on this value.

From figure 3.10 it can be seen that the overall difference in distribution between 0.05 nM ligation samples before and after BAL-31 treatment is negligible, however, in 0.5 nM ligation samples the overabundance of 47 class molecules is wiped out relative to 77 and 107 class following BAL-31 treatment. In the 0.05 nM ligation most of the inter-class events are likely dimer circles. Only linear molecules are digested by BAL-31, therefore the decrease in the relative proportion of 47 class molecules upon BAL-31 digestion in the 0.5 nM ligation, suggests that more of these molecules were linear. Under the workup procedure used here, it is not possible to distinguish between linear bimolecular and linear multimer products. From the relative decrease in 47 class molecules after BAL-31 digestion in the 0.5 nM ligation it can be concluded that the 47 class likely has a higher fraction of dead ends than the 77 class and 107 class.

In order to examine this further the size class pairings present from inter-class A-B molecules can be seen in table 3.3. If the "B" ends (from the SC side XhoI) from each size class have equivalent dead fractions then we would expect to see each set of A-B pairings proportional to one another because the "A" end (from the SV side XhoI) across all classes is the same. This is the case for both 77-107 pairs, however, the 47-77 pairs and 47-107 pairs show higher proportions of 47 class molecules interacting with other classes when the 47 class is the A end than when it is the B end. This further suggests that the 47 class molecules are likely to have a dead end as discussed above.

Inter-class A-B composition by size class						
	47+77		47+107		77+107	
Sample	A47-B77	A77-B47	A47-B107	A107-B47	A77-B107	A107-B77
0.5 nM Ligation	585	218	1,151	450	264	300
0.5 nM Ligation BAL-31 Treated	167	81	387	181	251	244
0.05 nM Ligation	51	21	74	62	78	110
0.05 nM Ligation BAL-31 Treated	18	9	27	24	164	178
Total	821	329	1,639	717	757	832

Table 3.3 Inter-class A-B molecules shown by composition of size class components. A side identifies a molecule coming from the SV side XhoI junction, B side identifies a molecule coming from the SC junction. If the B end across size classes has the same dead fraction then each A-B pairing should yield relatively similar amounts of products. For each of the 77-107 pairs this is true, however in both 47-77 and 47-107 pairings the 47 class B end participates less than the 77 class or 107 class B end. This furthers our hypothesis that the 47 class XhoI B end has a higher proportion of dead fraction than does the 77 class XhoI B end or 107 class XhoI B end.

A summary of all observed bimolecular ligation events that can be identified as bimolecular events is in table 3.4. As mentioned previously, the linear dimers that lead to A-B or A-A/B-B products should result in roughly equivalent amounts of the two, yet far more B-B bimolecular products are observed than A-B bimolecular

products. This is in part due to the fact that all B-B products are known to be bimolecular where some A-B bimolecular products will be indistinguishable from unimolecular products (see figure 3.4 above). Even with this in mind, the discrepancy in observed A-B bimolecular and B-B bimolecular products is greater than we anticipated and the difference is not fully understood.

Summary of All Observed Bimolecular Products			
Sample	B-B Bimolecular	A-A Bimolecular	A-B Bimolecular
0.5 nM Ligation	8,971	219	2,968
0.5 nM Ligation BAL-31 Treated	4,745	63	1,311
0.05 nM Ligation	716	46	396
0.05 nM Ligation BAL-31 Treated	2,131	59	420
Total	16,563	387	5,095

Table 3.4 A summary of all reads that can be identified as bimolecular.

Section 3.5.2: Unimolecular Reactions

As can be seen in figure 3.3 and figure 3.4, identical products are obtained from unimolecular cyclization and bimolecular A-B intra-class cyclization. As such, not all identified A-B intra-class products are definitively unimolecular cyclization products, some portion of these products are actually bimolecular. A total of 77,522 molecules were identified in this classification (table 3.5).

A-B Unimolecular and Intra-Class A-B Bimolecular Products	
0.5 nM Ligation	23,338
0.5 nM Ligation BAL-31 Treated	22,789
0.05 nM Ligation	12,075
0.05 nM Ligation BAL-31 Treated	17,320
Total	77,522

Table 3.5 A-B intra-class products: A summary of all unimolecular or indistinguishable from unimolecular A-B intra-class ligation products.

From the observed intra-class A-B ligation products we fundamentally observe the behavior we expect to observe (figure 3.12). As the concentration of DNA approaches the J factor, in the 0.5 nM ligation, the torsional modulus of the DNA can be observed in the BAL-31 digested sample (figure 3.12 b). While this effect is subtle, it can be seen in peaks that are separated by ~10.55 bp. This effect can be highlighted by dividing the BAL-31 treated sample by the non-BAL-31 treated sample (figure 3.13). At 0.05 nM the occurrence of bimolecular events is rare enough that as molecules increase in length the likelihood of unimolecular cyclization increases regardless of helical repeat, so at 0.05 nM the ratio increases and eventually plateaus. We still see the 47 class products disappearing with BAL-31 digestion as was observed above in the inter-class reactions, suggesting that many of these 47 class intra-class A-B products are actually linear dimers or linear multimers and not unimolecular circles. From figure 3.12 and figure 3.13 there is no evidence seen to indicate that unimolecular cyclization is occurring below 150 bp.

Figure 3.12 Intra-class products by size: A comparison of intra-class A-B ligation products across all lengths before (a) and after (b) treatment with BAL-31 nuclease. 0.5 nM ligation reactions are shown in orange, 0.05 nM ligation reactions are shown in blue. For each length all molecules composed of different sequences that resulted in that particular length are summed.

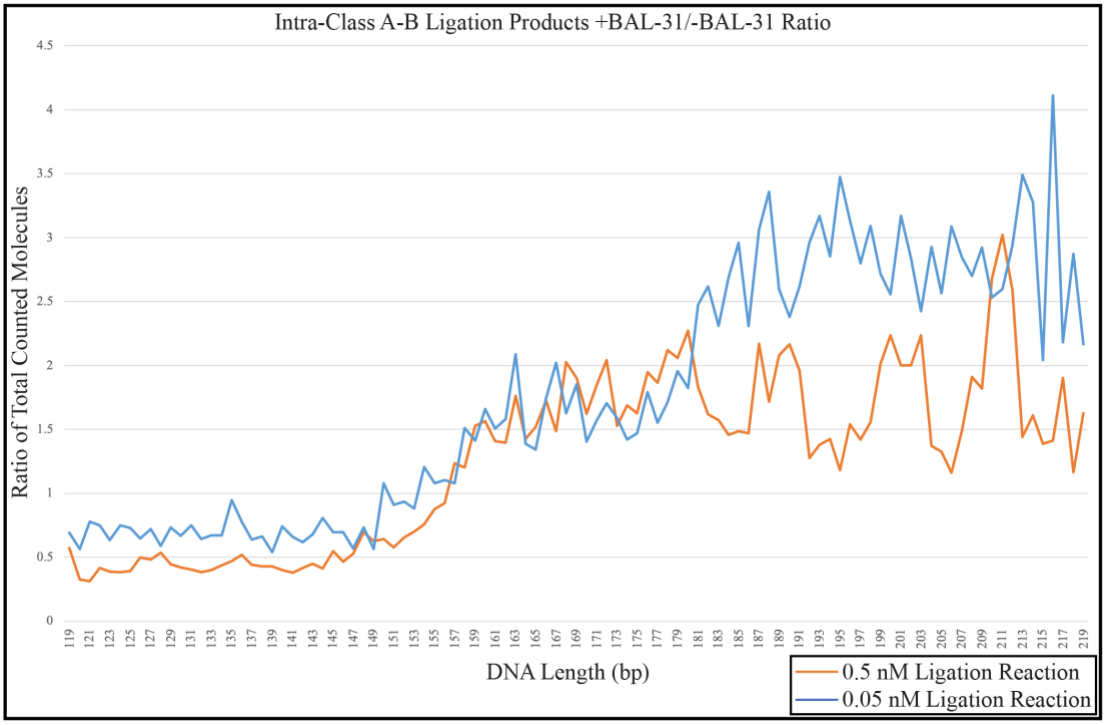

Figure 3.13 Intra-class A-B product ratio by size: A comparison of intra-class A-B ligation products across all lengths between 0.5 nM (orange) and 0.05 nM (blue) ligation reactions. As the length of the DNA increases the torsional modulus of the DNA can be seen to impact the number of molecules counted when the DNA concentration is near the J factor. The ratio is obtained by dividing the BAL-31 digested counts (figure 3.12b) by the non-BAL-31 digested counts (figure 3.12a).

A basin can be seen for both concentrations below approximately 150 bp. We believe this baseline level of intra-class A-B junction products from intra-class AB-AB bimolecular reactions, which are indistinguishable from unimolecular reactions. This is why no oscillatory nature can be observed as the lengths span full helical repeats for either the 0.5 nM or 0.05 nM reactions at low lengths even though there were molecules in this size range sequenced. If we look at figure 3.13 in comparison to the theoretical Shimada and Yamakawa curve (figure 3.14), the peaks in predicted J factor correlate in length with the peaks in the count ratio +/- BAL-31.

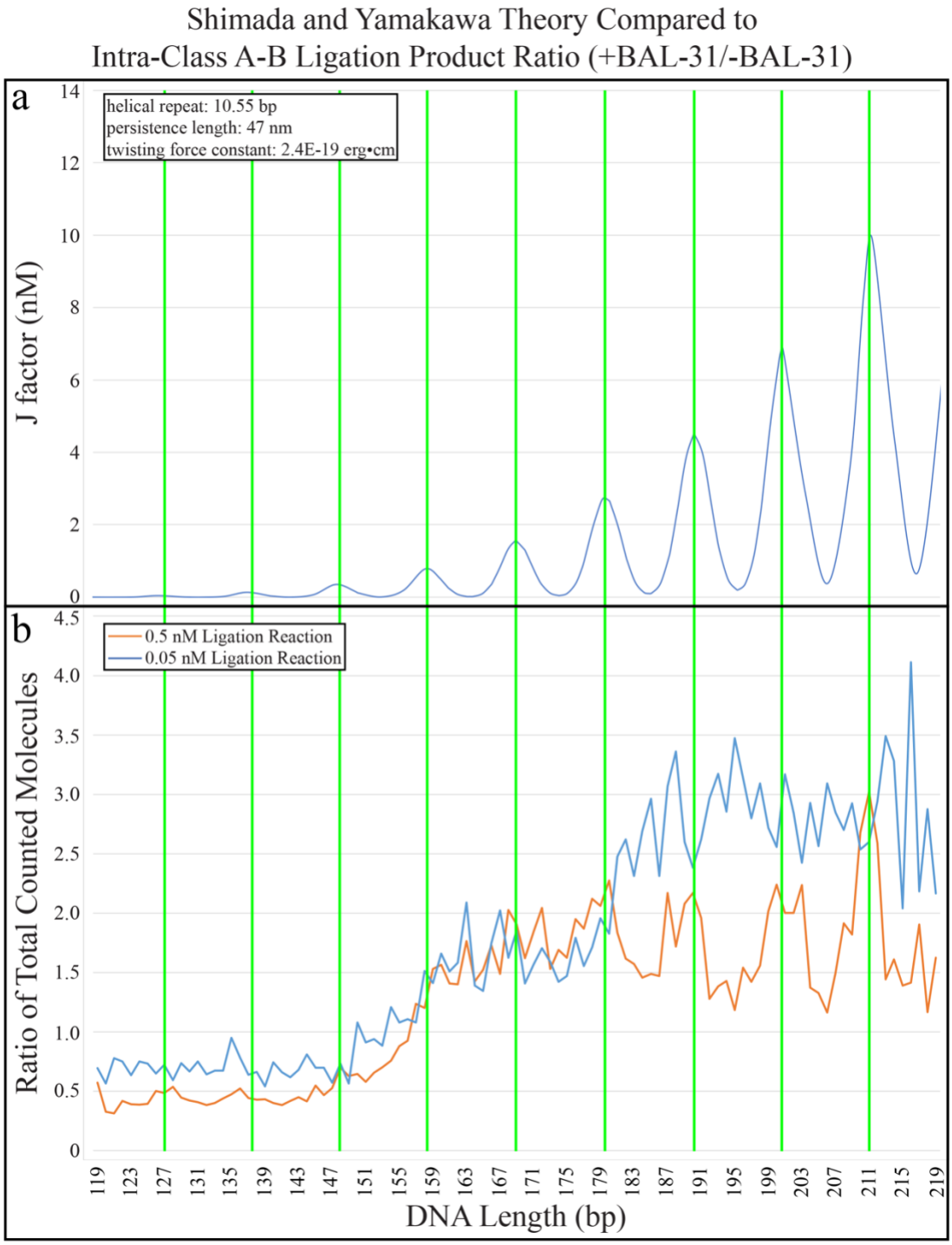

Figure 3.14 Intra-class ratio compared to SY Theory: A comparison of the Shimada and Yamakawa theory (a) to the ratio of BAL-31 digested to non-BAL-31 digested intra-class A-B ligation products (b). Vertical green lines on both graphs correspond to integral multiples of the DNA helical repeat and

peaks can be seen occurring at these lines in both Shimada and Yamakawa and within 0.5 nM ligation reaction products.

Section 3.5.3: Dead End Fractions

In order to estimate fractions of dead ends, all of the possible assembly components were programmed to be sorted.

From the non-amplified library a total of 276,593 reads were fully assembled molecules and 89,870 reads had only the sequence variant component (no size class component). From this we can estimate an upper bound of dead fraction for the BstEII synthesis junction on the sequence variable side at 24.5%. On the size class variant side, 48,953 reads had only the size class indices, giving us a size class variant BstEII dead fraction upper bound of 15%. Given the difference in means of construction for the size class variants it is worth noting that of the 48,953 reads: 15,025 are 47 class, 6,719 are 77 class, and 27,209 are 107 class. As these values are not skewed toward either the 47 class or 77 and 107 class, it can be concluded that the dead end fraction itself is not simply coming from the fact that the 47 class was a synthetic oligo. The reason the reported dead fraction percentages are an upper bound comes down to clustering efficiency on the Illumina flow cell. In the non-amplified library the range of sizes for the fully assembled library are 119-219 base pairs, but reads for any of the size class variants will be 47, 77, or 107 bp and reads for the sequence variant side will range from 72 to 112 bp.

Using the same logic looking at the non-BAL-31 treated samples we can estimate the upper bound for dead end fraction of XhoI ends. If a library molecule assembles completely it is capable of being split at the SspI site to give two

molecules, one with a single index from the size class variant and one with three indices (two variable sequence indices and one size class variant index). These fragments will also occur from a linear multimer that is capped by a dead end as shown in figure 3.6. Across the 0.5 nM and 0.05 nM reactions that were not treated with BAL-31 a total of 1,182,939 reads were found including the region from XhoI to SspI from the size class variant side. This makes up 47.9% of all identified reads from the non-BAL-31 treated samples. However, while the size discrepancies in cluster formation for the non-amplified library were small, these fragments are as small as 17 base pairs. With this in mind, dead end fractions for the XhoI ends cannot be accurately calculated from the sequencing data itself.

Section 3.6: Conclusion

From these results we believe that there is no evidence for extreme DNA bendability. As seen in figure 3.14 the periodicity of cyclization efficiencies only begins to increase after the DNA length is greater than the persistence length. We believe the basin seen in product formation of molecules below the persistence length is actually due to bimolecular and multimer formation and not unimolecular cyclization. If any unimolecular circles are forming in the length ranges below 150 bp whatever leads to these circles wipes out the DNAs torsional dependence.

Chapter 4: Modeling of Results

Section 4.1: Overview

From the previous chapter, multiple observations within the results were discussed. The first notable observation from the data is the existence of a baseline ratio of products in the sub 150 base pair length range. Due to this baseline lacking any evidence of oscillation as would be expected for straightforward ring closure, we propose the most likely cause of the existence of intra-class A-B products that are BAL-31 resistant seen in this size range is due to bimolecular and multimer circular products. As seen in figure 3.4, when two molecules from the same size class form a linear A-B dimer and then that linear dimer cyclizes, the resulting products seen in sequencing are indistinguishable from unimolecular cyclization products. Given the random nature of bimolecular ligation, the length of the linear dimer that forms from intra-class A-B ligation will not be consistent, as a given monomer will have an equal chance to be paired to any other unreacted monomer. Since the length of an intra-class A-B ligated product varies for each assigned length, no oscillation will be seen in the total counts at each assigned size.

The second unexpected result was the drastic change in pre and post BAL-31 digestion distributions in both intra-class and inter-class products. BAL-31 digestion removes linear products, the 47 size class has a considerably greater degree of digestion than either of the other size classes.

To check whether our proposal that the baseline seen in the intra-class A-B product ratio is coming from intra-class bimolecular ligation followed by cyclization

and to better understand the differences in BAL-31 digestion across size classes, we simulated the kinetics of the ligation mixture of monomers in MATLAB.

Section 4.2: MATLAB Modeling

Section 4.2.1: Modeling Setup

The full code of the MATLAB program used to model results can be found in Appendix 5.3. A full list of each tracked species can be found in Appendix 5.4. In short, the model integrates the differential rate laws for all of the species of interest to predict concentrations as a function of time. Input parameters include: monomer concentrations, monomer dead end fractions for each end of each monomer, the rate constant for cyclization of each monomer (k_c), the common rate constant for bimolecular ligation (k_b), rate constants for bimolecular cyclization (k_{dc}), and a rate constant for multimer cyclization (k_{mc}). For each of these input variables an experimentally measured number or a range of best guess inputs was used and then we varied rate constants and dead end fractions until the results qualitatively resembled our data.

The input monomer concentrations were chosen to be proportional to the overall counted population abundance of each class (47, 77, and 107) in the non-amplified library. All 341 SV variants were collapsed into their individual size class. Dead end fractions were based upon initial guesses from the sequenced non-amplified library and adjusted to align with what we see in the non-BAL-31 digested samples. The dead end fraction for the "A" side of the each class is assigned to be identical, as the "A" side is coming from the same sequence variant library prep and is constant

across all three size classes. Rate constants were chosen to reflect values that are in line with the Shimada and Yamakawa theory when compared to the other rate constants in the model. As the time in the model is arbitrary and unitless, these values are only meaningful with respect to one another.

Section 4.2.2: Modeling Conclusions Regarding Cyclization Baseline

Over time the monomer concentration can be seen decreasing as unimolecular circles and bimolecular linear products build up. Linear bimolecular products then either form dimer circles if they have two live ends or multimers if they have either one or two live ends. The output from the set of input parameters that we believe to provide the best fit to our results can be seen in figures 4.1 and 4.2. In figure 4.1 the difference in starting monomer concentration can be seen yielding a large end monomer concentration for 47 class monomers. When the bottom tenth of the concentration range is zoomed in upon, you can see the product distribution of the rest of products capable of giving intra-class A-B products. What can be seen in this model is the growth of 47 class intra-class bimolecular A-B circular products (dark blue line) with relatively few 47 class unimolecular (dark green, barely above y-axis) products forming. With each intra-class bimolecular A-B cyclized product appearing twice in the sequenced library, the observed number of counts will be twice that of a unimolecular product of the same concentration. As such, while the dark blue line is lower in value than the 77 class and 107 class unimolecular circles, the apparent concentration by count will be much closer to the 107 class unimolecular circles.

In addition to 47 class intra-molecular circle formation, there are three other products seen in the model that will increase the number of intra-class A-B products

observed. These are intra-class A-B linear dimers (47 class dimers in dark red), linear multimers, and cyclized multimers. While the multimer products will be a mixture comprised of each of the different size classes, the 47 class is the largest contributor to multimer formation. This can be seen within the model by killing the 47 class molecules with 100% dead ends while leaving all rate constants and concentrations the same. With the majority of multimers including 47 class monomers, we can anticipate that a large number of both A-B and A-A/B-B reactions will be seen within the multimers that appear to be 47 class intra-class. From this model we can see that observed intra-class A-B junctions for 77 class and 107 class products are predominantly formed from unimolecular cyclization while the observed 47 class basin in figure 3.13 is due to the formation of products indistinguishable from unimolecular circles, not an intrinsic property of the DNA molecules of this length to cyclize.

Model Input Parameters:

k_c: [0.01, 0.9, 2.25] (time^{-1})

k_b: [3] (conc^{-1} time^{-1})

k_{dc}: [2.5, 4, 5, 5, 6, 6.5] (time^{-1})

for respective dimer circles 47-47, 47-77, 47-107, 77-77, 77-107, and 107-107

k_{mc}: [6.5]

[Monomer]: [0.5, 0.08, 0.1] (concentration)

Dead fraction A: [0.1]

Dead fraction B: [0.5, 0.02, 0.1] (different per monomer)

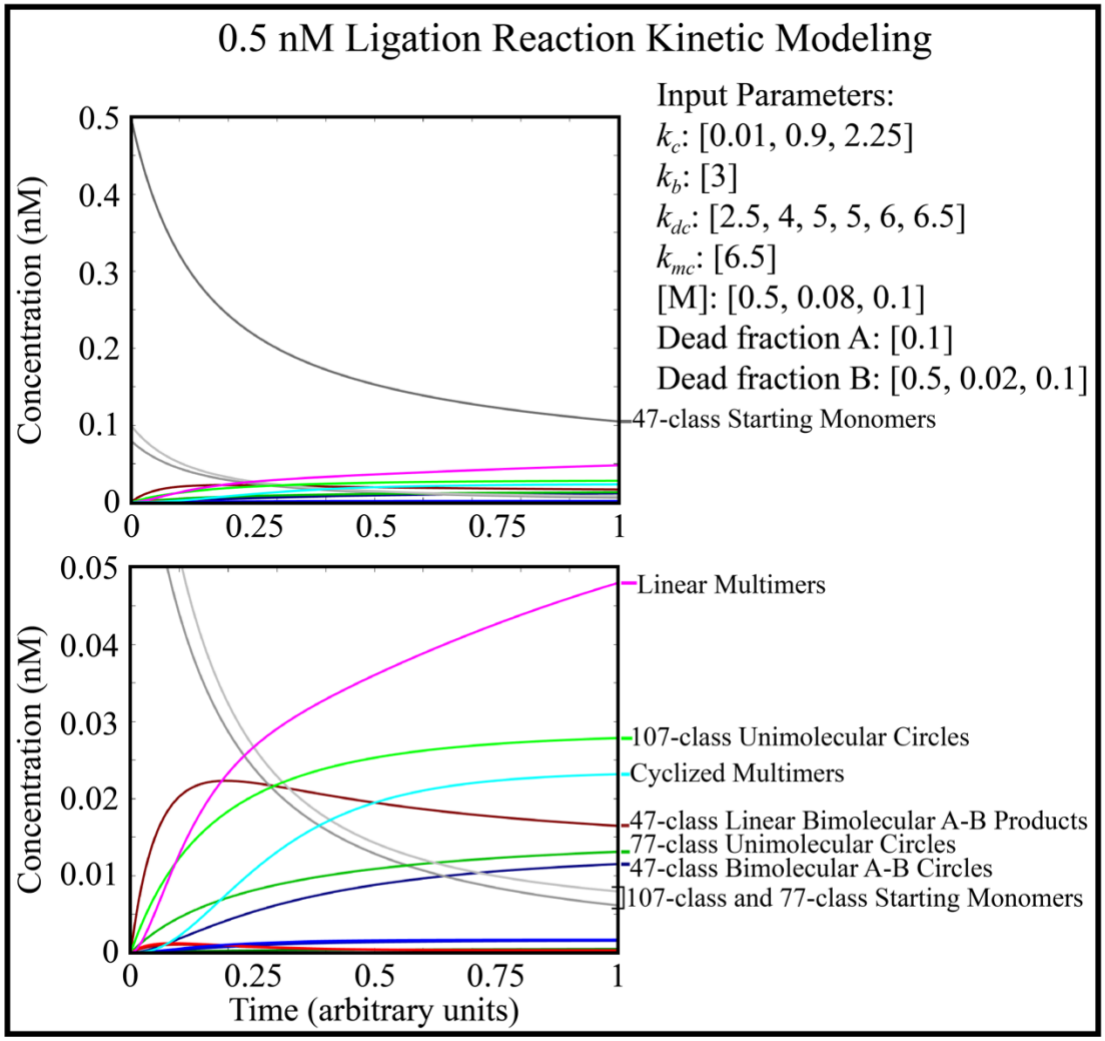

Figure 4.1 Kinetic modeling of 0.5 nM ligation: A model of the change in product distribution over time for all intra-class A-B ligation products and multimerization products in the 0.5 nM ligation reaction. These products are indistinguishable from the corresponding unimolecular circles upon sequencing. Parameters for the model are shown in the figure. Each set of curves for a given product type is a gradient from darkest to lightest corresponding to 47 class, 77 class, and 107 class respectively.

Section 4.2.3: Modeling Conclusions Regarding 47 class Disappearance Post BAL-31 Digestion

In looking at the intra-class A-B products before and after BAL-31 digestion, the 47 class molecules appear to be preferentially digested relative to other classes in the 0.5 nM ligation reaction but in the 0.05 nM ligation reaction there is only a slight preference for digestion of the 47 class size products (figure 3.11). As seen in figure 4.1, a large amount of the 47 class molecules participate in ligation reactions leading to linear bimolecular products and linear multimer products. In order to see the overall impact of BAL-31 digestion at both ligation reaction concentrations, the model was run using both input concentrations (figure 4.2). With DNA concentrations as they are in the 0.5 nM ligation reaction, the observed number of linear multimer products is substantial, but at the 0.05 nM ligation reaction, very few linear multimer products form. This fundamentally makes sense: a decrease in concentration will result in fewer bimolecular collision events necessary for formation of larger molecules. When only circular products (those that would survive BAL-31 digestion) are graphed, we can see why the 0.5 nM reaction shows a drastic decrease in 47 class molecules relative to the other two classes after BAL-31 digest but the 0.05 nM reaction does not. In the 0.5 nM reaction there are two large contributors to the overall observed 47 class intra-class A-B products that are BAL-31 sensitive: 47 class intra-class linear bimolecular A-B products (dark red) and all linear multimers (pink). As discussed in the previous section, the composition of these linear multimers is in fact mostly 47 class. When looking at the 0.05 nM reaction both of these classes are substantially decreased such that their contribution to the total

number of counts is much less than at higher concentration, so BAL-31 digestion makes very little difference to the observed product distribution.

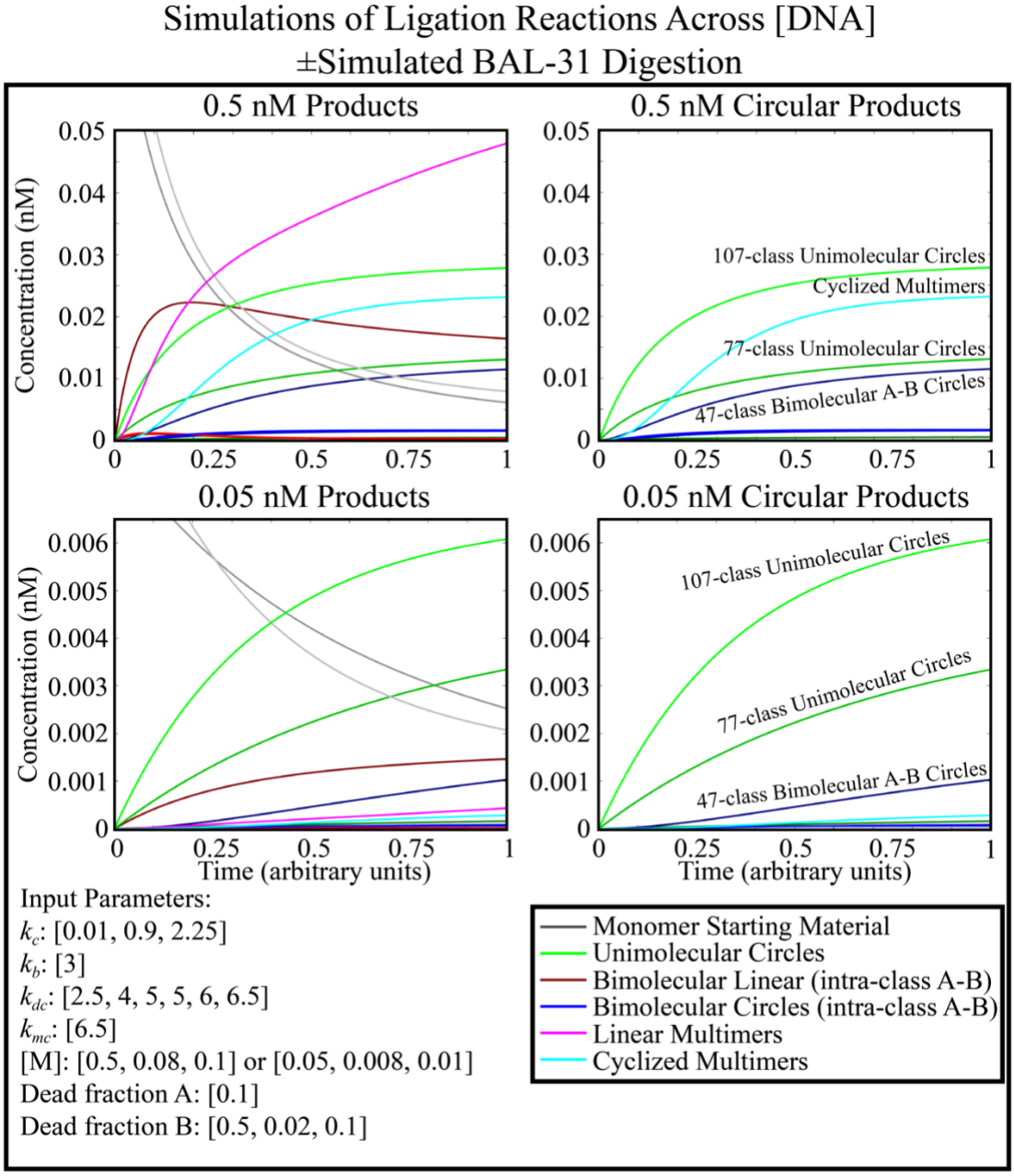

Figure 4.2 Kinetic modeling with BAL-31 digestion: A model of the change in product distribution over time for all intra-class A-B ligation products and multimerization products, for both the 0.5 nM and 0.05 nM ligation reactions. Parameters for the model are shown in the figure. Each set of lines is a gradient from darkest to lightest corresponding to 47 class, 77 class, and 107 class respectively. The

panels on the left show the product distribution as it is expected to be seen in the sample that has not been digested with BAL-31, the right two panels show the product distribution as would be seen after digestion with BAL-31 (only cyclized products).

Section 4.3: Discussion of Results

We believe the above modelling bolsters our conclusions from chapter 3 that our observed results demonstrate no evidence for hyper-bendability in DNA that would allow for unimolecular cyclization at short DNA lengths. While we cannot conclusively rule out unimolecular cyclization from these results alone, the observed counts can be explained by the presence of bimolecular and multimer product formation. Additionally, the shift in relative populations observed between non-BAL-31 digested and BAL-31 digested samples can be explained by the presence of linear bimolecular and linear multimer molecules. In the model these linear products are at higher concentrations coming from 47 class in part because of the excess of 47 class entering the experiment, but also in part because we believe the 47 class cyclization junction to contain a greater dead fraction. Prior to creating the size class DNA segments, we checked for potential secondary structures using the RNAstructure online interface from the Mathews Lab[47]. None of the sequences showed structural components outside of the desired base pairing using the RNAstructure tool, and we moved forward using these sequences. After ligation, sequencing, and analysis it became clear that the 47 class variant may have a higher dead fraction than we anticipated. This can be seen by both the preferential digestion of products that include the 47 class as well as looking at the inter-class A-B pair presence of 47 class XhoI end. In looking at the XhoI end of the 47 class sequence by

hand, a pseudoknot could be drawn. Using the KineFold DNA pseudoknot prediction software, the full pseudoknot was not formed, but the basic structure needed to form the pseudoknot is generated (figure 4.3)[48].

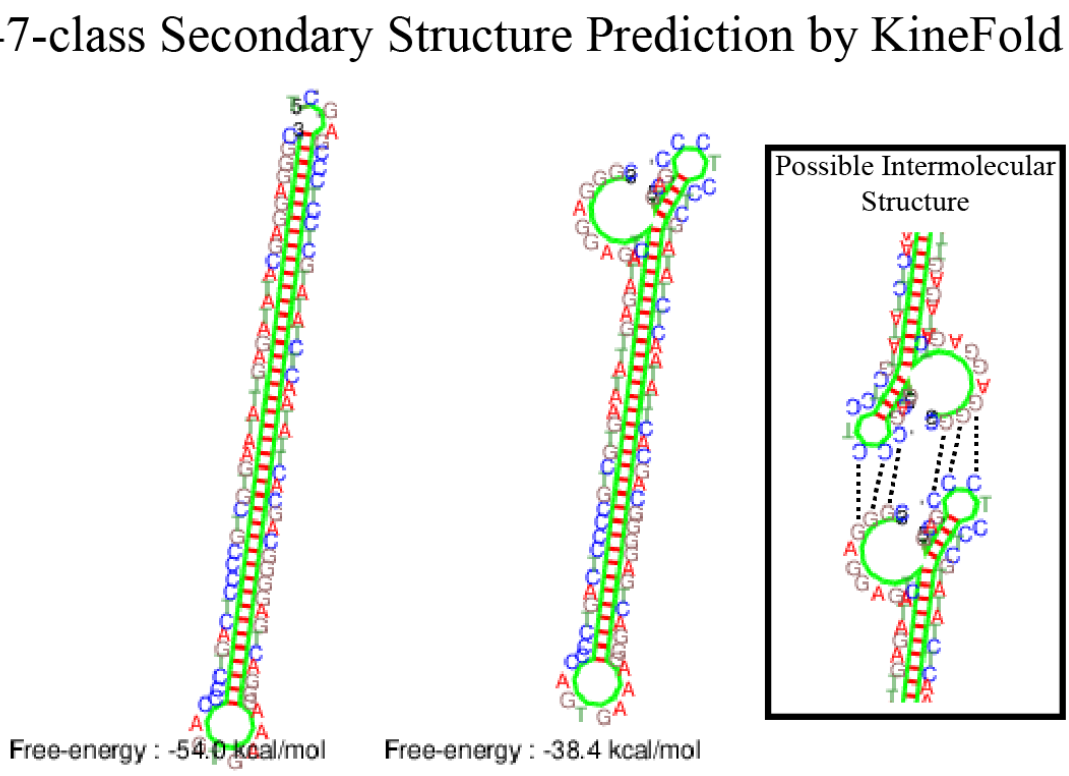

Figure 4.3 SC 47-class secondary structure: The desired secondary structure of the 47 class molecule with the XhoI overhang visible at the top can be seen on the left. A nearly identically folded 47 class molecule with a single stranded flap can be seen in the middle. It is possible that this structure will in fact form a pseudoknot where the 3' end that is not annealed in the prediction software (…-GGGC-3') will anneal to the C triplet in the adjacent hairpin, forming a pseudoknot. An additional possible structure that may form between two misfolded molecules is shown to the right where the flaps from two mis-annealed 47 class ends form a structure with one another. In order to obtain these structures the two strands of the 47-mer were entered contiguously with an insertion of "AAA" between the sequences from the two strands to allow for a hairpin to form half way through the sequence in the KineFold software thus allowing the desired doubled stranded sequence to anneal.

Chapter 5: Conclusions and Future Directions

Section 5.1: Demonstration of High-Throughput Sequencing Applied to Length Libraries

The successful application of high-throughput sequencing to a length library as a means of assaying a ring closure experiment opens up several avenues for future experimentation. By tracking all of the inputs and all of the observed products this method can be used on a refined set of sequences and with minimal experimental changes as detailed in section 5.2 to allow for accurate J factor determination. Along with these changes, the sequence variation side of the library construction was designed to flow directly into DNA looping experimentation as detailed in section 5.3. Additionally, through the work provided here further applications of this method to a broad enough sequence library could allow for the discovery of previously unknown intrinsically bent sequences.

Section 5.2: Future Library Design Directions

While much was learned from these experiments, if we were to design the experiment again knowing what we have learned there are some changes that would improve the results obtained from this method. First, as many bimolecular reactions were not visible due to hairpin formation and many other bimolecular reactions were not visible as they were indistinguishable from unimolecular reactions, we were unable to calculate absolute J factors from our data. While conclusions on the overall existence of closed unimolecular circles at short lengths could be made, application of

sequence dependent length data to help improve coarse grain models[11] requires the calculation of J factors for each size and sequence.

Section 5.2.1: Future Changes to Size Class Variant Design

Three changes to the size class variants could be performed to help improve analysis of the results. First, our experimental design utilized one sequence at each size class. By doing so this opened us up to errors such as the high XhoI dead fraction in the 47 class as discussed in chapter 4. If more sequences were utilized for each size class, there would be multiple independent size classes at each length. With this, the likelihood of a sequence dependent dead fraction being uniform in all sequences of a single size decreases. The addition of more sequence variation in the size class variants also would decrease the abundance of intra-class A-B products that are indistinguishable from unimolecular products. This becomes of particular interest when at shorter lengths where bimolecular and multimer products dominate.

The second suggested change to the size class variants is the addition of an intermediary set of size classes between each existent size class. Right now there are 47, 77, and 107 size classes; by adding size classes that at 62 and 92 base pairs in length abrupt transitions between size classes that are partially due to concentration differences could be minimized. If these classes were added they also should be added with more than one sequence at their size.

The final change that I suggest to the size class variants is the addition of a very short class. While our work indicates that cyclization below the persistence length of DNA is highly unlikely, if we wish to fully resolve any remaining controversy about the cyclization efficiency of DNA around 100 base pairs we need

to span this size. A 17 base pair size class would result in assembled library molecules of 89 base pairs to 129 base pairs. If this size class is added it should be done making sure that the concentrations are accurate and with special care to make sure only assembled molecules enter the library. At 17 base pairs if this class failed to assemble, the size of fragments after restriction with SspI would be so small that they would cluster very efficiently on the Illumina flow cell. For this reason, to prevent clustering efficiency issues from arising I would suggest gel purification of the assembled library and not just pre-assembly components if this size class is added.

Section 5.2.2: Future Changes to Sequence Variable Region Design

Modifications to the sequence variable region are not absolutely necessary. However, J factors cannot be calculated not knowing the true bimolecular concentration. If adequate sequence diversity in the size class were instituted as discussed above all A-B ligations should become visible. Likewise, B-B intra-class ligations that form hairpins and therefore do not cluster should diminish drastically. Knowing the full A-B bimolecular and B-B bimolecular ligation product distribution would allow for accurate estimation of the true quantity of A-A products without requiring redesign of the sequence variant region. This would only be an estimate though, if significant dead fractions from any of the components is present the estimates of A-A products would be inaccurate. In order to truly count all A-A products and not rely upon estimates, the sequence variable region should be modified to contain greater sequence diversity to prevent hairpins during cluster formation.

Section 5.2.3: Future Changes to Experimental Design

In combination with or independent of the above suggested sequence changes, the experimental workflow would benefit from other modifications. Performing the experiments at a larger scale such that a greater quantity of DNA is submitted for sequencing would allow for the amplification step to be skipped and would increase the total number of reads coming from desired molecules instead of from contamination (or control sequences). The limiting factor that prevented our pursuit of this route was the total amount of library generated. Much more of each library component would need to be generated to be able to perform the experiments with at least ten times as much DNA. In total 11 ng of DNA was submitted to IGS across the five amplified samples while a minimum of 50 ng per sample was requested.

A second experimental change that I would suggest is the addition of a third ligation concentration at 5 nM. This addition lets the experiment span a full 100 fold change in concentration. This change can easily still be performed while using the MiSeq. Up to 12 adapters can be used per flow cell and only 6 adapters were used for the experimentation presented here. By adding an additional concentration +/- BAL-31 digestion 8 samples with 8 different adapters would be needed. Finally, I would suggest varying ligase concentrations utilized and/or collecting multiple time points as the reaction progresses. If this were to be done there are many ways to still operate the experiment on a single flow cell, the easiest of these is to double index the samples. While the total number of reads per sample decreases, the number of distinguishable input samples can be expanded virtually indefinitely[49].

The third experimental change that I would suggest is using a synthesis junction other than BstEII. A restriction enzyme which yields a longer overhang would be ideal. The best way to do this while retaining an asymmetric overhang is to use a restriction enzyme that cuts outside of its recognition site. Using such an enzyme would allow for the overhang to be entirely designed by hand to avoid any self annealing.

Section 5.3: Future Directions in DNA Looping

Work by Dr. Daniel Gowetski using a family of leucine zipper dual DNA binding proteins (LZD) demonstrated the ability to use an artificial protein to induce DNA looping[34]. This family of proteins uses a coiled-coil alpha helix with a CREB DNA binding site on the N-terminus and an INV-2 DNA binding site on the C-terminus of the leucine zipper. In using this protein family for DNA looping the protein is believed to be rigid, minimizing the overall variation in binding site location within three-dimensional space. As such, the changes that occur in forming protein-DNA loops with LZD at different DNA lengths are indicative of changes occurring within the looped DNA alone.

To this end, a sequence library for looping has already been designed for LZD-DNA looping. This library design uses the same structure as discussed in chapter 2 and the sequence variation side of the library utilizes the same 341 molecules already generated. In fact, with this application in mind, the sequence variation side of the library was designed to include a CREB binding sequence. Instead of three size class molecules, a set of nine large size class molecules has been designed (figure 5.1).

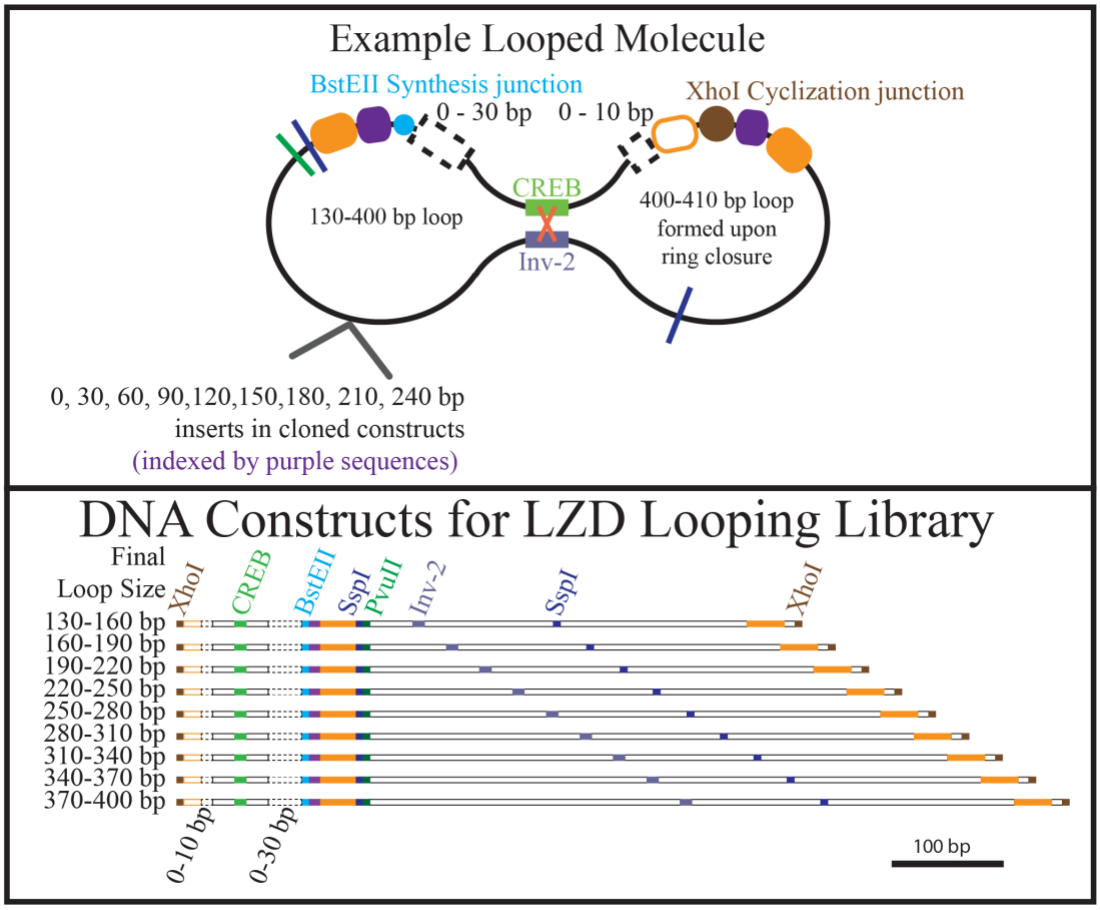

Figure 5.1 SV looping library construction: Looping library using SV library annealed to nine large size class molecules capable of undergoing looping with the LZD family of proteins. From end to end the total size of these DNAs would be 530 bp up to 810 bp with internal loops ranging from 130-400 bp. Internal loops increment up by 30 bp using the 0-30 bp insert in the SV library, while the 0-10 bp insert ranges a full helical repeat allowing for all looped molecules to cyclize to the constant length outer fragment from the other end of the molecule.

As seen in figure 5.1, the portion to the left of the BstEII synthesis junction is identical to the sequence variation library constructed in chapter 2. The portion to the right of the BstEII synthesis junction in figure 5.1 contains a constant external loop that is 400-410 base pairs in length upon protein binding and ligation of the two XhoI ends. Figure 5.2 demonstrates the overall schematic for DNA loop formation and

ligation using LZD. These nine looping size class molecules have been generated via PCR using pVx6(448) as a template[34] and cloning into pUC18 (figure 5.3).

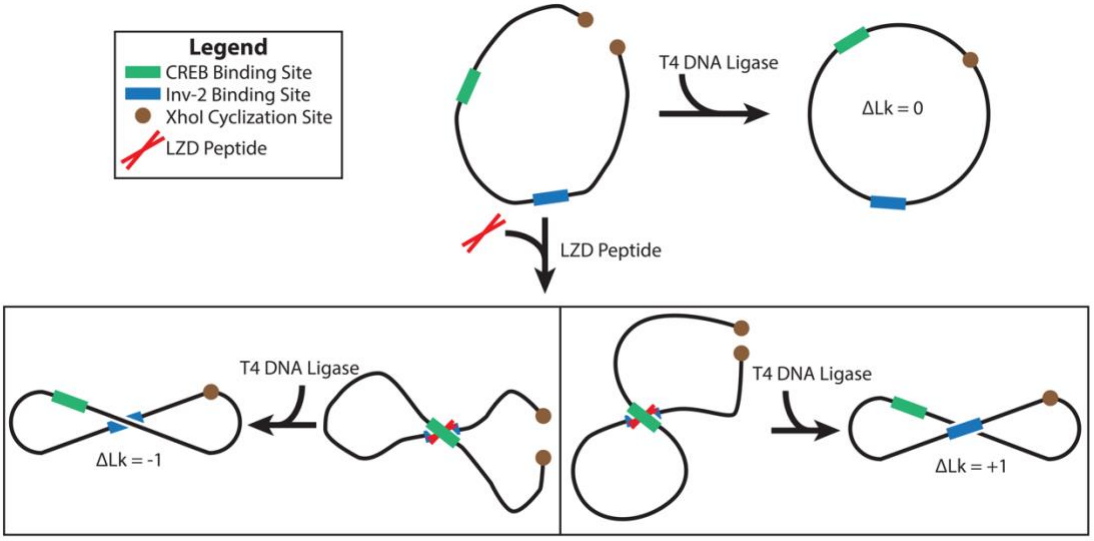

Figure 5.2 Looping library topoisomer formation: A schematic showing the formation of Lk ±1 topoisomers after cyclization of looped DNA complexes bound to LZD.

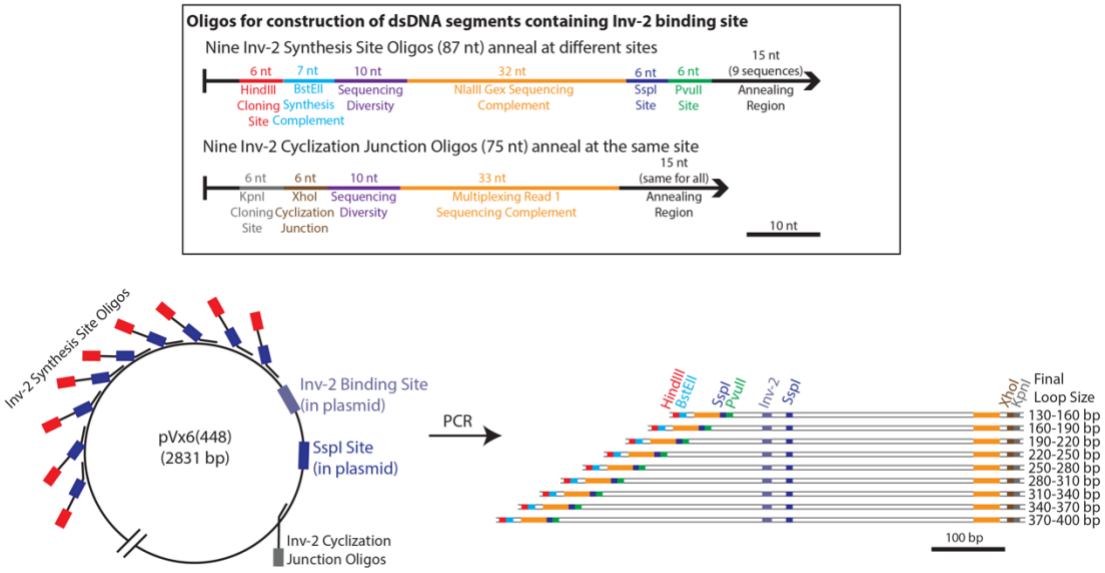

Figure 5.3 Formation of looping library: A schematic showing the formation of the 9 external Inv-2 binding site containing size class molecules. These molecules contain a BstEII synthesis junction to allow them to be assembled with the SV library as shown in figure 5.1.

As discussed in section 5.2, a greater number of sequences used in the size class region would increase the ability to differentiate between any potential bimolecular products. In this design the use of additional size class molecules may also be beneficial, but as there are already nine variants it may not be necessary. The sequences for the nine size class inserts that have been designed are present in appendix 4.3.

Section 5.4: Conclusions

DNA flexibility in nature affects the ability for a biological organism to package its genome and modulate gene regulation. One important means of studying DNA flexibility is through DNA ring closure. Decades of research have solidified our understanding of the impact of the helical repeat on DNA flexibility and the minimum size where DNA will cyclize without the introduction of kinks, localized melting, or other abnormal deformations to the DNA. In 2005 it was suggested by Cloutier and Widom that the previous decades of concurrence within the field was wrong[27]. While others have looked into these claims[37], there is still a controversy within the field about the hyperflexibility of short DNAs[22,23].

DNA ring closure experiments were performed on a library of 1023 different sequences from 119 to 219 base pairs in an effort to further resolve this controversy. While a baseline amount of products that result in intra-class cyclization is observed to occur with molecules we identify as being within the 119-159 base pair range, we believe this baseline is coming from bimolecular ligation followed by cyclization. We suggest changes to the methods used that would decrease the amount of bimolecular or multimer products that are simply indistinguishable from unimolecular products in

our current workup. If unimolecular cyclization is in fact occurring in this length range, it is occurring in a manner that wipes out the oscillatory impact of the helical repeat on cyclization efficiency. This furthers the idea presented in the model as random intra-class bimolecular circular products will not display an oscillatory nature to ring closure as only half of the needed information is known about the full cyclized length.

Further work in this field using the high throughput methodology we have developed is not only useful to better understanding DNA flexibility on its own, but also to better understanding DNA looping. To that end, many aspects of the method were intentionally developed to be extensible to future work. The library design can easily add different and additional size class variants and the MATLAB and Perl scripts were written with future applications in mind.

Appendix 1: Toward Protein-DNA Nanostructures: Creating DNA Triangles

Appendix 1.1: Overview

While DNA based nanostructures were initially developed in the 1990's as small two-dimensional building blocks[50], these structures had little practical functionality. Recent developments, as reviewed by Saccà and Niemeyer, show a broad range of potential future avenues of development for biologically based nanostructures far beyond what was initially possible[51]. Complex artificial nanostructures have been developed that are capable of functioning as DNA walkers, as simple molecular robots, and even as artificial controls on the function of already existent enzymes[52–54]. Beyond using nucleic acids as nano-machines, researchers have been able to decorate programmed structures with artificial ligands for the purposes of developing rudimentary biosensors[55,56]. At present, while proteins and nucleic acids have been used together in these nanostructures, the scaffold has been heavily reliant upon non-biological building blocks or strictly on nucleic acids.

The Kahn Lab has developed and produced a family of rigid artificial leucine zipper dual-DNA binding (LZD) proteins[34]. These LZD proteins consist of a coiled-coil α-helix homodimer with CREB and Inv-2 DNA binding regions on the N- and C-termini of the dimer respectively. Presently four LZD length variants have been developed, whose development and purification is discussed in Appendix 3. Previous work has been done using the LZD proteins as DNA looping proteins, however, as rigid coiled-coil α-helices the LZD family may also be ideal for use in the

development of protein-nucleic acid nanostructures. While current DNA nanostructures are limited by the biophysical properties of the nucleic acids themselves, the introduction of the LZD family offers a range of binding angles through which a great number of potential structures can be imagined where two portions of DNA are stapled together by the LZD. This could allow fairly simple biochemical manipulations to lead to a wide array of different nanostructures.

As a proof of concept, DNA triangles were designed by Dr. Jason Kahn as described in appendix 1.2 below. Using these triangles one of the simplest structures that could be formed is a 2-D grid as seen in figure A1.1, where two different triangular dsDNAs are held together in a network by LZD87, which is expected to hold its two dsDNA partners parallel to each other.

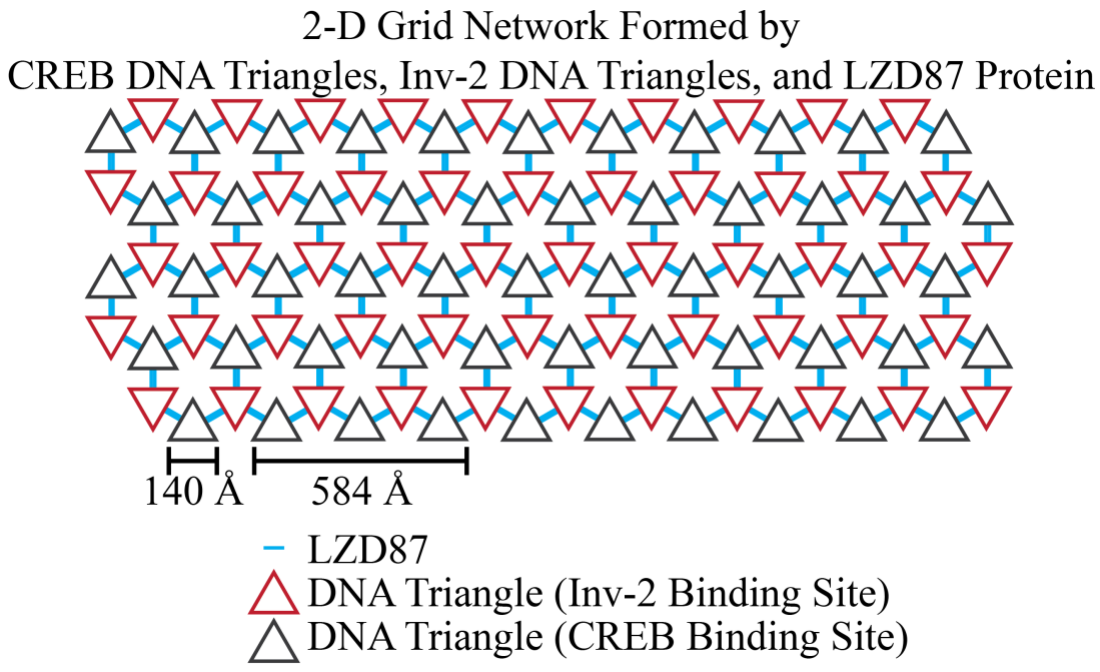

Figure A1.1 2-D DNA-protein nanostructure: A two dimensional network formed using DNA triangles that contain either a CREB binding site (red) or Inv-2 binding site (grey). The two types of triangles are held together by LZD87 (blue) in this network.

Appendix 1.2: DNA Triangle Design

The DNA triangles designed by Dr. Jason Kahn have 41 bp dsDNA sides with corner angles created by dAATAA bulge sequences[57]. The triangle is formed from 3 pieces that each create a corner and parts of two sides. Each segment is formed by annealing two ssDNA oligonucleotides of different lengths, the two oligonucleotides being reverse complements of one another except for the sequence AATAA is inserted into one of the strands. The resulting structure is two B-form segments of DNA with a bulge. The bulge seen in the NMR structure with the nucleotides continuing stacking on top of the underlying helix (figure A1.2).

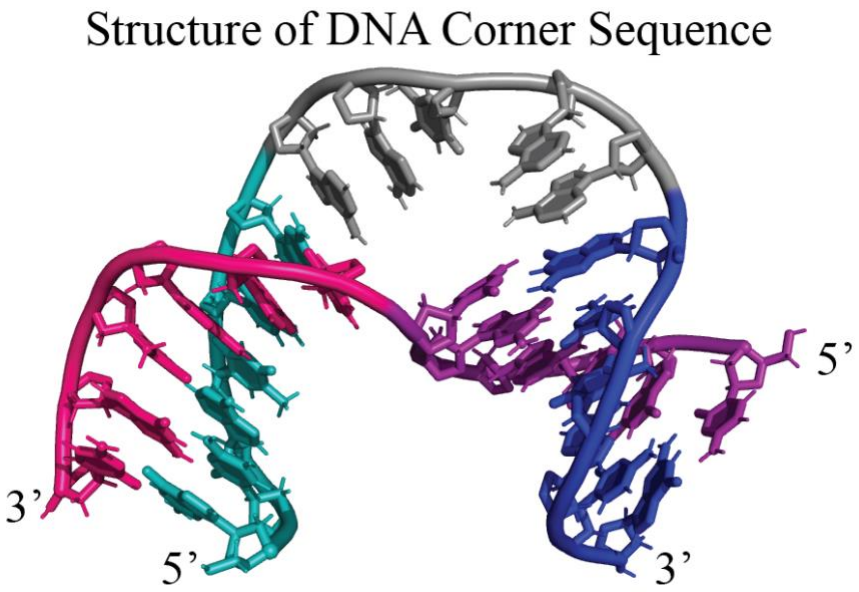

Figure A1.2 A crystal structure of the dAATAA artificial DNA corner[57]. The blue and cyan segments of one strand are interrupted by a five base insertion. The cyan and pink DNA maintain traditional base pairing as do the purple and blue segments. Yields an approximately 104 degree angle.

The triangle was designed by extending the dsDNA at the ends of the corner structure. Based on the orientation of the bases going into the corners it was decided that 41 base pair long edges between the corners would be the most likely to yield

actual triangles. With this decision in mind, Dr. Kahn designed four DNA sequences (two for CREB triangles and two for Inv-2 triangles) as seen in figure A1.3.

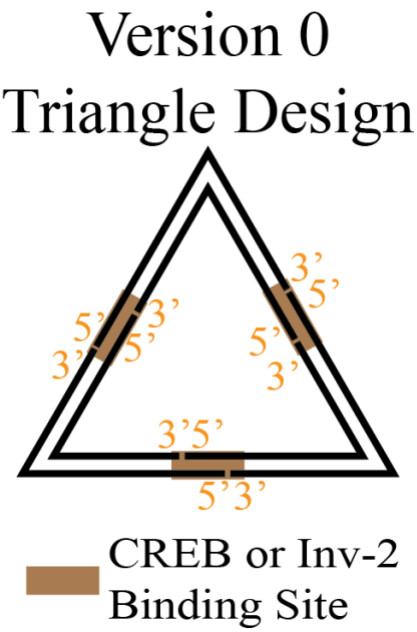

Figure A1.3 Version 0 dsDNA triangle design: The version 0 design schematic as designed by Dr. Jason Kahn. Both Inv-2 and CREB binding site containing triangles could be formed using this design using only two oligonucleotides for each assembly.

The overlapping regions between individual oligonucleotides in the version 0 design were internal to the CREB or Inv-2 binding region. In the crystal structure the corner is seen with a kink angle of 104° ±12°[57]. At a kink angle of 120° perfect triangles would be expected to form, however, at a kink angle of 90° squares would be expected to form. As the sticky ends in this design are all identical, there is no means of driving formation of DNA of only one geometry. With this limitation in mind, a new triangle design was made by myself. In this new design (version 1, figure A1.4) the corner-to-corner annealing region was moved out of the binding site uniformly in the same direction. By doing this, the annealing site could be changed for each edge of the triangle. With distinct corners and annealing sites, by swapping

the appropriate oligos during assembly, either triangle or square formation can be performed using the same corner sequence (oligo 3a is replaced by 3d and both 4d and 4a are added to switch from a triangle to a square).

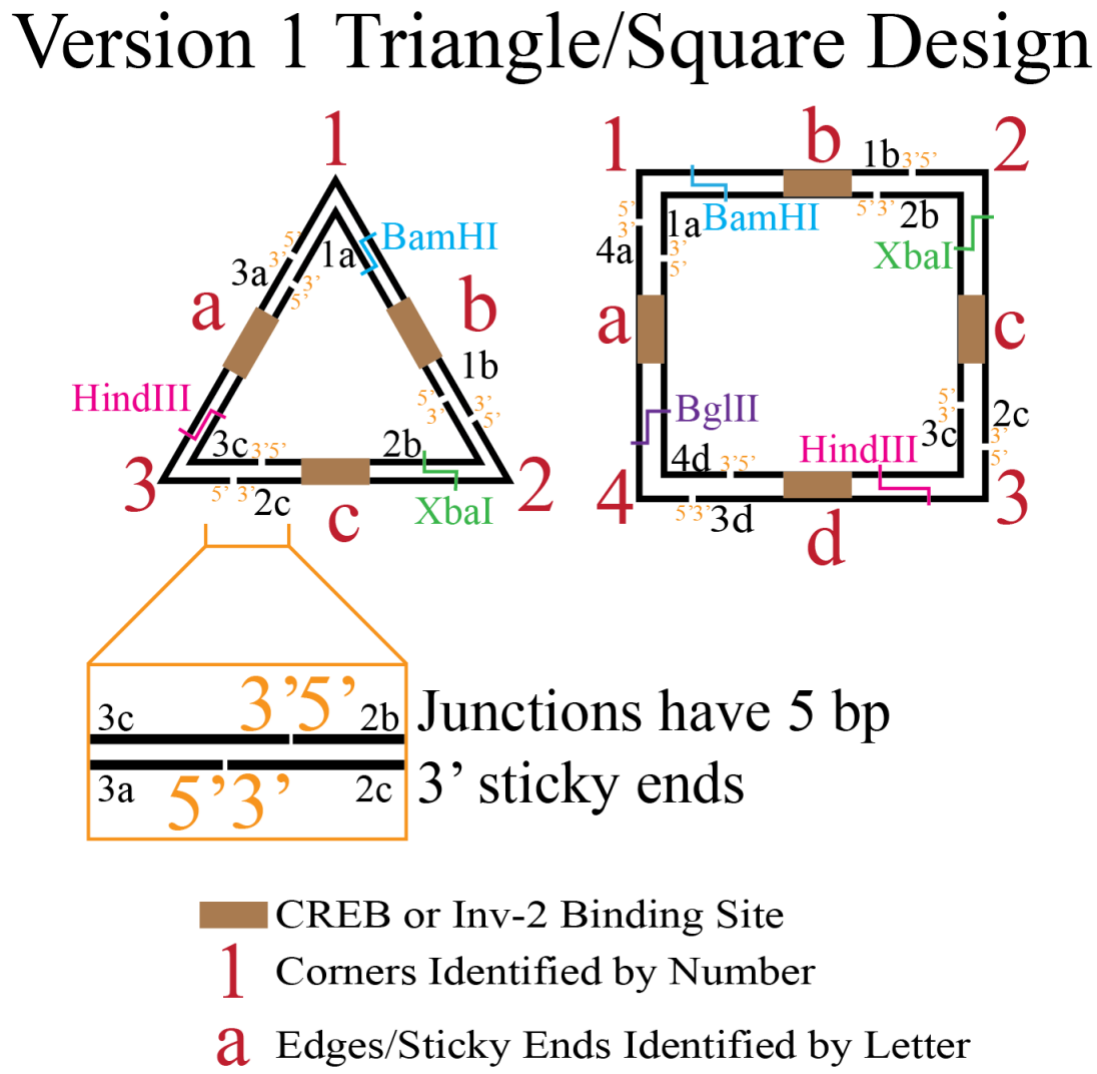

Figure A1.4 Version 1 dsDNA triangle design: A schematic of the design of DNA triangles and squares with different annealing regions (a/b/c/d). The brown boxes are capable of being either a CREB binding sequence or an Inv-2 binding sequence. In addition to having different annealing sites, each corner has a different restriction site inserted as a means of assaying successful assembly.

Within this design individual oligonucleotides are named: binding site identity (CREB or Inv-2), corner number (1, 2, 3, or 4), edge letter where overhang will occur

(a, b, c, or d), and spacer length (length from corner bulge to binding site e.g. Inv1a15). The spacer length is the length in base pairs from the corner to the binding site. The 15 bp spacer is our best guess for a triangle that will have DNA binding sites for LZD positioned such that, the bound LZD will be in plane with the triangle. By moving to a spacer length of 14, the direction of the bound LZD should change by approximately 34°. Using this new design, Inv-2 triangle oligos were ordered, individually phosphorylated and ligated (see appendix 1.3). Each of the possible corners, edges, and triangles was digested with the appropriate restriction enzyme to verify appropriate assembly. Products were analyzed on a native 10% polyacrylamide gel (75:1) containing 1x TBE.

From this test assembly and the gel in figure A1.5 we can conclude that both corner 1 and corner 2 are capable of annealing to themselves and forming double corners (see figure A1.6). This is also visible in the c edge where the 3a sticky end self-annealed to form a double edge. In addition to the overhangs self-annealing, a large amount of the starting corners did not ligate. It is our belief that the cause of the low levels of ligation is due to the design of the version 1 triangles and not an issue with the kinase enzyme itself. Kinase functions by phosphorylating 5' ends, however, in the version 1 design, after annealing the 5' is internal to the overhang (a 3' overhang) so the kinase reaction had to be performed on ssDNA prior to annealing. Any possible secondary structures in the ssDNA can inhibit the kinase reaction, decreasing its overall efficiency on that particular oligonucleotide.

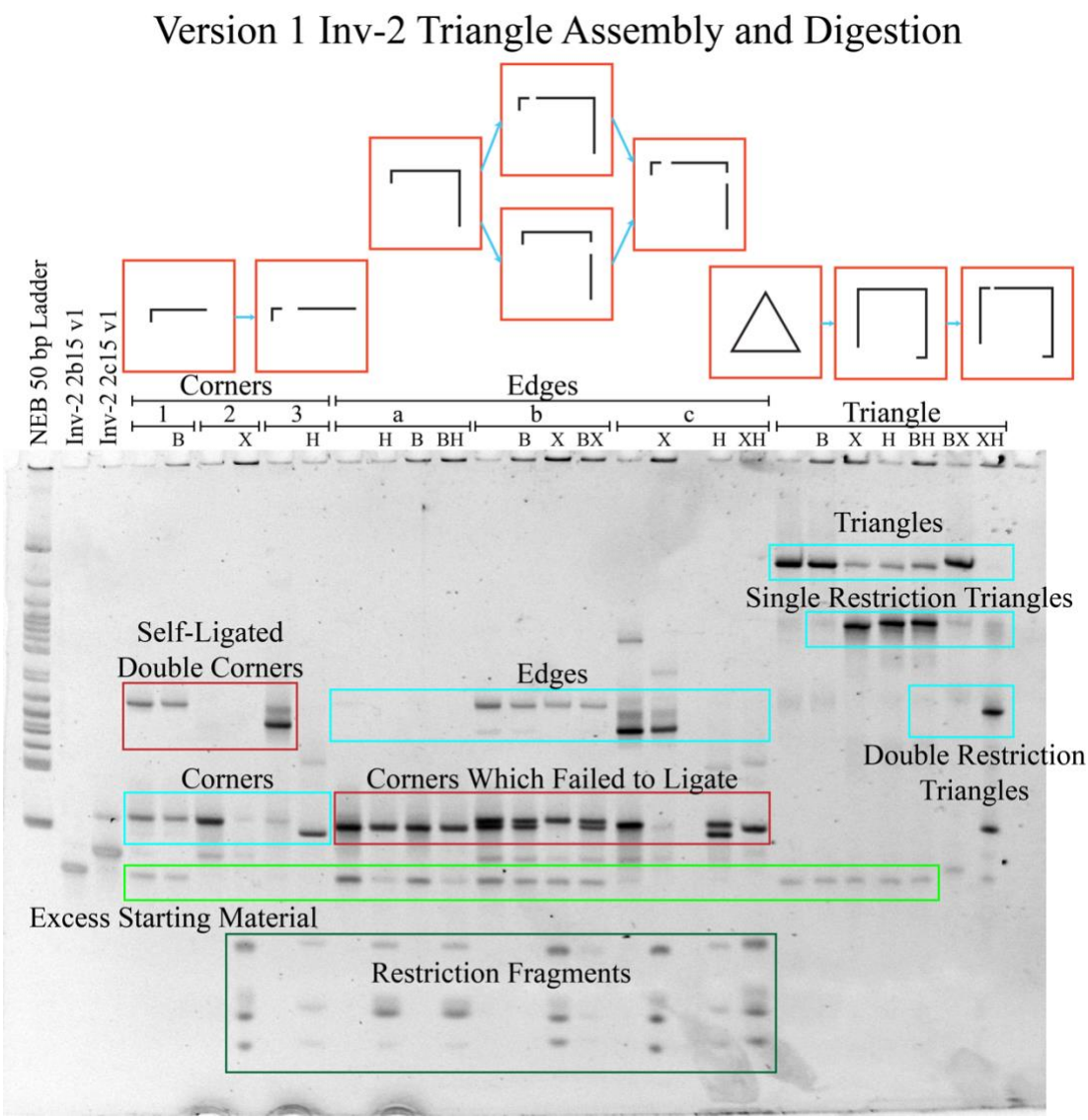

Figure A1.5 Version 1 dsDNA triangle test assembly gel: Assembly and digestion products from the Inv-2 binding DNA triangle version 1. Starting oligonucleotides, corners (sets of two annealed oligos), edges (two corners ligated to one another), and full triangles were each run on the gel with the relevant restriction digestion (B=BamHI, X=XbaI, and H=HindIII) to confirm proper annealing of each product. From this gel corners 1 and 3 can be seen forming double corners through self-annealing. Additionally, it is clear that the majority of sticky ends were unable to ligate even with the proper partner present. Upon further experimentation it was realized that the BamHI used on the products in this gel was dead and that is why no BamHI restrictions occurred.

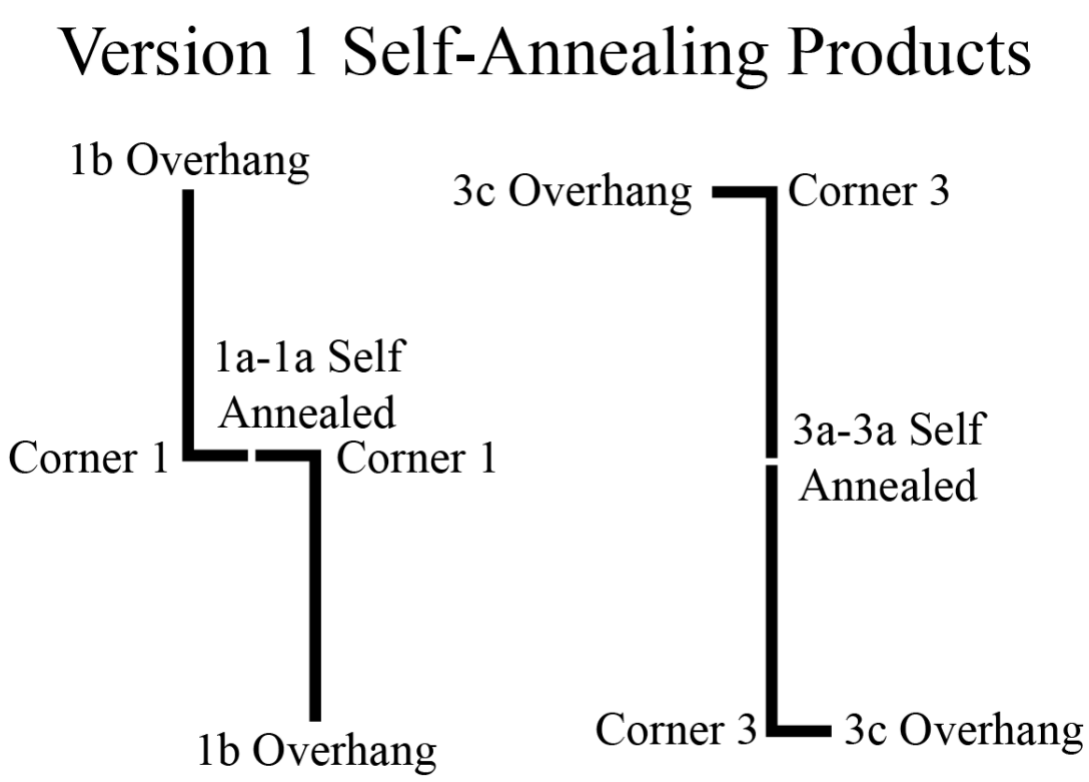

Figure A1.6 Version 1 dsDNA triangle self-annealing products: A diagram showing the self-annealed corners from 1a and 3a necessitating redesign.

Although the version 1 sequence clearly provided a reasonable yield of DNA triangles, we sought a synthesis that would be high yield enough to allow moving forward without requiring gel purification. Based on the issues seen in the version 1 triangles, instead of moving forward with the version 1 triangles, we decided on a redesign of the sequences. In the version 2 design the sticky ends for each annealing site have been changed to 5' overhangs instead of 3' overhangs. This was done in order to allow for the kinase reaction to be performed after annealing each corner. The a edge overhangs were redesigned entirely, and all overhangs were increased from 5 bp overhangs to 7 bp. A schematic of the version 2 design can be seen in

figure A1.7. The version 2 oligonucleotides were ordered from IDT, at which time the project was paused to continue work with the HTS project. Triangle work would later be resumed by Sindhu Muppala and Savannah Miller.

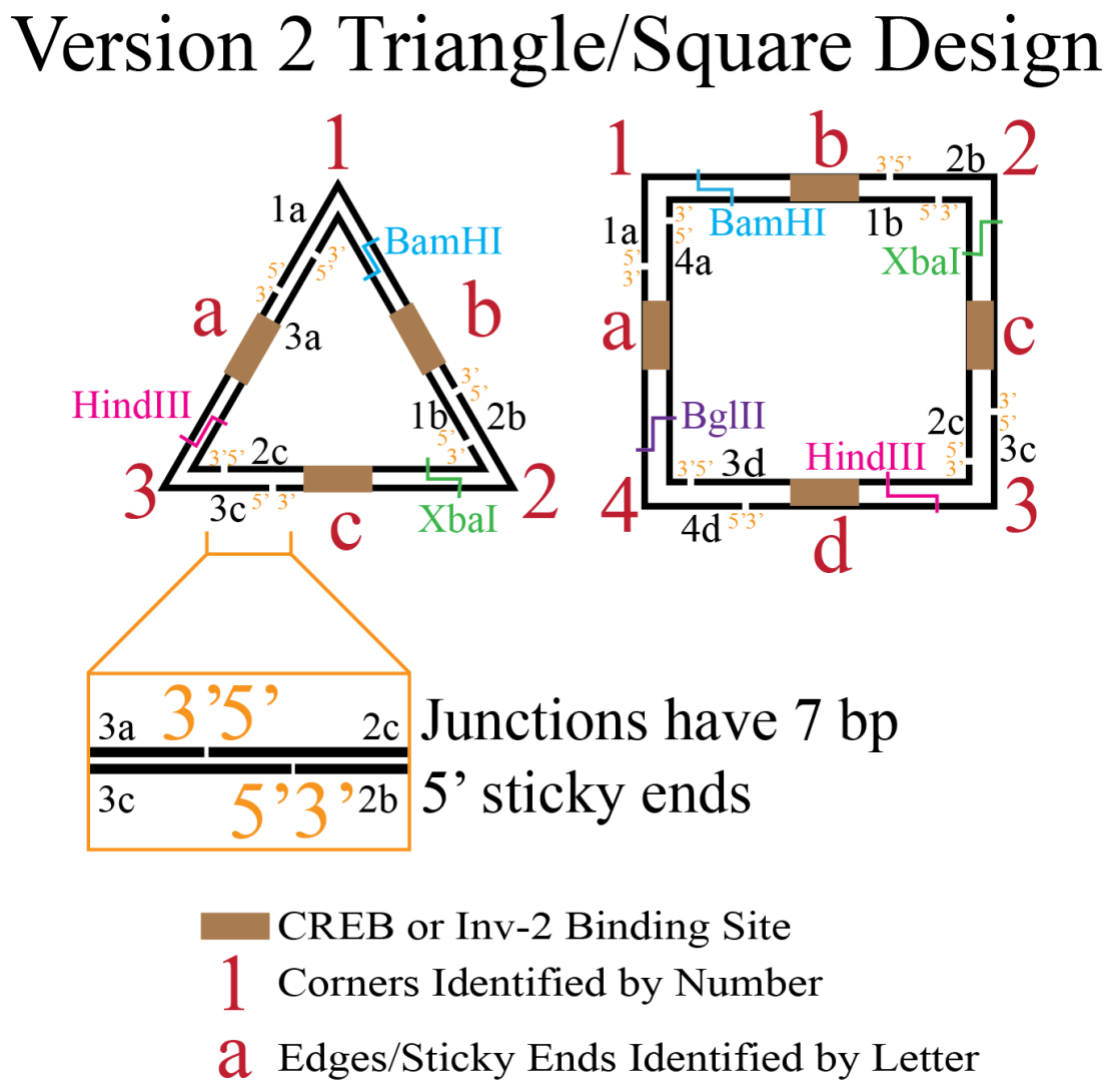

Figure A1.7 Version 2 dsDNA triangle design: A schematic showing the design of the version 2 triangles and squares. Version 2 uses 5' overhangs and 7 base pair overlap regions for each annealing site. In addition to these two changes, the sequences themselves at the overhang region were modified where necessary to prevent the previously observed self-annealing at the (a) edge.

Appendix 1.3: Materials and Methods

All enzymatic reagents were purchased from New England Biolabs, DNA oligonucleotides were synthesized by IDT, and all other reagents were purchased from Fisher Scientific.

Appendix 1.3.1: Kinase Reaction

Version 1 oligonucleotides were phosphorylated while single stranded using 10 units (NEB) T4 polynucleotide kinase in 1x NEB T4 Ligase Buffer at 37 °C for 1 hour. Each version 1 kinase reaction contained roughly 3 μg of oligonucleotide and a total volume of 20 μL.

Appendix 1.3.2: Corner Annealing Reactions

Corners were annealed by heating to 90 °C and immediately allowed to cool to room temperature in a total volume of 20 μL, with 40 mM NaCl in 1x TE.

Appendix 1.3.3: Ligation Reactions

Complete triangles were created by mixing an equimolar amount of each corner together in a 100 μL reaction containing 1x T4 Ligase Buffer and 400 units (NEB) T4 Ligase. The ligation reaction was allowed to proceed overnight at room temperature with remaining ligase being heat killed at 65 °C for 1 hour the following morning.

Appendix 1.3.4: DNA Triangle Oligonucleotide Sequences

Table A1.1 shows the oligonucleotide sequences for the version 1 Inv-2 and CREB triangles and squares. Table A1.2 shows the oligonucleotide sequences for the version 2 Inv-2 and CREB triangles and squares.

CREB Binding Site: ATGACGTCAT
Inv-2 Binding Site: GTCATATGAC

Version 1 Inv-2 Binding Site Oligonucleotides (5'-3')	
Inv-2 1a15 v1	ATTGGTCATATGACGGCCAGGATCCCGCCGACTTACTCCGA
Inv-2 1b15 v1	GTAAGTCGAATAAGCGGGATCCTGGCCGTCATATGACCAATACAGA
Inv-2 2b15 v1	ATTGGTCATATGACATTCCTCTAGACGCCGAGTCGTTCTGT
Inv-2 2c15 v1	ACGACTCGAATAAGCGTCTAGAGGAATGTCATATGACCAATGCGGT
Inv-2 3c15 v1	ATTGGTCATATGACAAGTTAAGCTTCCCCGATTCTCACCGC
Inv-2 3a15 v1	GAGAATCGAATAAGGGAAGCTTAACTTGTCATATGACCAATTCGGA
Inv-2 3d15 v1	GAGAATCGAATAAGGGAAGCTTAACTTGTCATATGACCAATGCTGA
Inv-2 4d15 v1	AATGGTCATATGACATGCTAGATCTCGCCGATTCTCTCAGC
Inv-2 4a15 v1	GAGAATCGAATAAGCGAGATCTAGCATGTCATATGACCATTTCGGA
Version 1 CREB Binding Site Oligonucleotides (5'-3')	
CREB 1a15 v1	ATTGATGACGTCATCACCAGGATCCCGCCGATGAGGTTAGC
CREB 1b15 v1	CCTCATCGAATAAGCGGGATCCTGGTGATGACGTCATCAATTGACT
CREB 2b15 v1	ATTGATGACGTCATTGCATTCTAGACGCCGAAGCTCAGTCA
CREB 2c15 v1	GAGCTTCGAATAAGCGTCTAGAATGCAATGACGTCATCAATCTGTA
CREB 3c15 v1	ATTGATGACGTCATGATGCAAGCTTCCCCGATCACGTACAG
CREB 3a15 v1	CGTGATCGAATAAGGGAAGCTTGCATCATGACGTCATCAATGCTAA
CREB 3d15 v1	CGTGATCGAATAAGGGAAGCTTGCATCATGACGTCATCAATACCCT
CREB 4d15 v1	ATTGATGACGTCATTAGCCAGATCTCGCCCTACAGGAGGGT
CREB 4a15 v1	CCTGTAGGAATAAGCGAGATCTGGCTAATGACGTCATCAATGCTAA

Table A1.1 Version 1 dsDNA triangle and square sequences: A table showing all version 1 Inv-2 and CREB triangle and square oligonucleotides.

Version 2 Inv-2 Binding Site Oligonucleotides (5'-3')	
INV 1a15 v2	TCGGAGTAAGTCGAATAAGCGGGATCCTGGCCGTCATATGACCAAT
INV 1b15 v2	GTTCTGTATTGGTCATATGACGGCCAGGATCCCGCCGACTT
INV 2b15 v2	ACAGAACGACTCGAATAAGCGTCTAGAGGAATGTCATATGACCAAT
INV 2c15 v2	TCACCGCATTGGTCATATGACATTCCTCTAGACGCCGAGTC
INV 3c15 v2	GCGGTGAGAATCGAATAAGGGAAGCTTAACTTGTCATATGACCAAT
INV 3a15 v2	ACTCCGAATTGGTCATATGACAAGTTAAGCTTCCCCGATTC
INV 3d15 v2	TCTCAGCATTGGTCATATGACAAGTTAAGCTTCCCCGATTC
INV 4d15 v2	GCTGAGAGAATCGAATAAGCGAGATCTAGCATGTCATATGACCATT
INV 4a15 v2	ACTCCGAAATGGTCATATGACATGCTAGATCTCGCCGATTC
Version 2 CREB Binding Site Oligonucleotides (5'-3')	
CREB 1a15 v2	GCTAACCTCATCGAATAAGCGGGATCCTGGTGATGACGTCATCAAT
CREB 1b15 v2	TCAGTCAATTGATGACGTCATCACCAGGATCCCGCCGATGA
CREB 2b15 v2	TGACTGAGCTTCGAATAAGCGTCTAGAATGCAATGACGTCATCAAT
CREB 2c15 v2	CGTACAGATTGATGACGTCATTGCATTCTAGACGCCGAAGC
CREB 3c15 v2	CTGTACGTGATCGAATAAGGGAAGCTTGCATCATGACGTCATCAAT
CREB 3a15 v2	GGTTAGCATTGATGACGTCATGATGCAAGCTTCCCCGATCA
CREB 3d15 v2	GGAGGGTATTGATGACGTCATGATGCAAGCTTCCCCGATCA
CREB 4d15 v2	ACCCTCCTGTAGGAATAAGCGAGATCTGGCTAATGACGTCATCAAT
CREB 4a15 v2	GGTTAGCATTGATGACGTCATTAGCCAGATCTCGCCCTACA

Table A1.2 Version 2 dsDNA triangle and square sequences: A table showing all version 2 Inv-2 and CREB triangle and square oligonucleotides.

Appendix 2: Histidine-lysine Peptides as Transfection Agents

Appendix 2.1: Overview

The transfection of nucleic acids into cells has become an area of particular interest for the treatment of numerous diseases. Through the use of exogenous DNA or small interfering RNA (siRNA), cellular processes may be manipulated[58,59]. In gene therapy, the introduction of DNA into cells allows for new gene products to be made that are not encoded for by the cell naturally or a gene may be introduced to supplement an existing gene product that is mutated or not expressed at the desired quantity in the host cell. siRNA can be introduced to a cell to selectively silence transcription via RNA interference[58,60,61].

Transfection is hard because nucleic acids are fragile and charged. Many different agents have been developed for non-viral packaging of exogenous nucleic acids for transfection, including liposomes, artificial peptides, and non-peptide polymers[59]. While liposomes and non-peptide polymers have shown promise as efficient transfection agents, they may be immunogenic or non-biodegradable, which can pose a risk of toxicity at the concentrations needed for effective transfection[62,63]. Artificially designed peptide-based transfection agents have the potential to effectively transfect nucleic acids into cells without being inherently toxic at the concentrations needed. To this end, modified, branched histidine-lysine (HK) peptides have been developed by Dr. A. James Mixson and others. A general structure of the HK peptides can be seen in Figure A2.1 where three core lysine residues are used as branch points for the peptide. These three core residues contain four branches: one is covalently

bound to the core via the backbone amine of the *N*-terminal core lysine, while the other three branches are covalently bound to the side chain amino groups of the core lysines.

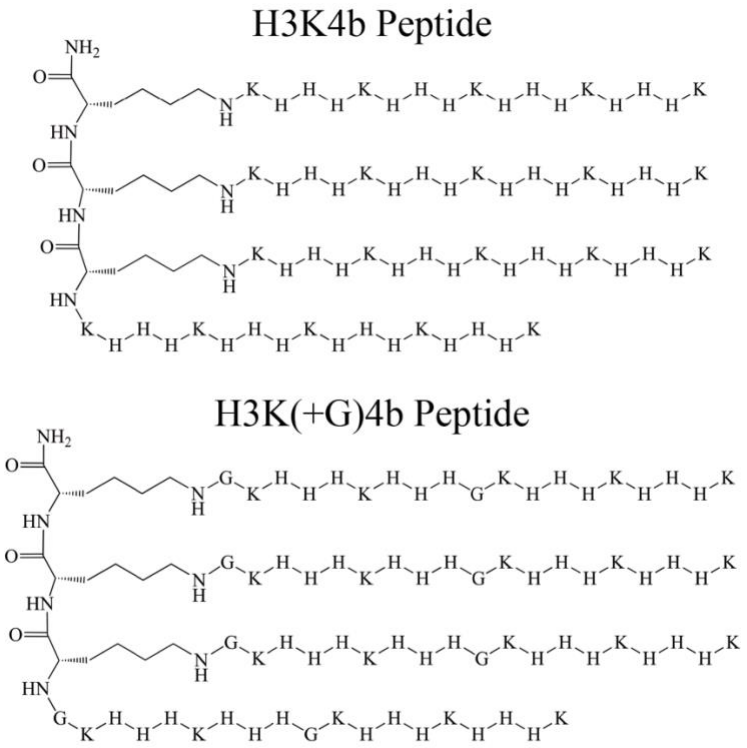

Figure A2.1 HK peptide schematic: A schematic showing the two HK peptides studied: H3K4b and H3K(+G)4b. The three lysine core of each polymer has four branches, one continuing off the N-terminus of the core and the other three attached to the amino group at the end of each core lysine. The +G variant of the peptide has two glycines inserted into each branch.

Appendix 2.2: Materials and Methods

All peptides, double stranded siRNA, and plasmid DNA used in HK work were provided by the Mixson laboratory. Preparation of peptides and siRNA can be found in previous publications by the Mixson lab[64]. Mixtures of DNA and peptide were formed using 32.9 nM plasmid DNA (provided by Mixson lab from a pCpG vector with the luciferase gene from a NcoI-NheI dual digestion of pMOD-Luc

inserted) with 60 μM peptide. Mixtures of siRNA and peptide were formed using 10 μM siRNA with 26.6 μM peptide. When preparing samples the peptide was always added to a solution with nucleic acid already present in order to minimize adsorption of peptide to the walls of sample tubes. Mixtures were allowed to incubate for 30 minutes prior to application onto freshly cleaved mica.

AFM scans were performed using an Agilent 5500 atomic force microscope. Peptide-nucleic acid complexes were applied to the surface and allowed to dry under ambient conditions. Samples dried to the mica were scanned in tapping mode using PPP-NCLR tips from NanoSensors. Force compression of peptide-nucleic acid complexes was performed using PPP-FM force modulation tips from NanoSensors. The force modulation tips all had spring constants between 2 and 4 N/m.

Appendix 2.3: Analysis of AFM on HK-nucleic acid Complexes

Complexes formed with the HK peptides, both H3K4b and H3K(+G)4b, were analyzed both with 21mer double stranded siRNA and plasmid DNA. In figure A2.2 complexes of a variety of sizes can be seen. Numerous small structures can be seen coalescing into larger structures, for both H3K4b and H3K(+G)4b peptides when complexed with DNA. As with the HK-DNA complexes, when the HK peptides are complexed with siRNA the size does vary; however, the observed larger complexes no longer appear to be aggregates of smaller particles but simply a larger version of uniformly distributed peptide and nucleic acid. These complexes are also observed flattening on the mica surface, making the complexes increase in diameter while decreasing in observed height above the surface, as compared to DNA complexes.

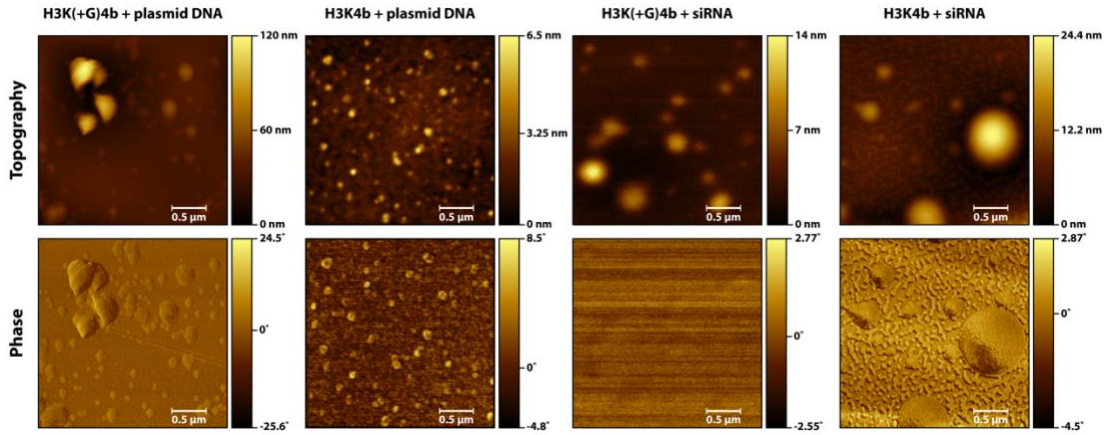

Figure A2.2 HK complexes with nucleic acids: A comparison of topographic and phase images of each HK peptide with either plasmid DNA or siRNA. The height to width ratio observed for the H3K4b and H3K(+G)4b peptides when in complex with plasmid DNA is roughly 1:1. However, with the siRNA complexes, the complex appears to be much wider than it is tall. We do not believe this to be a statement on the shape of the complex in solution, but an observation made only after application to a mica surface. AFM scans were processed using Gwyddion v2.53, background subtraction in Gwyddion was performed using 3 degrees of freedom in the polynomial background subtraction tool.

Beyond the differences in aggregation that have been observed, complexes composed of H3K(+G)4b and plasmid DNA show a uniform surface chemistry via phase imaging while all other complexes (H3K4b with siRNA or DNA and H3K(+G)4b with siRNA) present a non-uniform chemistry on the surface. It is not known whether the surface of the H3K(+G)4b complexes is mostly peptide or nucleic acid, although the contrast in phasing relative to the negative substrate surface would suggest that the former is more likely.

The variability observed in the degree to which the different complexes maintain height relative to diameter upon application to mica suggests that the physical modes of binding are quite different. In order to better understand these different modes of binding, work to quantitate the compressive and critical forces of individual

complexes was attempted. Due to the sizes of the complexes compression profiles could only be obtained for H3K(+G)4b complexes, both with siRNA and with DNA. H3K4b-DNA complexes were too small to accurately land a tip onto for force compression and H3K4b-siRNA complexes showed no difference from the mica itself. We believe this to be due to the fact that H3K4b-siRNA complexes are a very thin layer on top of the mica, so the force compression profile observed mimics that of mica itself. As would be expected, all H3K(+G)4b complexes have an observed compressive force smaller than that of the mica substrate, which can be seen in the initial slope of compression being lower than that of the mica substrate in figure A2.3.

However, unlike similar studies performed on protein crystals[65], the HK-nucleic acid complexes have a discernable critical force of compression giving a nearly horizontal region in the compression curves. This compressive force, observed as the slope between initial contact and the critical force, is -2.70 ± 0.05 nN/nm for H3K(+G)4b-DNA, -2.8 ± 0.5 nN/nm for H3K(+G)4b-siRNA, and -3.6 ± 0.2 nN/nm for the mica substrate. Both of the H3K(+G)4b complexes show a smaller compressive force than the mica, indicating that they are more compressible or softer than the substrate. After compression of these complexes has occurred, subsequent compression of the same location is no longer possible and significant hysteresis is observed in the initial approach-retract cycle. This indicates that the complexes themselves are non-elastic.

The critical forces for the DNA and siRNA complexes range from 70-100 nN and 30-50 nN respectively when compressing larger complexes. The increased critical

force for DNA complexes indicates that these complexes require the application of more mechanical energy before they experience deformation.

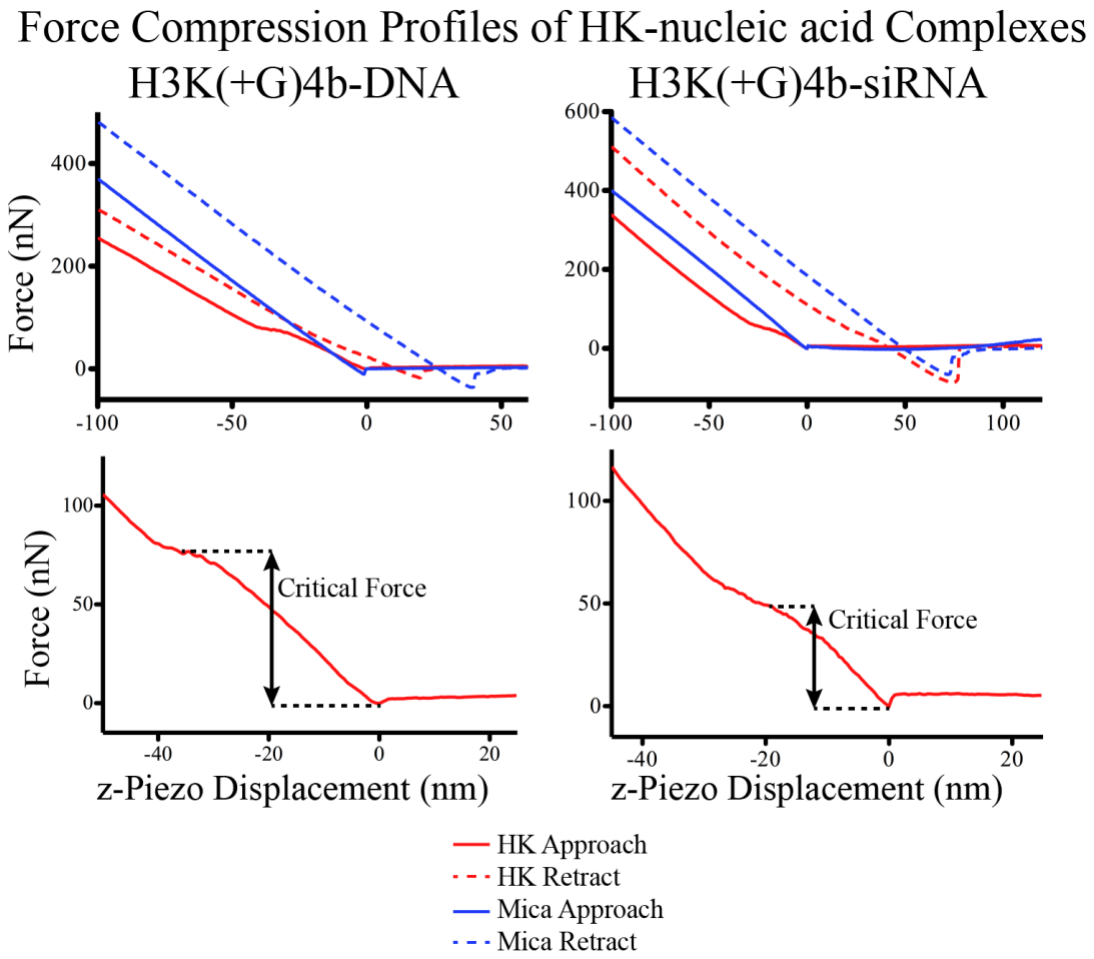

Figure A2.3 Force compression profiles of HK-nucleic acid complexes. H3K(+G)4b-DNA compression compared to mica on the left and H3K(+G)4b-siRNA compression compared to mica on the right. Sample compression is visible in red with mica compression in blue. The compression profile from approach is seen in a solid line with retraction of the tip after compression shown as a dashed line. The critical force for the curves on top is displayed in zoomed graphs on the bottom of the figure. Critical forces were observed to be -2.70 ± 0.05 nN/nm for H3K(+G)4b-DNA, -2.8 ± 0.5 nN/nm for H3K(+G)4b-siRNA, and -3.6 ± 0.2 nN/nm for the mica substrate.

Appendix 3: Improvements to LZD Protein Family

Appendix 3.1: Overview

As mentioned in chapter 5, the leucine zipper dual-binding proteins (LZD) contain an N-terminal CREB binding region and a C-terminal Inv-2 binding region. The proteins were designed by Dr. Daniel Gowetski and are assumed to be relatively rigid due to their coiled-coil α-helix structure[34]. The LZD family is comprised of four individual proteins: LZD66, LZD73, LZD80, and LZD87, each named based on the length in amino acids from binding site to binding site. The full amino acid sequence for each protein and all relevant DNA sequences for the LZD family are present in Appendix 4.

When Dr. Gowetski performed his DNA looping experiments, only LZD73 and LZD87 were generated. At this time the protein was expressed out of a pRSET-A vector in BL(21)-DE3-pLysS cells. The expression was prone to killing the cells and in order to obtain substantial quantities of protein the expression had to be done very slowly in large volumes.

While the protein itself was functional, the process of making it in large quantities that would be needed for crystallization was not feasible. In order to overcome this we decided to migrate the LZD family of proteins from the pRSET vector into pMAL-c2T (provided by Dr. Paul Paukstelis). The pMAL-c2T vector has a number of improvements over the former system. First, as an MBP-fusion the pre-purified protein is incredibly soluble and can be lysed at high cellular density via sonication. Second, we have observed no ill effects on cellular toxicity during

expression, allowing the growth time to be cut from 36 hours to just overnight. Third, this vector includes a TEV protease cleavage site immediately adjacent to the N-terminal binding region of the protein, which eliminates excess amino acids that were present from the pRSET expression system.

Appendix 3.2: Materials and Methods

Columns used in FPLC purification were purchased from GE Healthcare. All other reagents were purchased from Fischer Scientific. An AKTA FPLC was used for all column purification using the program in appendix 5.

In order to clone into the new vector, the current LZD sequences were removed from each individual pRSET backbone via restriction digestion with HindIII and BamHI. Restricted LZD DNA sequences were gel purified via 1% agarose gel in 1x TBE. The pMAL-c2T backbone was also prepared using the same dual restriction and the backbone was phosphatased to prevent reinsertion of the removed cloning site. The BamHI to HindIII fragment was mixed in a 1:1 ratio with the new backbone in 1x NEB T4 ligase buffer with T4 ligase. The cloned plasmids were introduced into DH-5α cells via electroporation and grown on LB-Amp plates at 37 °C overnight. Plasmids from colonies that grew were extracted using a Qiagen miniprep plasmid purification kit and confirmed by Sanger sequencing at GeneWiz.

Expression of pMAL-LZD73 was performed in BL21 cells at 37 °C with shaking overnight. The growth media for expression contained Autoinduction A and Autoinduction B (see appendix 6). Using this procedure the cells grow without expressing the protein until they have exhausted the glucose added to the solution, at

which point the cells transition to using the lactose and expression of the gene of interest is activated.

Cells containing MBP-LZD73 were centrifuged at 5,000x g for 30 minutes and resuspended with 20 mL maltose equilibration buffer (see appendix 6) per gram of cell paste with PMSF (1 mg/mL) added. The resuspended cells were lysed using a microfluidizer. Subsequent work on the proteins has transitioned to using sonication to lyse the cells.

Appendix 3.3: Results

Cellular lysate was loaded onto the AKTA FPLC with a GE MBP-Trap column attached using the protocol present in appendix 5. A sample chromatograph of LZD purification can be seen in figure A3.1.

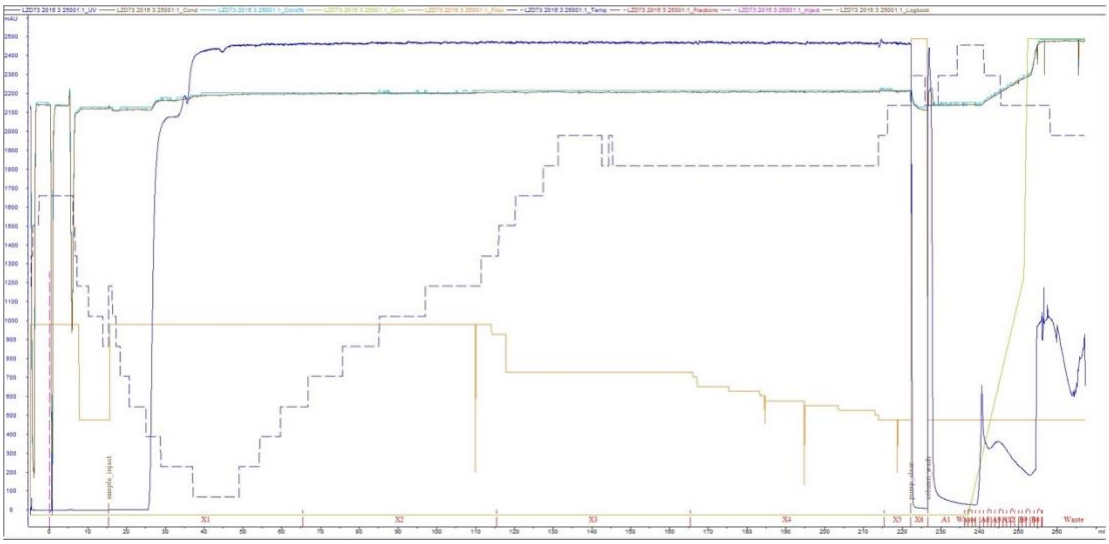

Figure A3.1 A chromatogram of MBP-LZD73 purification via FPLC.

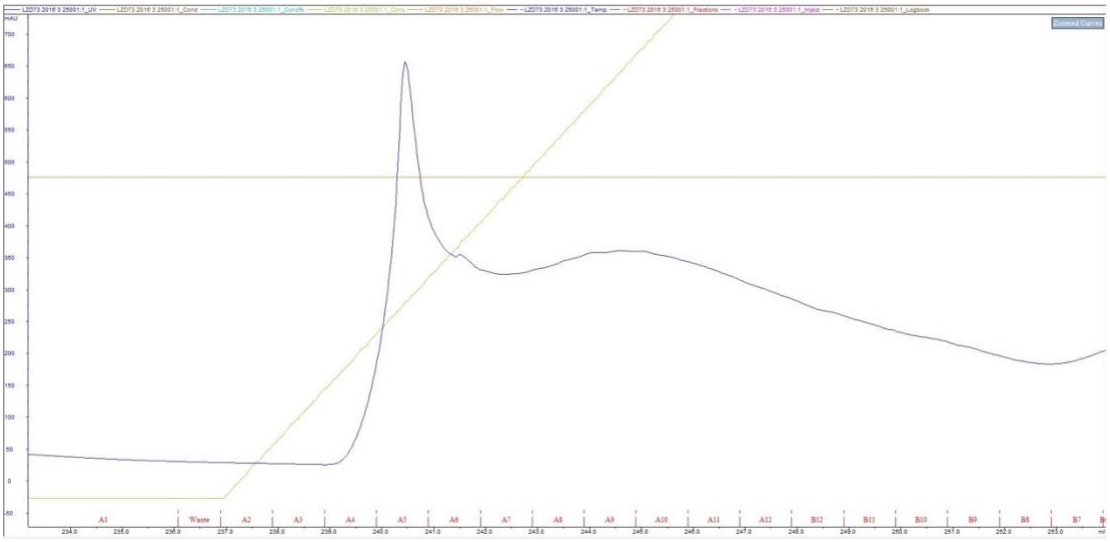

Figure A3.2 Zoomed portion of MBP-LZD73 purification: Selected portion of the chromatogram from figure A3.1 showing the elution of MBP-LZD73. Two peaks are observed as explained in figure A3.3.

Portions of the fractions where elution was observed were analyzed via SDS-PAGE and can be seen in figure A3.3. The initial elution peak is a smaller contaminant that co-eluted with the MBP-LZD73, however enough purified fractions eluted to allow for purification of large quantities of fusion protein from this workup.

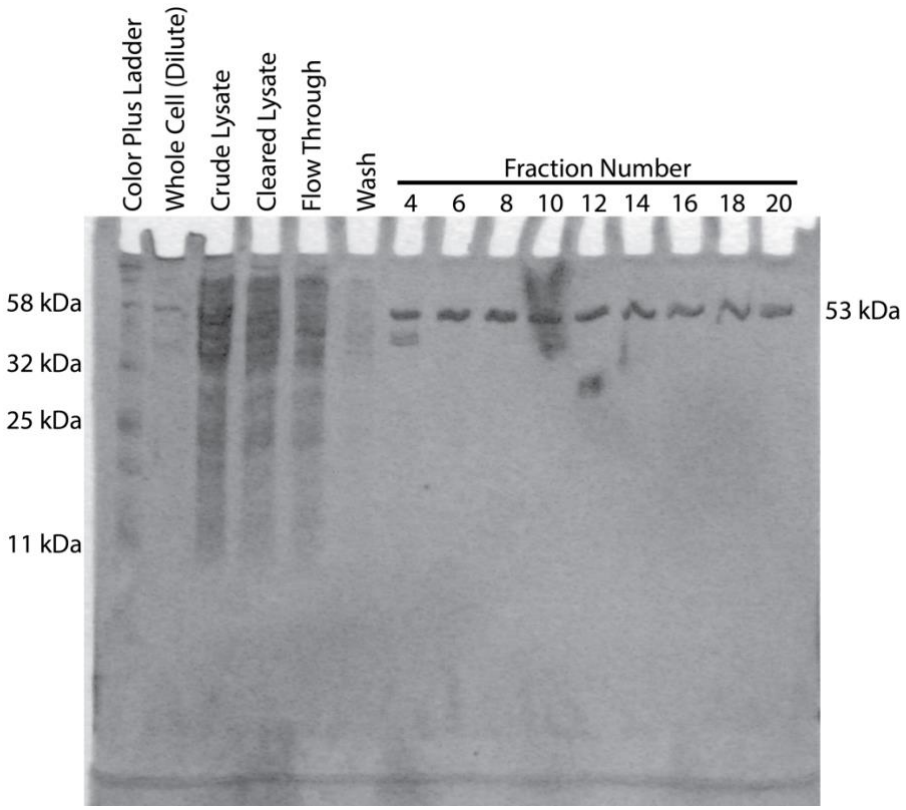

Figure A3.3 MBP-LZD73 purification verification via SDS-PAGE: SDS-PAGE showing elution fractions from the above FPLC purification of MBP-LZD73. The initial elution peak seen in fraction 4 is a co-elution of MBP-LZD73 and a second protein, after this elution purified MBP-LZD73 can be seen eluting across numerous fractions that were subsequently pooled and concentrated.

Appendix 4: Relevant DNA and Protein Sequences

Appendix 4.1: Library Construction Sequences (oligonucleotides and gBlocks)

All oligonucleotides are listed from 5'-3'. Modifications are specifically mentioned.

Appendix 4.1.1: Variable Sequence Synthesis Side Oligonucleotides

Variable sequence synthesis side oligonucleotides as ordered from IDT will be labelled as "VS##" and not "SV-S##" in frozen stocks.

SV-S30	ATGAGCGGTCACCCATTAGGTCCAGCAGGCATTGCAGTCAGACGCAGTTGAGGTGCTACGAGT
SV-S29	ATGAGCGGTCACCCATTAGCTCCAGCAGGCATTGCAGTCAGAGCAGTTGAGGTGCTACGAGT
SV-S28	ATGAGCGGTCACCCATTAGATCCAGCAGGCATTGCAGTCAGGCAGTTGAGGTGCTACGAGT
SV-S27	ATGAGCGGTCACCCATGAGTTCCAGCAGGCATTGCAGTCAGCAGTTGAGGTGCTACGAGT
SV-S26	ATGAGCGGTCACCCATGAGGTCCAGCAGGCATTGCAGTCGCAGTTGAGGTGCTACGAGT
SV-S25	ATGAGCGGTCACCCATGAGCTCCAGCAGGCATTGCAGTGCAGTTGAGGTGCTACGAGT
SV-S24	ATGAGCGGTCACCCATGAGATCCAGCAGGCATTGCAGGCAGTTGAGGTGCTACGAGT
SV-S23	ATGAGCGGTCACCCATCAGTTCCAGCAGGCATTGCAGCAGTTGAGGTGCTACGAGT
SV-S22	ATGAGCGGTCACCCATCAGGTCCAGCAGGCATTGCGCAGTTGAGGTGCTACGAGT
SV-S21	ATGAGCGGTCACCCATCAGCTCCAGCAGGCATTGGCAGTTGAGGTGCTACGAGT
SV-S20	ATGAGCGGTCACCCATCAGATCCAGCAGGCATTGCAGTTGAGGTGCTACGAGT
SV-S19	ATGAGCGGTCACCCATAAGTTCCAGCAGGCATGCAGTTGAGGTGCTACGAGT
SV-S18	ATGAGCGGTCACCCATAAGGTCCAGCAGGCAGCAGTTGAGGTGCTACGAGT
SV-S17	ATGAGCGGTCACCCATAAGCTCCAGCAGGCGCAGTTGAGGTGCTACGAGT
SV-S16	ATGAGCGGTCACCCATAAGATCCAGCAGGGCAGTTGAGGTGCTACGAGT
SV-S15	ATGAGCGGTCACCAATTAGTTCCAGCAGGCAGTTGAGGTGCTACGAGT
SV-S14	ATGAGCGGTCACCAATTAGGTCCAGCAGCAGTTGAGGTGCTACGAGT
SV-S13	ATGAGCGGTCACCAATTAGCTCCAGCGCAGTTGAGGTGCTACGAGT
SV-S12	ATGAGCGGTCACCAATTAGATCCAGGCAGTTGAGGTGCTACGAGT
SV-S11	ATGAGCGGTCACCAATGAGTTCCAGCAGTTGAGGTGCTACGAGT
SV-S10	ATGAGCGGTCACCAATGAGGTCCGCAGTTGAGGTGCTACGAGT
SV-S09	ATGAGCGGTCACCAATGAGCTCGCAGTTGAGGTGCTACGAGT
SV-S08	ATGAGCGGTCACCAATGAGATGCAGTTGAGGTGCTACGAGT
SV-S07	ATGAGCGGTCACCAATCAGTGCAGTTGAGGTGCTACGAGT
SV-S06	ATGAGCGGTCACCCATGAGGCAGTTGAGGTGCTACGAGT
SV-S05	ATGAGCGGTCACCCATCAGCAGTTGAGGTGCTACGAGT
SV-S04	ATGAGCGGTCACCCATAGCAGTTGAGGTGCTACGAGT

SV-S03	ATGAGCGGTCACCTATGCAGTTGAGGTGCTACGAGT
SV-S02	ATGAGCGGTCACCGAGCAGTTGAGGTGCTACGAGT
SV-S01	ATGAGCGGTCACCCGCAGTTGAGGTGCTACGAGT
SV-S00	ATGAGCGGTCACCGCAGTTGAGGTGCTACGAGT

Table A4.1 Sequence variant synthesis side oligonucleotides.

Appendix 4.1.2: Sequence Variant Cyclization Side Oligos

Sequence variant cyclization side oligos underwent a number of changes in nomenclature throughout development. Oligonucleotides as ordered from IDT will be labelled as "VC##" and not "SV-C##" in frozen stocks.

SV-C10	GCTATCTCGAGGTGTCGTGGGCTCGGAATGGTAGCTCTCGATGATGCAGTTGGACTA
SV-C09	GCTATCTCGAGGTGTCGTGGGCTCGGAATGGTACCTTCGATGATGCAGTTGGACTA
SV-C08	GCTATCTCGAGGTGTCGTGGGCTCGGAATGGTAACTCGATGATGCAGTTGGACTA
SV-C07	GCTATCTCGAGGTGTCGTGGGCTCGGAATGCTATTCGATGATGCAGTTGGACTA
SV-C06	GCTATCTCGAGGTGTCGTGGGCTCGGACTGGTATCGATGATGCAGTTGGACTA
SV-C05	GCTATCTCGAGGTGTCGTGGGCTCGGACTGCTTCGATGATGCAGTTGGACTA
SV-C04	GCTATCTCGAGGTGTCGTGGGCTCGGACTGATCGATGATGCAGTTGGACTA
SV-C03	GCTATCTCGAGGTGTCGTGGGCTCGGATGTTCGATGATGCAGTTGGACTA
SV-C02	GCTATCTCGAGGTGTCGTGGGCTCGGAGTTCGATGATGCAGTTGGACTA
SV-C01	GCTATCTCGAGGTGTCGTGGGCTCGGACTCGATGATGCAGTTGGACTA
SV-C00	GCTATCTCGAGGTGTCGTGGGCTCGGATCGATGATGCAGTTGGACTA

Table A4.2 Sequence variant cyclization side oligonucleotides

Appendix 4.1.3: Variable Sequence Side Splint Oligo

This oligonucleotide was ordered from IDT containing a 3'-3' linked T at the end. This inverted linked T prevented extension of the splint oligo.

```
GATGCAGTTGGACTAATGACGTCATACTCGTAGCACCTCAAC-/T/
```

Appendix 4.1.4: pJ4 gBlock

```
GTCGCGACGCGCGTTGGTGACCCCTGACTCCCCGTCGTGAATATTGAGATATACGAGGAGGGCTCGAG
GGTGATGAAGGCCTGTGATCACGGTGACCCAGCCGGAAGGCCGAGCGAATATTCGAAGTGGTCTGCAA
CTGTATCCGCCTCCGGGAGCTCCGGTTCCCACTCGAGCTGACACACACGTGCAGCTCCCGGTGACCTT
ATGGCAGCACTGCAATATTCTCTTACTGTCATGCCATCCGTAAGATGCTGCTCTTGCCCGGCGTCAAT
```

ACGGGATAATACCGCGCCACATAGCAGAACCTCGAGAGACGGTCGTCGACAGCTTGTCGGTCACCCTT
CGGGGCGAGCACTCTCAAGGATCTTACGATATC

Appendix 4.1.5: 47 class Oligonucleotides

These oligonucleotides were ordered 5'-phosphorylated from IDT.

5′-3′ BstEII Overhang	GTGACCCCTGACTCCCCGTCGTGAATATTGAGATATACGAGGAGGGC
5′-3′ XhoI Overhang	TCGAGCCCTCCTCGTATATCTCAATATTCACGACGGGGAGTCAGGG

Table A4.3 Synthesized 47 class phosphorylated sequences.

Appendix 4.1.6: Amplification Primers for Radiolabeling

VariableSequence-Nest-for	GTCAGAGATGAGCGGTCACC
VariableSequence-Nest-rev	ACTACTGGCTATCTCGAGGT
pJ4-Nest-for	CGTTGGTGACCCCTGACTCC
pJ4-Nest-rev	GACCGTCTCTCGAGGTTCTGCTATGTG

Table A4.4 Amplification primers for radiolabeling.

Appendix 4.2: Constructed Library Sequences

To construct all 1023 sequences: append 0-30 bp insert region from variable sequence library to the end of each size class sequence, then append each 0-10 bp insert region from variable sequence library to the end of each previously appended sequence.

47 class: CTCGAGCCCTCCTCGTATATCTCAATATTCACGACGGGGAGTCAGGGGTCACC
77 class: CTCGAGTGGGAACCGGAGCTCCCGGAGGCGGATACAGTTGCAGACCACTTCGAATATT
CGCTCGGCCTTCCGGCTGGGTCACC
107 class: CTCGAGGTTCTGCTATGTGGCGCGGTATTATCCCGTATTGACGCCGGGCAAGAGCAG
CATCTTACGGATGGCATGACAGTAAGAGAATATTGCAGTGCTGCCATAAGGTCACC

Region containing 0-30 bp insert in variable sequence library	
C30	CATTAGGTCCAGCAGGCATTGCAGTCAGACGCAGTTGAGGTGCTACGAGTATGACGTCAT
C29	CATTAGCTCCAGCAGGCATTGCAGTCAGAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C28	CATTAGATCCAGCAGGCATTGCAGTCAGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C27	CATGAGTTCCAGCAGGCATTGCAGTCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C26	CATGAGGTCCAGCAGGCATTGCAGTCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C25	CATGAGCTCCAGCAGGCATTGCAGTGCAGTTGAGGTGCTACGAGTATGACGTCAT
C24	CATGAGATCCAGCAGGCATTGCAGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C23	CATCAGTTCCAGCAGGCATTGCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C22	CATCAGGTCCAGCAGGCATTGCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C21	CATCAGCTCCAGCAGGCATTGGCAGTTGAGGTGCTACGAGTATGACGTCAT

C20	CATCAGATCCAGCAGGCATTGCAGTTGAGGTGCTACGAGTATGACGTCAT
C19	CATAAGTTCCAGCAGGCATGCAGTTGAGGTGCTACGAGTATGACGTCAT
C18	CATAAGGTCCAGCAGGCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C17	CATAAGCTCCAGCAGGCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C16	CATAAGATCCAGCAGGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C15	AATTAGTTCCAGCAGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C14	AATTAGGTCCAGCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C13	AATTAGCTCCAGCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C12	AATTAGATCCAGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C11	AATGAGTTCCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C10	AATGAGGTCCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C09	AATGAGCTCGCAGTTGAGGTGCTACGAGTATGACGTCAT
C08	AATGAGATGCAGTTGAGGTGCTACGAGTATGACGTCAT
C07	AATCAGTGCAGTTGAGGTGCTACGAGTATGACGTCAT
C06	CATGAGGCAGTTGAGGTGCTACGAGTATGACGTCAT
C05	CATCAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C04	CATAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C03	TATGCAGTTGAGGTGCTACGAGTATGACGTCAT
C02	GAGCAGTTGAGGTGCTACGAGTATGACGTCAT
C01	CGCAGTTGAGGTGCTACGAGTATGACGTCAT
C00	GCAGTTGAGGTGCTACGAGTATGACGTCAT

Region containing 0-10 bp insert in variable sequence library	
C--10	TAGTCCAACTGCATCATCGAGAGCTACCATTCCGAGCCCACGACACCTCGAG
C--09	TAGTCCAACTGCATCATCGAAGGTACCATTCCGAGCCCACGACACCTCGAG
C--08	TAGTCCAACTGCATCATCGAGTTACCATTCCGAGCCCACGACACCTCGAG
C--07	TAGTCCAACTGCATCATCGAATAGCATTCCGAGCCCACGACACCTCGAG
C--06	TAGTCCAACTGCATCATCGATACCAGTCCGAGCCCACGACACCTCGAG
C--05	TAGTCCAACTGCATCATCGAAGCAGTCCGAGCCCACGACACCTCGAG
C--04	TAGTCCAACTGCATCATCGATCAGTCCGAGCCCACGACACCTCGAG
C--03	TAGTCCAACTGCATCATCGAACATCCGAGCCCACGACACCTCGAG
C--02	TAGTCCAACTGCATCATCGAACTCCGAGCCCACGACACCTCGAG
C--01	TAGTCCAACTGCATCATCGAGTCCGAGCCCACGACACCTCGAG
C--00	TAGTCCAACTGCATCATCGATCCGAGCCCACGACACCTCGAG

Table A4.5 Regions containing 0-30 and 0-10 bp inserts needed for construction of full library.

A fully constructed sequence library that includes all 1023 sequences is available with the dissertation in the University of Maryland Digital Repository (DRUM) as an spreadsheet.

Appendix 4.3: Protein-DNA Looping Constructs

The following constructs are inside a pUC18 vector. The shown region is from the HindIII to EcoRI restriction sites. Insertion in pUC18 occurred through dual HindIII-EcoRI digestion and subsequent cloning. To assemble full plasmid sequences simply put the following sections inside pUC18 in place of the MCS.

The plasmid name indicates the internal loop size in the final construct (p130 will have a 130 bp internal loop).

Looping construct DNA amplification primers:

Forward Nesting Primer: GACTGACTATGAGCAAGCTT
Reverse Nesting Primer: GAACCAGTGACGGGTACCCT

Appendix 4.3.1: p130 Insert

```
TGAGCAAGCTTGGTGACCGAAGCTGAAGCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGTCTTGTTCCAAACTGGAACAACACTCAACCCTATCTCGCGTCATATGACCAAGCTGAATTCGCG
CGCTGACCTCGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTATC
CGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGTATTCAA
CATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAAC
GCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCA
ACAGCGGTAAGATCCTTGACACTCTTCTCCTACACGACGCTCTTCCGATCTCGGCACGGCACTCGAGG
GTACCCGTCAGATATC
```

Appendix 4.3.2: p160 Insert

```
TGAGCAAGCTTGGTGACCGGACAGGGACCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGCGTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAACCCTAT
CTCGCGTCATATGACCAAGCTGAATTCGCGCGCTGACCTCGGAAATGTGCGCGGAACCCCTATTTGTT
TATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAA
TATTGAAAAAGGAAGAGTATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATT
TTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTG
CACGAGTGGGTTACATCGAACTGGATCTCAACGCGGTAAGATCCTTGACACTCTTCTCCTACACGACG
CTCTTCCGATCTCGGCACGGCACTCGAGGGTACCCGTCAGATATC
```

Appendix 4.3.3: p190 Insert

TGAGCAAGCTTGGTGACCTAGAGCTAGACATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGCGCCCTGATAGACGGTTTTTCGCCCTTTGACGTTGGAGTCCACGTTCTTTAATAGTGGACTCTT
GTTCCAAACTGGAACAACACTCAACCCTATCTCGCGTCATATGACCAAGCTGAATTCGCGCGCTGACC
TCGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATG
AGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGTATTCAACATTTCCG
TGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGA
AAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGT
AAGATCCTTGACACTCTTCTCCTACACGACGCTCTTCCGATCTCGGCACGGCACTCGAGGGTACCCGT
CAGATATC

Appendix 4.3.4: p220 Insert

TGAGCAAGCTTGGTGACCTCTTGATCTTCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTTCGCCCTTTGACGTT
GGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAACCCTATCTCGCGTC
ATATGACCAAGCTGAATTCGCGCGCTGACCTCGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTC
TAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAA
AAGGAAGAGTATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTC
CTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTG
GGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCTTGACACTCTTCTCCTACACGACGCTCTTCC
GATCTCGGCACGGCACTCGAGGGTACCCGTCAGATATC

Appendix 4.3.5: p250 Insert

TGAGCAAGCTTGGTGACCTTCAGCTTCACATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGCTTTACGGCACCTCGACCCCAAAAAACTTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCC
CTGATAGACGGTTTTTCGCCCTTTGACGTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAA
CTGGAACAACACTCAACCCTATCTCGCGTCATATGACCAAGCTGAATTCGCGCGCTGACCTCGGAAAT
GTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATA
ACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGTATTCAACATTTCCGTGTCGCCC
TTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAA
GATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCT
TGACACTCTTCTCCTACACGACGCTCTTCCGATCTCGGCACGGCACTCGAGGGTACCCGTCAGATATC

Appendix 4.3.6: p280 Insert

TGAGCAAGCTTGGTGACCCACTCCACTCCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGGGGGGCTCCCTTTAGGGTTCCGATTTAGTGCTTTACGGCACCTCGACCCCAAAAAACTTGATTA
GGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTTCGCCCTTTGACGTTGGAGTCCA
CGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAACCCTATCTCGCGTCATATGACC
AAGCTGAATTCGCGCGCTGACCTCGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACA
TTCAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGA
GTATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTT
GCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACAT
CGAACTGGATCTCAACAGCGGTAAGATCCTTGACACTCTTCTCCTACACGACGCTCTTCCGATCTCGG
CACGGCACTCGAGGGTACCCGTCAGATATC

Appendix 4.3.7: p310 Insert

TGAGCAAGCTTGGTGACCCATTACATTACATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGTCGCCGGCTTTCCCCGTCAAGCTCTAAATCGGGGGCTCCCTTTAGGGTTCCGATTTAGTGCTTT
ACGGCACCTCGACCCCAAAAAACTTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGA
CGGTTTTTCGCCCTTTGACGTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACA
ACACTCAACCCTATCTCGCGTCATATGACCAAGCTGAATTCGCGCGCTGACCTCGGAAATGTGCGCGG
AACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGAT
AAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCC
TTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGA
AGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCTTGACACTC
TTCTCCTACACGACGCTCTTCCGATCTCGGCACGGCACTCGAGGGTACCCGTCAGATATC

Appendix 4.3.8: p340 Insert

TGAGCAAGCTTGGTGACCCCCGTCCCGTCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGTCGCTTTCTTCCCTTCCTTTCTCGCCACGTTCGCCGGCTTTCCCCGTCAAGCTCTAAATCGGGG
GCTCCCTTTAGGGTTCCGATTTAGTGCTTTACGGCACCTCGACCCCAAAAAACTTGATTAGGGTGATG
GTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTTCGCCCTTTGACGTTGGAGTCCACGTTCTTT
AATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAACCCTATCTCGCGTCATATGACCAAGCTGAA
TTCGCGCGCTGACCTCGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATA
TGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGT
ATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCC
AGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGG
ATCTCAACAGCGGTAAGATCCTTGACACTCTTCTCCTACACGACGCTCTTCCGATCTCGGCACGGCAC
TCGAGGGTACCCGTCAGATATC

Appendix 4.3.9: p370 Insert

TGAGCAAGCTTGGTGACCCCTGGCCTGGCATGTCGGACTGTAGAACTCTGAACCTGTCGGAATATTCA
GCTGCACTTGCCAGCGCCCTATCGCCCGCTCCTTTCGCTTTCTTCCCTTCCTTTCTCGCCACGTTCGC
CGGCTTTCCCCGTCAAGCTCTAAATCGGGGGCTCCCTTTAGGGTTCCGATTTAGTGCTTTACGGCACC
TCGACCCCAAAAAACTTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTT
CGCCCTTTGACGTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAA
CCCTATCTCGCGTCATATGACCAAGCTGAATTCGCGCGCTGACCTCGGAAATGTGCGCGGAACCCCTA
TTTGTTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTT
CAATAATATTGAAAAAGGAAGAGTATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGC
GGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGT
TGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCTTGACACTCTTCTCCTA
CACGACGCTCTTCCGATCTCGGCACGGCACTCGAGGGTACCCGTCAGATATC

Appendix 4.4: LZD DNA and Protein Sequences

Appendix 4.4.1: pMAL-c2T full plasmid sequence

Each full plasmid can be generated by inserting ORF from subsequent sections into the labelled region in this plasmid.

CCGACACCATCGAATGGTGCAAAACCTTTCGCGGTATGGCATGATAGCGCCCGGAAGAGAGTCAATTC
AGGGTGGTGAATGTGAAACCAGTAACGTTATACGATGTCGCAGAGTATGCCGGTGTCTCTTATCAGAC

CGTTTCCCGCGTGGTGAACCAGGCCAGCCACGTTTCTGCGAAAACGCGGGAAAAAGTGGAAGCGGCGA
TGGCGGAGCTGAATTACATTCCCAACCGCGTGGCACAACAACTGGCGGGCAAACAGTCGTTGCTGATT
GGCGTTGCCACCTCCAGTCTGGCCCTGCACGCGCCGTCGCAAATTGTCGCGGCGATTAAATCTCGCGC
CGATCAACTGGGTGCCAGCGTGGTGGTGTCGATGGTAGAACGAAGCGGCGTCGAAGCCTGTAAAGCGG
CGGTGCACAATCTTCTCGCGCAACGCGTCAGTGGGCTGATCATTAACTATCCGCTGGATGACCAGGAT
GCCATTGCTGTGGAAGCTGCCTGCACTAATGTTCCGGCGTTATTTCTTGATGTCTCTGACCAGACACC
CATCAACAGTATTATTTTCTCCCATGAAGACGGTACGCGACTGGGCGTGGAGCATCTGGTCGCATTGG
GTCACCAGCAAATCGCGCTGTTAGCGGGCCCATTAAGTTCTGTCTCGGCGCGTCTGCGTCTGGCTGGC
TGGCATAAATATCTCACTCGCAATCAAATTCAGCCGATAGCGGAACGGGAAGGCGACTGGAGTGCCAT
GTCCGGTTTTCAACAAACCATGCAAATGCTGAATGAGGGCATCGTTCCCACTGCGATGCTGGTTGCCA
ACGATCAGATGGCGCTGGGCGCAATGCGCGCCATTACCGAGTCCGGGCTGCGCGTTGGTGCGGATATC
TCGGTAGTGGGATACGACGATACCGAAGACAGCTCATGTTATATCCCGCCGTTAACCACCATCAAACA
GGATTTTCGCCTGCTGGGGCAAACCAGCGTGGACCGCTTGCTGCAACTCTCTCAGGGCCAGGCGGTGA
AGGGCAATCAGCTGTTGCCCGTCTCACTGGTGAAAAGAAAAACCACCCTGGCGCCCAATACGCAAACC
GCCTCTCCCCGCGCGTTGGCCGATTCATTAATGCAGCTGGCACGACAGGTTTCCCGACTGGAAAGCGG
GCAGTGAGCGCAACGCAATTAATGTAAGTTAGCTCACTCATTAGGCACAATTCTCATGTTTGACAGCT
TATCATCGACTGCACGGTGCACCAATGCTTCTGGCGTCAGGCAGCCATCGGAAGCTGTGGTATGGCTG
TGCAGGTCGTAAATCACTGCATAATTCGTGTCGCTCAAGGCGCACTCCCGTTCTGGATAATGTTTTTT
GCGCCGACATCATAACGGTTCTGGCAAATATTCTGAAATGAGCTGTTGACAATTAATCATCGGCTCGT
ATAATGTGTGGAATTGTGAGCGGATAACAATTTCACACAGGAAACAGCCAGTCCGTTTAGGTGTTTTC
ACGAGCACTTCACCAACAAGGACCATAGCATATGAAAATCGAAGAAGGTAAACTGGTAATCTGGATTA
ACGGCGATAAAGGCTATAACGGTCTCGCTGAAGTCGGTAAGAAATTCGAGAAAGATACCGGAATTAAA
GTCACCGTTGAGCATCCGGATAAACTGGAAGAGAAATTCCCACAGGTTGCGGCAACTGGCGATGGCCC
TGACATTATCTTCTGGGCACACGACCGCTTTGGTGGCTACGCTCAATCTGGCCTGTTGGCTGAAATCA
CCCCGGACAAAGCGTTCCAGGACAAGCTGTATCCGTTTACCTGGGATGCCGTACGTTACAACGGCAAG
CTGATTGCTTACCCGATCGCTGTTGAAGCGTTATCGCTGATTTATAACAAAGATCTGCTGCCGAACCC
GCCAAAAACCTGGGAAGAGATCCCGGCGCTGGATAAAGAACTGAAAGCGAAAGGTAAGAGCGCGCTGA
TGTTCAACCTGCAAGAACCGTACTTCACCTGGCCGCTGATTGCTGCTGACGGGGGTTATGCGTTCAAG
TATGAAAACGGCAAGTACGACATTAAAGACGTGGGCGTGGATAACGCTGGCGCGAAAGCGGGTCTGAC
CTTCCTGGTTGACCTGATTAAAAACAAACACATGAATGCAGACACCGATTACTCCATCGCAGAAGCTG
CCTTTAATAAAGGCGAAACAGCGATGACCATCAACGGCCCGTGGGCATGGTCCAACATCGACACCAGC
AAAGTGAATTATGGTGTAACGGTACTGCCGACCTTCAAGGGTCAACCATCCAAACCGTTCGTTGGCGT
GCTGAGCGCAGGTATTAACGCCGCCAGTCCGAACAAAGAGCTGGCAAAAGAGTTCCTCGAAAACTATC
TGCTGACTGATGAAGGTCTGGAAGCGGTTAATAAAGACAAACCGCTGGGTGCCGTAGCGCTGAAGTCT
TACGAGGAAGAGTTGGCGAAAGATCCACGTATTGCCGCCACTATGGAAAACGCCCAGAAAGGTGAAAT
CATGCCGAACATCCCGCAGATGTCCGCTTTCTGGTATGCCGTGCGTACTGCGGTGATCAACGCCGCCA
GCGGTCGTCAGACTGTCGATGAAGCCCTGAAAGACGCGCAGACTAATTCGAGCTCGAACAACAACAAC
AATAACAATAACAACAACCTCGGGGAAAACCTGTATTTTCAGG**<ORF_INSERT>**AAGCTTGGCACTG
GCCGTCGTTTTACAACGTCGTGACTGGGAAAACCCTGGCGTTACCCAACTTAATCGCCTTGCAGCACA
TCCCCCTTTCGCCAGCTGGCGTAATAGCGAAGAGGCCCGCACCGATCGCCCTTCCCAACAGTTGCGCA
GCCTGAATGGCGAATGGCAGCTTGGCTGTTTTGGCGGATGAGATAAGATTTTCAGCCTGATACAGATT
AAATCAGAACGCAGAAGCGGTCTGATAAAACAGAATTTGCCTGGCGGCAGTAGCGCGGTGGTCCCACC
TGACCCCATGCCGAACTCAGAAGTGAAACGCCGTAGCGCCGATGGTAGTGTGGGGTCTCCCCATGCGA
GAGTAGGGAACTGCCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTCGTTTTAT
CTGTTGTTTGTCGGTGAACGCTCTCCTGAGTAGGACAAATCCGCCGGGAGCGGATTTGAACGTTGCGA
AGCAACGGCCCGGAGGGTGGCGGGCAGGACGCCCGCCATAAACTGCCAGGCATCAAATTAAGCAGAAG
GCCATCCTGACGGATGGCCTTTTTGCGTTTCTACAAACTCTTTTTGTTTATTTTTCTAAATACATTCA
AATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTAT
GAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTC
ACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAA
CTGGATCTCAACAGCGGTAAGATCCTTGAGAGTTTTCGCCCCGAAGAACGTTCTCCAATGATGAGCAC
TTTTAAAGTTCTGCTATGTGGCGCGGTATTATCCCGTGTTGACGCCGGGCAAGAGCAACTCGGTCGCC
GCATACACTATTCTCAGAATGACTTGGTTGAGTACTCACCAGTCACAGAAAAGCATCTTACGGATGGC
ATGACAGTAAGAGAATTATGCAGTGCTGCCATAACCATGAGTGATAACACTGCGGCCAACTTACTTCT
GACAACGATCGGAGGACCGAAGGAGCTAACCGCTTTTTTGCACAACATGGGGGATCATGTAACTCGCC
TTGATCGTTGGGAACCGGAGCTGAATGAAGCCATACCAAACGACGAGCGTGACACCACGATGCCTGTA

GCAATGGCAACAACGTTGCGCAAACTATTAACTGGCGAACTACTTACTCTAGCTTCCCGGCAACAATT
AATAGACTGGATGGAGGCGGATAAAGTTGCAGGACCACTTCTGCGCTCGGCCCTTCCGGCTGGCTGGT
TTATTGCTGATAAATCTGGAGCCGGTGAGCGTGGGTCTCGCGGTATCATTGCAGCACTGGGGCCAGAT
GGTAAGCCCTCCCGTATCGTAGTTATCTACACGACGGGGAGTCAGGCAACTATGGATGAACGAAATAG
ACAGATCGCTGAGATAGGTGCCTCACTGATTAAGCATTGGTAACTGTCAGACCAAGTTTACTCATATA
TACTTTAGATTGATTTACCCCGGTTGATAATCAGAAAAGCCCCAAAAACAGGAAGATTGTATAAGCAA
ATATTTAAATTGTAAACGTTAATATTTTGTTAAAATTCGCGTTAAATTTTTGTTAAATCAGCTCATTT
TTTAACCAATAGGCCGAAATCGGCAAAATCCCTTATAAATCAAAAGAATAGACCGAGATAGGGTTGAG
TGTTGTTCCAGTTTGGAACAAGAGTCCACTATTAAAGAACGTGGACTCCAACGTCAAAGGGCGAAAAA
CCGTCTATCAGGGCGATGGCCCACTACGTGAACCATCACCCAAATCAAGTTTTTTGGGGTCGAGGTGC
CGTAAAGCACTAAATCGGAACCCTAAAGGGAGCCCCCGATTTAGAGCTTGACGGGGAAAGCCGGCGAA
CGTGGCGAGAAAGGAAGGGAAGAAAGCGAAAGGAGCGGGCGCTAGGGCGCTGGCAAGTGTAGCGGTCA
CGCTGCGCGTAACCACCACACCCGCCGCGCTTAATGCGCCGCTACAGGGCGCGTAAAAGGATCTAGGT
GAAGATCCTTTTTGATAATCTCATGACCAAAATCCCTTAACGTGAGTTTTCGTTCCACTGAGCGTCAG
ACCCCGTAGAAAAGATCAAAGGATCTTCTTGAGATCCTTTTTTTCTGCGCGTAATCTGCTGCTTGCAA
ACAAAAAAACCACCGCTACCAGCGGTGGTTTGTTTGCCGGATCAAGAGCTACCAACTCTTTTTCCGAA
GGTAACTGGCTTCAGCAGAGCGCAGATACCAAATACTGTCCTTCTAGTGTAGCCGTAGTTAGGCCACC
ACTTCAAGAACTCTGTAGCACCGCCTACATACCTCGCTCTGCTAATCCTGTTACCAGTGGCTGCTGCC
AGTGGCGATAAGTCGTGTCTTACCGGGTTGGACTCAAGACGATAGTTACCGGATAAGGCGCAGCGGTC
GGGCTGAACGGGGGGTTCGTGCACACAGCCCAGCTTGGAGCGAACGACCTACACCGAACTGAGATACC
TACAGCGTGAGCTATGAGAAAGCGCCACGCTTCCCGAAGGGAGAAAGGCGGACAGGTATCCGGTAAGC
GGCAGGGTCGGAACAGGAGAGCGCACGAGGGAGCTTCCAGGGGGAAACGCCTGGTATCTTTATAGTCC
TGTCGGGTTTCGCCACCTCTGACTTGAGCGTCGATTTTTGTGATGCTCGTCAGGGGGGCGGAGCCTAT
GGAAAAACGCCAGCAACGCGGCCTTTTTACGGTTCCTGGCCTTTTGCTGGCCTTTTGCTCACATGTTC
TTTCCTGCGTTATCCCCTGATTCTGTGGATAACCGTATTACCGCCTTTGAGTGAGCTGATACCGCTCG
CCGCAGCCGAACGACCGAGCGCAGCGAGTCAGTGAGCGAGGAAGCGGAAGAGCGCCTGATGCGGTATT
TTCTCCTTACGCATCTGTGCGGTATTTCACACCGCATATATGGTGCACTCTCAGTACAATCTGCTCTG
ATGCCGCATAGTTAAGCCAGTATACACTCCGCTATCGCTACGTGACTGGGTCATGGCTGCGCCCCGAC
ACCCGCCAACACCCGCTGACGCGCCCTGACGGGCTTGTCTGCTCCCGGCATCCGCTTACAGACAAGCT
GTGACCGTCTCCGGGAGCTGCATGTGTCAGAGGTTTTCACCGTCATCACCGAAACGCGCGAGGCAGCT
GCGGTAAAGCTCATCAGCGTGGTCGTGCAGCGATTCACAGATGTCTGCCTGTTCATCCGCGTCCAGCT
CGTTGAGTTTCTCCAGAAGCGTTAATGTCTGGCTTCTGATAAAGCGGGCCATGTTAAGGGCGGTTTTT
TCCTGTTTGGTCACTGATGCCTCCGTGTAAGGGGGATTTCTGTTCATGGGGGTAATGATACCGATGAA
ACGAGAGAGGATGCTCACGATACGGGTTACTGATGATGAACATGCCCGGTTACTGGAACGTTGTGAGG
GTAAACAACTGGCGGTATGGATGCGGCGGGACCAGAGAAAAATCACTCAGGGTCAATGCCAGCGCTTC
GTTAATACAGATGTAGGTGTTCCACAGGGTAGCCAGCAGCATCCTGCGATGCAGATCCGGAACATAAT
GGTGCAGGGCGCTGACTTCCGCGTTTCCAGACTTTACGAAACACGGAAACCGAAGACCATTCATGTTG
TTGCTCAGGTCGCAGACGTTTTGCAGCAGCAGTCGCTTCACGTTCGCTCGCGTATCGGTGATTCATTC
TGCTAACCAGTAAGGCAACCCCGCCAGCCTAGCCGGGTCCTCAACGACAGGAGCACGATCATGCGCAC
CCGTGGCCAGGACCCAACGCTGCCCGAAATT

Appendix 4.4.2: pMAL-LZD66 ORF

ATGAAAATCGAAGAAGGTAAACTGGTAATCTGGATTAACGGCGATAAAGGCTATAACGGTCTCGCTGA
AGTCGGTAAGAAATTCGAGAAAGATACCGGAATTAAAGTCACCGTTGAGCATCCGGATAAACTGGAAG
AGAAATTCCCACAGGTTGCGGCAACTGGCGATGGCCCTGACATTATCTTCTGGGCACACGACCGCTTT
GGTGGCTACGCTCAATCTGGCCTGTTGGCTGAAATCACCCCGGACAAAGCGTTCCAGGACAAGCTGTA
TCCGTTTACCTGGGATGCCGTACGTTACAACGGCAAGCTGATTGCTTACCCGATCGCTGTTGAAGCGT
TATCGCTGATTTATAACAAAGATCTGCTGCCGAACCCGCCAAAAACCTGGGAAGAGATCCCGGCGCTG
GATAAAGAACTGAAAGCGAAAGGTAAGAGCGCGCTGATGTTCAACCTGCAAGAACCGTACTTCACCTG
GCCGCTGATTGCTGCTGACGGGGGTTATGCGTTCAAGTATGAAAACGGCAAGTACGACATTAAAGACG
TGGGCGTGGATAACGCTGGCGCGAAAGCGGGTCTGACCTTCCTGGTTGACCTGATTAAAAACAAACAC
ATGAATGCAGACACCGATTACTCCATCGCAGAAGCTGCCTTTAATAAAGGCGAAACAGCGATGACCAT
CAACGGCCCGTGGGCATGGTCCAACATCGACACCAGCAAAGTGAATTATGGTGTAACGGTACTGCCGA
CCTTCAAGGGTCAACCATCCAAACCGTTCGTTGGCGTGCTGAGCGCAGGTATTAACGCCGCCAGTCCG

AACAAAGAGCTGGCAAAAGAGTTCCTCGAAAACTATCTGCTGACTGATGAAGGTCTGGAAGCGGTTAA
TAAAGACAAACCGCTGGGTGCCGTAGCGCTGAAGTCTTACGAGGAAGAGTTGGCGAAAGATCCACGTA
TTGCCGCCACTATGGAAAACGCCCAGAAAGGTGAAATCATGCCGAACATCCCGCAGATGTCCGCTTTC
TGGTATGCCGTGCGTACTGCGGTGATCAACGCCGCCAGCGGTCGTCAGACTGTCGATGAAGCCCTGAA
AGACGCGCAGACTAATTCGAGCTCGAACAACAACAACAATAACAATAACAACAACCTCGGGGAAAACC
TGTATTTTCAGGGATCCGATCCAGCTGCTCTGAAGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGC
CGAGCTCGGAAGCTGCAACGAATGAAGCAGCTGGAAGACAAGGTGTACCACCTGGAGAACGAAGTTGC
GCGCCTGAAGAAGCTGGTGGGTGAACTGCAGAAGTTACAGCGGGTGAAGCGAGCTCGGAACACTGAAG
CTGCTCGACGGAGCCGAGCTCGAAAGGCTGCTCTGAAGGGATAGTAAGAATTCG

Appendix 4.4.3: pMAL-LZD66 Protein Sequence

MKIEEGKLVIWINGDKGYNGLAEVGKKFEKDTGIKVTVEHPDKLEEKFPQVAATGDGPDIIFWAHDRF
GGYAQSGLLAEITPDKAFQDKLYPFTWDAVRYNGKLIAYPIAVEALSLIYNKDLLPNPPKTWEEIPAL
DKELKAKGKSALMFNLQEPYFTWPLIAADGGYAFKYENGKYDIKDVGVDNAGAKAGLTFLVDLIKNKH
MNADTDYSIAEAAFNKGETAMTINGPWAWSNIDTSKVNYGVTVLPTFKGQPSKPFVGVLSAGINAASP
NKELAKEFLENYLLTDEGLEAVNKDKPLGAVALKSYEEELAKDPRIAATMENAQKGEIMPNIPQMSAF
WYAVRTAVINAASGRQTVDEALKDAQTNSSSNNNNNNNNNNLGENLYFQGSDPAALKRARNTEAARRS
RARKLQRMKQLEDKVYHLENEVARLKKLVGELQKLQRVKRARNTEAARRSRARKAALKG

Appendix 4.4.4: pMAL-LZD73 ORF

ATGAAAATCGAAGAAGGTAAACTGGTAATCTGGATTAACGGCGATAAAGGCTATAACGGTCTCGCTGA
AGTCGGTAAGAAATTCGAGAAAGATACCGGAATTAAAGTCACCGTTGAGCATCCGGATAAACTGGAAG
AGAAATTCCCACAGGTTGCGGCAACTGGCGATGGCCCTGACATTATCTTCTGGGCACACGACCGCTTT
GGTGGCTACGCTCAATCTGGCCTGTTGGCTGAAATCACCCCGGACAAAGCGTTCCAGGACAAGCTGTA
TCCGTTTACCTGGGATGCCGTACGTTACAACGGCAAGCTGATTGCTTACCCGATCGCTGTTGAAGCGT
TATCGCTGATTTATAACAAAGATCTGCTGCCGAACCCGCCAAAAACCTGGGAAGAGATCCCGGCGCTG
GATAAAGAACTGAAAGCGAAAGGTAAGAGCGCGCTGATGTTCAACCTGCAAGAACCGTACTTCACCTG
GCCGCTGATTGCTGCTGACGGGGGTTATGCGTTCAAGTATGAAAACGGCAAGTACGACATTAAAGACG
TGGGCGTGGATAACGCTGGCGCGAAAGCGGGTCTGACCTTCCTGGTTGACCTGATTAAAAACAAACAC
ATGAATGCAGACACCGATTACTCCATCGCAGAAGCTGCCTTTAATAAAGGCGAAACAGCGATGACCAT
CAACGGCCCGTGGGCATGGTCCAACATCGACACCAGCAAAGTGAATTATGGTGTAACGGTACTGCCGA
CCTTCAAGGGTCAACCATCCAAACCGTTCGTTGGCGTGCTGAGCGCAGGTATTAACGCCGCCAGTCCG
AACAAAGAGCTGGCAAAAGAGTTCCTCGAAAACTATCTGCTGACTGATGAAGGTCTGGAAGCGGTTAA
TAAAGACAAACCGCTGGGTGCCGTAGCGCTGAAGTCTTACGAGGAAGAGTTGGCGAAAGATCCACGTA
TTGCCGCCACTATGGAAAACGCCCAGAAAGGTGAAATCATGCCGAACATCCCGCAGATGTCCGCTTTC
TGGTATGCCGTGCGTACTGCGGTGATCAACGCCGCCAGCGGTCGTCAGACTGTCGATGAAGCCCTGAA
AGACGCGCAGACTAATTCGAGCTCGAACAACAACAACAATAACAATAACAACAACCTCGGGGAAAACC
TGTATTTTCAGGGATCCGATCCAGCTGCTCTGAAGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGC
CGAGCTCGGAAGCTGCAACGAATGAAGCAGCTGGAAGACAAGGTGGAGGAACTGCTGAGCAAGAACTA
CCACCTGGAGAACGAAGTTGCGCGCCTGAAGAAGCTGGTGGGTGAACTGCAGAAGTTACAGCGGGTGA
AGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGCCGAGCTCGAAAGGCTGCTCTGAAGGGATAGTAA
GAATTCG

Appendix 4.4.5: pMAL-LZD73 Protein Sequence

MKIEEGKLVIWINGDKGYNGLAEVGKKFEKDTGIKVTVEHPDKLEEKFPQVAATGDGPDIIFWAHDRF
GGYAQSGLLAEITPDKAFQDKLYPFTWDAVRYNGKLIAYPIAVEALSLIYNKDLLPNPPKTWEEIPAL
DKELKAKGKSALMFNLQEPYFTWPLIAADGGYAFKYENGKYDIKDVGVDNAGAKAGLTFLVDLIKNKH
MNADTDYSIAEAAFNKGETAMTINGPWAWSNIDTSKVNYGVTVLPTFKGQPSKPFVGVLSAGINAASP
NKELAKEFLENYLLTDEGLEAVNKDKPLGAVALKSYEEELAKDPRIAATMENAQKGEIMPNIPQMSAF
WYAVRTAVINAASGRQTVDEALKDAQTNSSSNNNNNNNNNNLGENLYFQGSDPAALKRARNTEAARRS
RARKLQRMKQLEDKVEELLSKNYHLENEVARLKKLVGELQKLQRVKRARNTEAARRSRARKAALKG

Appendix 4.4.6: pMAL-LZD80 ORF

```
ATGAAAATCGAAGAAGGTAAACTGGTAATCTGGATTAACGGCGATAAAGGCTATAACGGTCTCGCTGA
AGTCGGTAAGAAATTCGAGAAAGATACCGGAATTAAAGTCACCGTTGAGCATCCGGATAAACTGGAAG
AGAAATTCCCACAGGTTGCGGCAACTGGCGATGGCCCTGACATTATCTTCTGGGCACACGACCGCTTT
GGTGGCTACGCTCAATCTGGCCTGTTGGCTGAAATCACCCCGGACAAAGCGTTCCAGGACAAGCTGTA
TCCGTTTACCTGGGATGCCGTACGTTACAACGGCAAGCTGATTGCTTACCCGATCGCTGTTGAAGCGT
TATCGCTGATTTATAACAAAGATCTGCTGCCGAACCCGCCAAAAACCTGGGAAGAGATCCCGGCGCTG
GATAAAGAACTGAAAGCGAAAGGTAAGAGCGCGCTGATGTTCAACCTGCAAGAACCGTACTTCACCTG
GCCGCTGATTGCTGCTGACGGGGGTTATGCGTTCAAGTATGAAAACGGCAAGTACGACATTAAAGACG
TGGGCGTGGATAACGCTGGCGCGAAAGCGGGTCTGACCTTCCTGGTTGACCTGATTAAAAACAAACAC
ATGAATGCAGACACCGATTACTCCATCGCAGAAGCTGCCTTTAATAAAGGCGAAACAGCGATGACCAT
CAACGGCCCGTGGGCATGGTCCAACATCGACACCAGCAAAGTGAATTATGGTGTAACGGTACTGCCGA
CCTTCAAGGGTCAACCATCCAAACCGTTCGTTGGCGTGCTGAGCGCAGGTATTAACGCCGCCAGTCCG
AACAAAGAGCTGGCAAAAGAGTTCCTCGAAAACTATCTGCTGACTGATGAAGGTCTGGAAGCGGTTAA
TAAAGACAAACCGCTGGGTGCCGTAGCGCTGAAGTCTTACGAGGAAGAGTTGGCGAAAGATCCACGTA
TTGCCGCCACTATGGAAAACGCCCAGAAAGGTGAAATCATGCCGAACATCCCGCAGATGTCCGCTTTC
TGGTATGCCGTGCGTACTGCGGTGATCAACGCCGCCAGCGGTCGTCAGACTGTCGATGAAGCCCTGAA
AGACGCGCAGACTAATTCGAGCTCGAACAACAACAACAATAACAATAACAACAACCTCGGGGAAAACC
TGTATTTTCAGGGATCCGATCCAGCTGCTCTGAAGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGC
CGAGCTCGGAAGCTGCAACGAATGAAGCAGCTGGAAGACAAGGTGGAGGAACTGCTGAGCAAGAACTA
CCACCTGGAGAACGAAGTTGCGCGCCTGAAAAAGCTGGTGGAAGAACTGCTGAGCAAAGTGGGCGAAC
TGCAGAAGTTACAGCGGGTGAAGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGCCGAGCTCGAAAG
GCTGCTCTGAAGGGATAGTAAGAATTCG
```

Appendix 4.4.7: pMAL-LZD80 Protein Sequence

```
MKIEEGKLVIWINGDKGYNGLAEVGKKFEKDTGIKVTVEHPDKLEEKFPQVAATGDGPDIIFWAHDRF
GGYAQSGLLAEITPDKAFQDKLYPFTWDAVRYNGKLIAYPIAVEALSLIYNKDLLPNPPKTWEEIPAL
DKELKAKGKSALMFNLQEPYFTWPLIAADGGYAFKYENGKYDIKDVGVDNAGAKAGLTFLVDLIKNKH
MNADTDYSIAEAAFNKGETAMTINGPWAWSNIDTSKVNYGVTVLPTFKGQPSKPFVGVLSAGINAASP
NKELAKEFLENYLLTDEGLEAVNKDKPLGAVALKSYEEELAKDPRIAATMENAQKGEIMPNIPQMSAF
WYAVRTAVINAASGRQTVDEALKDAQTNSSSNNNNNNNNNNLGENLYFQGSDPAALKRARNTEAARRS
RARKLQRMKQLEDKVEELLSKNYHLENEVARLKKLVEELLSKVGELQKLQRVKRARNTEAARRSRARK
AALKG
```

Appendix 4.4.8: pMAL-LZD87 ORF

```
ATGAAAATCGAAGAAGGTAAACTGGTAATCTGGATTAACGGCGATAAAGGCTATAACGGTCTCGCTGA
AGTCGGTAAGAAATTCGAGAAAGATACCGGAATTAAAGTCACCGTTGAGCATCCGGATAAACTGGAAG
AGAAATTCCCACAGGTTGCGGCAACTGGCGATGGCCCTGACATTATCTTCTGGGCACACGACCGCTTT
GGTGGCTACGCTCAATCTGGCCTGTTGGCTGAAATCACCCCGGACAAAGCGTTCCAGGACAAGCTGTA
TCCGTTTACCTGGGATGCCGTACGTTACAACGGCAAGCTGATTGCTTACCCGATCGCTGTTGAAGCGT
TATCGCTGATTTATAACAAAGATCTGCTGCCGAACCCGCCAAAAACCTGGGAAGAGATCCCGGCGCTG
GATAAAGAACTGAAAGCGAAAGGTAAGAGCGCGCTGATGTTCAACCTGCAAGAACCGTACTTCACCTG
GCCGCTGATTGCTGCTGACGGGGGTTATGCGTTCAAGTATGAAAACGGCAAGTACGACATTAAAGACG
TGGGCGTGGATAACGCTGGCGCGAAAGCGGGTCTGACCTTCCTGGTTGACCTGATTAAAAACAAACAC
ATGAATGCAGACACCGATTACTCCATCGCAGAAGCTGCCTTTAATAAAGGCGAAACAGCGATGACCAT
CAACGGCCCGTGGGCATGGTCCAACATCGACACCAGCAAAGTGAATTATGGTGTAACGGTACTGCCGA
CCTTCAAGGGTCAACCATCCAAACCGTTCGTTGGCGTGCTGAGCGCAGGTATTAACGCCGCCAGTCCG
AACAAAGAGCTGGCAAAAGAGTTCCTCGAAAACTATCTGCTGACTGATGAAGGTCTGGAAGCGGTTAA
TAAAGACAAACCGCTGGGTGCCGTAGCGCTGAAGTCTTACGAGGAAGAGTTGGCGAAAGATCCACGTA
TTGCCGCCACTATGGAAAACGCCCAGAAAGGTGAAATCATGCCGAACATCCCGCAGATGTCCGCTTTC
TGGTATGCCGTGCGTACTGCGGTGATCAACGCCGCCAGCGGTCGTCAGACTGTCGATGAAGCCCTGAA
```

AGACGCGCAGACTAATTCGAGCTCGAACAACAACAACAATAACAATAACAACAACCTCGGGGAAAACC
TGTATTTTCAGGGATCCGATCCAGCTGCTCTGAAGCGAGCTCGGAACACTGAAGCTGCTCGACGGAGC
CGAGCTCGGAAGCTGCAACGAATGAAGCAGCTGGAAGACAAGGTGGAGGAACTGCTGAGCAAGAACTA
CCACCTGGAGAACGAAGTTGCGCGCCTGAAAAAGCTGGTGGAAGAACTGCTGAGCAAAGTGCGTGCGC
TGGCGGATTCTCTGGGCGAACTGCAGAAGTTACAGCGGGTGAAGCGAGCTCGGAACACTGAAGCTGCT
CGACGGAGCCGAGCTCGAAAGGCTGCTCTGAAGGGATAGTAAGAATTCG

Appendix 4.4.9: pMAL-LZD87 Protein Sequence

MKIEEGKLVIWINGDKGYNGLAEVGKKFEKDTGIKVTVEHPDKLEEKFPQVAATGDGPDIIFWAHDRF
GGYAQSGLLAEITPDKAFQDKLYPFTWDAVRYNGKLIAYPIAVEALSLIYNKDLLPNPPKTWEEIPAL
DKELKAKGKSALMFNLQEPYFTWPLIAADGGYAFKYENGKYDIKDVGVDNAGAKAGLTFLVDLIKNKH
MNADTDYSIAEAAFNKGETAMTINGPWAWSNIDTSKVNYGVTVLPTFKGQPSKPFVGVLSAGINAASP
NKELAKEFLENYLLTDEGLEAVNKDKPLGAVALKSYEEELAKDPRIAATMENAQKGEIMPNIPQMSAF
WYAVRTAVINAASGRQTVDEALKDAQTNSSSNNNNNNNNNNLGENLYFQGSDPAALKRARNTEAARRS
RARKLQRMKQLEDKVEELLSKNYHLENEVARLKKLVEELLSKVRALADSLGELQKLQRVKRARNTEAA
RRSRARKAALKG

Appendix 4.4.10: TEV Protease Protein Sequence

TEV protease was expressed and purified from the pRK793 vector which is available online. After internal cleavage of the expressed fusion the protease exists as the following sequence:

GHHHHHHHGESLFKGPRDYNPISSTICHLTNESDGHTTSLYGIGFGPFIITNKHLFRRNNGTLLVQSL
HGVFKVKNTTTLQQHLIDGRDMIIIRMPKDFPPFPQKLKFREPQREERICLVTTNFQTKSMSSMVSDT
SCTFPSSDGIFWKHWIQTKDGQCGSPLVSTRDGFIVGIHSASNFTNTNNYFTSVPKNFMELLTNQEAQ
QWVSGWRLNADSVLWGGHKVFMVKPEEPFQPVKEATQLMNRRRRR

Appendix 4.4.11: Removed pMAL-c2T-rev MCS Sequence

GATCCTCTAGAGTCGACCTGCAGGC

Appendix 5: Assorted Programs

Appendix 5.1: sizecount.pl

The following perl code was written to identify the indices in appendix 5.2 within an assembled fastq sequence file. Then, based on the found indices, each type of molecule is identified, counted, and full counts for all observed classifications of interest are reported out.

The code printed here was the code at the time final assignments for this dissertation were performed. The full code can also be found on GitHub with any updates at:

https://github.com/jhustedt/kahn-ngs

```
#!/usr/bin/env perl
use Modern::Perl;
use autodie qw":all";
use warnings qw"all";
use diagnostics;
use strict;

use Bio::Seq;
use Bio::SeqIO;
use Cwd qw(abs_path getcwd);
use File::Basename;
use File::Find;
use FileHandle;
use File::Path qw"make_path";
use File::Spec qw"rel2abs";
use Getopt::Long qw"GetOptions";
use PerlIO;
use PerlIO::gzip;
use String::Approx qw"amatch aindex";

$SIG{INT} = \&End_Handler;
$SIG{TERM} = \&End_Handler;

=head1 NAME

  sizecount.pl - A writeall script to sort sequences by user index
    and infer the sizes of the template DNA in a ring closure
    experiment.

=head1 SYNOPSIS

  This script has a few useful options:
```

```
--index  : The file defining the search strings and output files.
--input  : Either a filename or directory containing the (relatively) raw
   fastq data.
--outdir : Output directory, composed of two csv files, an output fastq file,
   and a summary file.
--outfastq : Output fastq file, by default this is placed into the output
   directory. File extension not necessary.
--substitution : String::Approx (Levenshtein) substitution distance, default
   is 0 (perfect match)
--insertion: Levenshtein insertion distance
--deletion: Levenshtein deletion distance
--spacer : specify spacer length (default 72)
--debug  : Print a bunch of debugging output
All of the output scripts have default names, any individual name can be changed
as an option, these are not listed here for brevity.

=head1 DESCRIPTION

This script was adapted from work by Dr. Ashton Trey Belew: https://github.com/abelew

=cut

#options from Getopt::Long; defaults
my %options = (
   debug => 0,
   indices => 'index.txt',
   input => 'test.fastq.gz',
   outdir => 'output',
   outfastq => 'out',
   summary => 'summary.txt',
   unicyc_csv => 'unimolecular_ligated_lengths.csv',
   unicycfull_csv => 'unimolecular_ligated_lengths_full.csv',
   fourhitlin_csv => 'linear_fourhit_lengths.csv',
   fourhitlinfull_csv => 'linear_fourhit_lengths_full.csv',
   threehitlin_csv => 'linear_threehit_lengths.csv',
   threehitlinfull_csv => 'linear_threehit_lengths_full.csv',
   onehitlincyc_csv => 'linear_onehit_cyc_lengths.csv',
   onehitlinsynth_csv => 'linear_onehit_synth_lengths.csv',
   bicyc4_csv => 'bimolecular4_ligated_lengths.csv',
   bicyc4full_csv => 'bimolecular4_ligated_lengths_full.csv',
   bicyc4varlib_csv => 'bimolecular4_ligated_variable_library_lengths.csv',
   bicyc4varlibfull_csv => 'bimolecular4_ligated_variable_library_lengths_full.csv',
   bicyc4steplib_csv => 'bimolecular4_ligated_step_library_lengths.csv',
   bicyc4steplibfull_csv => 'bimolecular4_ligated_step_library_lengths_full.csv',
   bicyc6_csv => 'bimolecular6_ligated_lengths.csv',
   bicyc6full_csv => 'bimolecular6_ligated_lengths_full.csv',
   bicyc5_csv => 'bimolecular5_ligated_lengths.csv',
   bicyc5full_csv => 'bimolecular5_ligated_lengths_full.csv',
   bicyc5frag_csv => 'bimolecular5_ligated_lengths_frag.csv',
   bicyc2_csv => 'bimolecular2_ligated_lengths.csv',
   bicyc2full_csv => 'bimolecular2_ligated_lengths_full.csv',
   varlibfull_csv => 'library_variable_full.csv',
   steplibfull_csv => 'library_step_full.csv',
   spacer => 72,
   substitution => 0,
```

```
    insertion => 0,
    deletion => 0,
);
## This is a hash counting off how many times _every_ index is observed.
my $observed_reads = 0;
my %observations = (
    helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
## This counts how often each index is observed when 1-10 indices are observed.
my %singles = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
my %doubles = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
my %triples = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
my %quads = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
my %fives = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
my %sixes = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
## in theory hits below here should not exist
my %sevenup = (
    sum => 0, helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,);
## set up tags for counting and appropriate final lengths outputs for each match
my $datestring = localtime();
my $found_all_four = 0;
my $found_four_uni = 0;
my $found_four_lin = 0;
my $found_three_lin = 0;
my $found_one_lin_cyc = 0;
my $found_one_lin_synth = 0;
my $found_three_unknown = 0;
my $found_one_unknown = 0;
my $found_four_unknown = 0;
my %unicyclized4_final_lengths = ();
my %unicyclized4_full_final_lengths = ();
my %linear4_final_lengths = ();
my %linear4_full_final_lengths = ();
my %linear3_final_lengths = ();
my %linear3_full_final_lengths = ();
my %linear1_final_lengths_cyc = ();
my %linear1_final_lengths_synth = ();
my $found_four_bi = 0;
my %bicyclized4_final_lengths = ();
my %bicyclized4_full_final_lengths = ();
my $found_four_varlib_bi = 0;
my $found_four_steplib_bi = 0;
```

```
my %bicyclized4_var_lib_final_lengths = ();
my %bicyclized4_var_lib_full_final_lengths = ();
my %bicyclized4_step_lib_final_lengths = ();
my %bicyclized4_step_lib_full_final_lengths = ();
my $found_six_bi = 0;
my $found_six_unknown = 0;
my %bicyclized6_final_lengths = ();
my %bicyclized6_full_final_lengths = ();
my $found_five_bi = 0;
my $found_five_unknown = 0;
my %bicyclized5_final_lengths = ();
my %bicyclized5_full_final_lengths = ();
my %bicyclized5_frag_final_lengths = ();
my $found_two_bi = 0;
my $found_two_unknown = 0;
my $found_two_lib_var = 0;
my $found_two_lib_step = 0;
my %bicyclized2_final_lengths = ();
my %bicyclized2_full_final_lengths = ();
my %bi2_full_lib_lengths = ();
my %bi2_full_step_lengths = ();
my $opt_result = GetOptions(
    "debug:i" => \$options{debug},
    "spacer:i" => \$options{spacer},
    "indices:s" => \$options{indices},
    "input:s" => \$options{input},
    "outdir:s" => \$options{outdir},
    "summary:s" => \$options{summary},
    "unicyc_csv:s" => \$options{unicyc_csv},
    "unicycfull_csv:s" => \$options{unicycfull_csv},
    "fourhitlin_csv:s" => \$options{fourhitlin_csv},
    "fourhitlinfull_csv:s" => \$options{fourhitlinfull_csv},
    "threehitlin_csv:s" => \$options{threehitlin_csv},
    "threehitlinfull_csv:s" => \$options{threehitlinfull_csv},
    "onehitlincyc_csv:s" => \$options{onehitlincyc_csv},
    "onehitlinsynth_csv:s" => \$options{onehitlinsynth_csv},
    "bicyc4_csv:s" => \$options{bicyc4_csv},
    "bicyc4full_csv:s" => \$options{bicyc4full_csv},
    "bicyc4varlib_csv:s" => \$options{bicyc4varlib_csv},
    "bicyc4varlibfull_csv:s" => \$options{bicyc4varlibfull_csv},
    "bicyc4steplib_csv:s" => \$options{bicyc4steplib_csv},
    "bicyc4steplibfull_csv:s" => \$options{bicyc4steplibfull_csv},
    "bicyc6_csv:s" => \$options{bicyc6_csv},
    "bicyc6full_csv:s" => \$options{bicyc6full_csv},
    "bicyc5_csv:s" => \$options{bicyc5_csv},
    "bicyc5full_csv:s" => \$options{bicyc5full_csv},
    "bicyc5frag_csv:s" => \$options{bicyc5frag_csv},
    "bicyc2_csv:s" => \$options{bicyc2_csv},
    "bicyc2full_csv:s" => \$options{bicyc2full_csv},
    "varlibfull_csv:s" => \$options{varlibfull_csv},
    "steplibfull_csv:s" => \$options{steplibfull_csv},
    "outfastq:s" => \$options{outfastq},
    "substitution:i" => \$options{substitution},
    "insertion:i" => \$options{insertion},
    "deletion:i" => \$options{deletion},
```

```
);
my $log = new FileHandle(">$options{outdir}/$options{summary}");
my $unicyc_csv = new FileHandle(">$options{outdir}/$options{unicyc_csv}");
my $unicycfull_csv = new FileHandle(">$options{outdir}/$options{unicycfull_csv}");
my $fourhitlin_csv = new FileHandle(">$options{outdir}/$options{fourhitlin_csv}");
my $fourhitlinfull_csv = new FileHandle(">$options{outdir}/$options{fourhitlinfull_csv}");
my $bicyc4_csv = new FileHandle(">$options{outdir}/$options{bicyc4_csv}");
my $bicyc4full_csv = new FileHandle(">$options{outdir}/$options{bicyc4full_csv}");
my $bicyc4varlib_csv = new FileHandle(">$options{outdir}/$options{bicyc4varlib_csv}");
my $bicyc4varlibfull_csv = new FileHandle(">$options{outdir}/$options{bicyc4varlibfull_csv}");
my $bicyc4steplib_csv = new FileHandle(">$options{outdir}/$options{bicyc4steplib_csv}");
my $bicyc4steplibfull_csv = new FileHandle(">$options{outdir}/$options{bicyc4steplibfull_csv}");
my $bicyc6_csv = new FileHandle(">$options{outdir}/$options{bicyc6_csv}");
my $bicyc6full_csv = new FileHandle(">$options{outdir}/$options{bicyc6full_csv}");
my $bicyc2_csv = new FileHandle(">$options{outdir}/$options{bicyc2_csv}");
my $bicyc2full_csv = new FileHandle(">$options{outdir}/$options{bicyc2full_csv}");
my $bicyc5_csv = new FileHandle(">$options{outdir}/$options{bicyc5_csv}");
my $bicyc5full_csv = new FileHandle(">$options{outdir}/$options{bicyc5full_csv}");
my $bicyc5frag_csv = new FileHandle(">$options{outdir}/$options{bicyc5frag_csv}");
my $threehitlin_csv = new FileHandle(">$options{outdir}/$options{threehitlin_csv}");
my $threehitlinfull_csv = new FileHandle(">$options{outdir}/$options{threehitlinfull_csv}");
my $onehitlincyc_csv = new FileHandle(">$options{outdir}/$options{onehitlincyc_csv}");
my $onehitlinsynth_csv = new FileHandle(">$options{outdir}/$options{onehitlinsynth_csv}");
my $varlibfull_csv = new FileHandle(">$options{outdir}/$options{varlibfull_csv}");
my $steplibfull_csv = new FileHandle(">$options{outdir}/$options{steplibfull_csv}");

if (!-r $options{input}) {
   die("The input file: $options{input} does not exist.");
}
if (!-r $options{indices}) {
   die("The index file: $options{indices} does not exist.");
}
if (!-d $options{outdir}) {
   die("The output directory $options{outdir} does not exist.");
}
my $index_hash = Read_indices();

my $abs_input = File::Spec->rel2abs($options{input});
my $fastq_output = qq"$options{outdir}/$options{outfastq}.fastq.gz";
my $abs_output = File::Spec->rel2abs($fastq_output);
my $out;
if ($abs_input eq $abs_output) {
   die("The input file and output file are the same.");
} else {
   $out = FileHandle->new("| gzip -f -9 > $fastq_output");
}
my $reads;
if (-f $options{input}) {
   $reads = Sort_File_Approx(
      input => $options{input},
      outdir => $options{outdir},
      index_hash => $index_hash,
   );
} elsif (-d $options{input}) {
   Sort_Dir(
```

```
        indir => $options{input},
        outdir => $options{outdir},
        index_hash => $index_hash,
    );
} else {
    die("I need a file/directory containing some sequence.") unless($options{input});
}
End_Handler();

=item Sort_File_Approx

  This function should look through every sequence for a reasonable
  match to the available indices. If it gets some hits, record them.
  If they are ambiguous, this should return where the possibilities
  lie. If no match, return that.

=cut

sub Sort_File_Approx {
    my %args = @_;
    my $data = $args{index_hash};
    return(undef) unless ($args{input} =~ /\.fastq/);
    if (!-d $args{outdir}) {
        make_path($args{outdir});
    }
    my $inputted = FileHandle->new("zless $args{input} 2>/dev/null |");
    my $in = Bio::SeqIO->new(-fh => $inputted, -format => 'Fastq');
    my $count = 0;
  READS: while (my $in_seq = $in->next_dataset()) {
        $count++;
        $observed_reads++;
        my $id = $in_seq->{'-descriptor'};
        my $sequence = $in_seq->{'-seq'};
        my $qual = $in_seq->{'-raw_quality'};
        my $comment = $in_seq->{'-comment'};
        my $seqlen = length($sequence);
        if (!defined($comment)) {
            $comment = "";
        }
        my $found = 0;
        my @index_list = @{$data->{possibilities}};
        ## Reminder of what the data structure looks like: template => {
        ##         AAAATTTTCCC => { verb => 'add', direction => 'fwd',
        ##         number => 10, found => 0, total_observed => 0,
        ##         ambiguous_observed => 0, unique_observed => 0 }, },
        my $found_id = '';
        ## substring looking from 0-9, switched to regular expression match to
        ## scan full seq line my $match_substring = substr($sequence, 0, 9);
        ## I am creating a hash of observations for each sequence, and one for
        ## all sequences.
        my %observe = (
            helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
            helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,
            unknown => 0);
        my %positions = (
```

```
    helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,
    unknown => 0);
my %numbers = (
    helical_fwd => 0, stepcyc_fwd => 0, variable_fwd => 0, stepsynth_fwd => 0,
    helical_rev => 0, stepcyc_rev => 0, variable_rev => 0, stepsynth_rev => 0,
    unknown => 0);
my $observed_indices = 0;
foreach my $index (@index_list) {
    my $info = $data->{$index};
    ## direct sequence match, without error
    ## set up array for parameters for string::approx
    my $params = [ "I$options{insertion}","D$options{deletion}","S$options{substitution}" ];
    my @starts = ();
    if ( !defined $options{insertion} && !defined $options{deletion} &&
          !defined $options{substitution} ) {
        $sequence =~ m/$index/;
        @starts = @-;
    } else {
        @starts = aindexes($index, $params, ($sequence));
    }
    if (@starts) {
        $found++;
        $found_id = $index;
        for my $st (@starts) {
            if ($options{debug} == 1) {
                print "TESTME: @starts\n";
            }
            $comment .= "$st:$info->{name}:$info->{number}:$info->{direction} ";
            ## where "st" is position within read, info name
            ## and number identify the index, and direction
            ## identifies fwd or rev.
            ## fwd & rev needed as adapter attachment to
            ## flow cell is random & can occur in either
            ## direction.
            if ($info->{name} eq 'helical' && $info->{direction} eq 'fwd') {
                $observations{helical_fwd}++;
                $observe{helical_fwd}++;
                $positions{helical_fwd} = $st;
                $observed_indices++;
                $numbers{helical_fwd} = $info->{number};
            } elsif ($info->{name} eq 'stepcyc' && $info->{direction} eq 'fwd') {
                $observations{stepcyc_fwd}++;
                $observe{stepcyc_fwd}++;
                $positions{stepcyc_fwd} = $st;
                $observed_indices++;
                $numbers{stepcyc_fwd} = $info->{number};
            } elsif ($info->{name} eq 'stepsynth' && $info->{direction} eq 'fwd') {
                $observations{stepsynth_fwd}++;
                $observe{stepsynth_fwd}++;
                $positions{stepsynth_fwd} = $st;
                $observed_indices++;
                $numbers{stepsynth_fwd} = $info->{number};
            } elsif ($info->{name} eq 'variable' && $info->{direction} eq 'fwd') {
                $observations{variable_fwd}++;
```

```
            $observe{variable_fwd}++;
            $positions{variable_fwd} = $st;
            $observed_indices++;
            $numbers{variable_fwd} = $info->{number};
          } elsif ($info->{name} eq 'helical' && $info->{direction} eq 'rev') {
            $observations{helical_rev}++;
            $observe{helical_rev}++;
            $positions{helical_rev} = $st;
            $observed_indices++;
            $numbers{helical_rev} = $info->{number};
          } elsif ($info->{name} eq 'stepcyc' && $info->{direction} eq 'rev') {
            $observations{stepcyc_rev}++;
            $observe{stepcyc_rev}++;
            $positions{stepcyc_rev} = $st;
            $observed_indices++;
            $numbers{stepcyc_rev} = $info->{number};
          } elsif ($info->{name} eq 'stepsynth' && $info->{direction} eq 'rev') {
            $observations{stepsynth_rev}++;
            $observe{stepsynth_rev}++;
            $positions{stepsynth_rev} = $st;
            $observed_indices++;
            $numbers{stepsynth_rev} = $info->{number};
          } elsif ($info->{name} eq 'variable' && $info->{direction} eq 'rev') {
            $observations{variable_rev}++;
            $observe{variable_rev}++;
            $positions{variable_rev} = $st;
            $observed_indices++;
            $numbers{variable_rev} = $info->{number};
          }
        }
      }
}       ## End each index
## to debug - print comment to standard out
if ($options{debug} == 1) {
   print STDOUT "$comment ";
}
## set counters to zero for each type we are looking for, only move forward if found
my $fwd_valid = 0;
my $rev_valid = 0;
my $helical = 0;
my $stepcyc = 0;
my $stepsynth = 0;
my $variable = 0;
my $bimol_valid = 0;
my $helicalfwd = 0;
my $helicalrev = 0;
my $variablefwd = 0;
my $variablerev = 0;
my $stepcycfwd = 0;
my $stepcycrev = 0;
my $stepsynthfwd = 0;
my $stepsynthrev = 0;
my $type = "yes";
## set hashes to zero for final size range if no molecule was found of that size
my $short_size = [119..219];
```

```
for my $shortsize (@{$short_size}) {
   if (!defined($unicyclized4_final_lengths{$shortsize})) {
      $unicyclized4_final_lengths{$shortsize} = 0;
   }
   if (!defined($linear4_final_lengths{$shortsize})) {
      $linear4_final_lengths{$shortsize} = 0;
   }
   if (!defined($linear3_final_lengths{$shortsize})) {
      $linear3_final_lengths{$shortsize} = 0;
   }
   if (!defined($bicyclized4_final_lengths{$shortsize})) {
      $bicyclized4_final_lengths{$shortsize} = 0;
   }
   if (!defined($bicyclized6_final_lengths{$shortsize})) {
      $bicyclized6_final_lengths{$shortsize} = 0;
   }
   if (!defined($bicyclized5_final_lengths{$shortsize})) {
      $bicyclized5_final_lengths{$shortsize} = 0;
   }
}
my $frag_size = ['00','01','02','03','04','05','06','07','08','09',10..40];
for my $fragsize (@{$frag_size}) {
   if (!defined($bicyclized5_final_lengths{"0".$fragsize})) {
      $bicyclized5_final_lengths{"0".$fragsize} = 0;
      # when pre-populating this hash I am doing so with
      # a leading zero such that the values are three digits,
      # this allows for them to end in the same hash as the
      # regular size library as the range of sizes here is
      # 000-040 and the regular size library is 119-219.
      # if a different size library is being used this
      # should be changed to a different hash entirely
   }
   if (!defined($bicyclized4_var_lib_final_lengths{"0".$fragsize})) {
      $bicyclized4_var_lib_final_lengths{"0".$fragsize} = 0;
   }
}
## set hashes to zero for full name if no molecule was found of that name
my $step_sizes = ['047','077','107'];
my $var_sizes = ['00','01','02','03','04','05','06','07','08','09',10..30];
my $hel_sizes = ['00','01','02','03','04','05','06','07','08','09','10'];
my @full_sizes = ();
foreach my $s (@{$step_sizes}) {
   foreach my $v (@{$var_sizes}) {
      foreach my $h (@{$hel_sizes}) {
         my $entry = qq"${s},${v},${h}";
         push (@full_sizes,$entry);
      }
   }
}
my @frag_full = ();
foreach my $v (@{$var_sizes}) {
   foreach my $h (@{$hel_sizes}) {
      my $entry = qq"${v},${h}";
      push (@frag_full,$entry);
   }
```

```
    }
    foreach my $fullsize (@{full_sizes}) {
      if (!defined($unicyclized4_full_final_lengths{$fullsize})) {
        $unicyclized4_full_final_lengths{$fullsize} = 0;
      }
      if (!defined($linear4_full_final_lengths{$fullsize})) {
        $linear4_full_final_lengths{$fullsize} = 0;
      }
      if (!defined($linear3_full_final_lengths{$fullsize})) {
        $linear3_full_final_lengths{$fullsize} = 0;
      }
      if (!defined($bicyclized4_full_final_lengths{$fullsize})) {
        $bicyclized4_full_final_lengths{$fullsize} = 0;
      }
      if (!defined($bicyclized6_full_final_lengths{$fullsize})) {
        $bicyclized6_full_final_lengths{$fullsize} = 0;
      }
      if (!defined($bicyclized5_full_final_lengths{$fullsize})) {
        $bicyclized5_full_final_lengths{$fullsize} = 0;
      }
    }
    for my $fragfull (@{frag_full}) {
      if (!defined($bicyclized5_frag_final_lengths{$fragfull})) {
        $bicyclized5_frag_final_lengths{$fragfull} = 0;
      }
      if (!defined($bicyclized4_var_lib_full_final_lengths{$fragfull})) {
        $bicyclized4_var_lib_full_final_lengths{$fragfull} = 0;
      }
      if (!defined($bi2_full_lib_lengths{$fragfull})) {
        $bi2_full_lib_lengths{$fragfull} = 0;
      }
    }
    ## start looking for matches & assigning them
    if ($observed_indices == 4 && $observe{stepsynth_fwd} > 0 && $observe{helical_fwd} > 0
&&
        $observe{stepcyc_fwd} > 0 && $observe{variable_fwd} > 0) {
      $fwd_valid = 1;
      $found_all_four++;
      my @pieces = split(/\s+/, $comment);
      for my $p (@pieces) {
        my ($position, $piece, $name, $dir)  = split(/:/, $p);
        if ($piece eq 'helical' && $dir eq 'fwd') {
          $helical = $name;
        } elsif ($piece eq 'stepcyc' && $dir eq 'fwd') {
          $stepcyc = $name;
        } elsif ($piece eq 'variable' && $dir eq 'fwd') {
          $variable = $name;
        } elsif ($piece eq 'stepsynth' && $dir eq 'fwd') {
          $stepsynth = $name;
        }
      }
      my $final_size = $options{spacer} + $helical + $stepsynth + $variable;
      $comment .= "final size: $final_size ";
      if ($positions{stepcyc_fwd} < $positions{helical_fwd} &&
          $positions{helical_fwd} < $positions{variable_fwd} &&
```

```
            $positions{variable_fwd} < $positions{stepsynth_fwd} &&
            $numbers{stepsynth_fwd} == $numbers{stepcyc_fwd}) {
          $type = "unimolecular";
          $found_four_uni++;
          if (!defined($unicyclized4_final_lengths{$final_size})) {
            $unicyclized4_final_lengths{$final_size} = 1;
          } else {
            $unicyclized4_final_lengths{$final_size}++;
          }
          if
(!defined($unicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$
numbers{helical_fwd}})) {

$unicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{
helical_fwd}} = 1;
          } else {

$unicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{
helical_fwd}}++;
          }
        } elsif ($positions{stepcyc_fwd} < $positions{helical_fwd} &&
                 $positions{helical_fwd} < $positions{variable_fwd} &&
                 $positions{variable_fwd} < $positions{stepsynth_fwd} &&
                 $numbers{stepsynth_fwd} != $numbers{stepcyc_fwd}) {
          $type = "bimolecular-4";
          $found_four_bi++;
          if (!defined($bicyclized4_final_lengths{$final_size})) {
            $bicyclized4_final_lengths{$final_size} = 1;
          } else {
            $bicyclized4_final_lengths{$final_size}++;
          }
          if
(!defined($bicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$n
umbers{helical_fwd}})) {

$bicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}} = 1;
          } else {

$bicyclized4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}}++;
          }
        } elsif ($positions{helical_fwd} < $positions{variable_fwd} &&
                 $positions{variable_fwd} < $positions{stepsynth_fwd} &&
                 $positions{stepsynth_fwd} < $positions{stepcyc_fwd}) {
          $type = "linear";
          $found_four_lin++;
          if (!defined($linear4_final_lengths{$final_size})) {
            $linear4_final_lengths{$final_size} = 1;
          } else {
            $linear4_final_lengths{$final_size}++;
          }
          if
(!defined($linear4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numb
ers{helical_fwd}})) {
```

```
$linear4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{helica
l_fwd}} = 1;
            } else {

$linear4_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{helica
l_fwd}}++;
            }
         } else {
            $type = "unknown4hit";
            $found_four_unknown++;
         }
         $comment .= "type: ${type} ";
      }
      ## now we look at the same thing as above, but for reverse molecules
      if ($observed_indices == 4 && $observe{stepsynth_rev} > 0 && $observe{helical_rev} > 0 &&
            $observe{stepcyc_rev} > 0 && $observe{variable_rev} > 0) {
         $rev_valid = 1;
         $found_all_four++;
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir)  = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'rev') {
               $helical = $name;
            } elsif ($piece eq 'stepcyc' && $dir eq 'rev') {
               $stepcyc = $name;
            } elsif ($piece eq 'variable' && $dir eq 'rev') {
               $variable = $name
            } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
               $stepsynth = $name
            }
         }
         my $final_size = $options{spacer} + $helical + $stepsynth + $variable;
         $comment .= "final size: $final_size ";
         if ($positions{stepcyc_rev} > $positions{helical_rev} &&
               $positions{helical_rev} > $positions{variable_rev} &&
               $positions{variable_rev} > $positions{stepsynth_rev} &&
               $numbers{stepsynth_rev} == $numbers{stepcyc_rev}) {
            $type = "unimolecular";
            $found_four_uni++;
            if (!defined($unicyclized4_final_lengths{$final_size})) {
               $unicyclized4_final_lengths{$final_size} = 1;
            } else {
               $unicyclized4_final_lengths{$final_size}++;
            }
            if
(!defined($unicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$n
umbers{helical_rev}})) {

$unicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{h
elical_rev}} = 1;
            } else {

$unicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{h
elical_rev}}++;
```

```
            }
        } elsif ($positions{stepcyc_rev} > $positions{helical_rev} &&
                 $positions{helical_rev} > $positions{variable_rev} &&
                 $positions{variable_rev} > $positions{stepsynth_rev} &&
                 $numbers{stepsynth_rev} != $numbers{stepcyc_rev}) {
            $type = "bimolecular-4";
            $found_four_bi++;
            if (!defined($bicyclized4_final_lengths{$final_size})) {
               $bicyclized4_final_lengths{$final_size} = 1;
            } else {
               $bicyclized4_final_lengths{$final_size}++;
            }
            if
(!defined($bicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$nu
mbers{helical_rev}})) {

$bicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}} = 1;
            } else {

$bicyclized4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}}++;
            }
        } elsif ($positions{helical_rev} > $positions{variable_rev} &&
                 $positions{variable_rev} > $positions{stepsynth_rev} &&
                 $positions{stepsynth_rev} > $positions{stepcyc_rev}) {
            $type = "linear4";
            $found_four_lin++;
            if (!defined($linear4_final_lengths{$final_size})) {
               $linear4_final_lengths{$final_size} = 1;
            } else {
               $linear4_final_lengths{$final_size}++;
            }
            if
(!defined($linear4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbe
rs{helical_rev}})) {

$linear4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{helical
_rev}} = 1;
            } else {

$linear4_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{helical
_rev}}++;
            }
        } else {
            $type = "unknown4hit";
            $found_four_unknown++;
        }
        $comment .= "type: ${type} ";
    }
    ## four hits that come purely from library fragments that did not
    ## have a live BstEII (synthesis junction) side
    if ($observed_indices == 4 && $observe{helical_fwd} > 0 && $observe{helical_rev} > 0 &&
$observe{variable_fwd} > 0 && $observe{variable_rev} > 0) {
        my @pieces = split(/\s+/, $comment);
```

```
        for my $p (@pieces) {
           my ($position, $piece, $name, $dir) = split(/:/, $p);
           if ($piece eq 'helical' && $dir eq 'fwd') {
              $helicalfwd = $name;
           } elsif ($piece eq 'helical' && $dir eq 'rev') {
              $helicalrev = $name;
           } elsif ($piece eq 'variable' && $dir eq 'fwd') {
              $variablefwd = $name;
           } elsif ($piece eq 'variable' && $dir eq 'rev') {
              $variablerev = $name;
           }
        }
        my $var_fwd_size = $helicalfwd + $variablefwd;
        my $var_rev_size = $helicalrev + $variablerev;
        if ($positions{variable_rev} < $positions{helical_rev} && $positions{helical_rev} <
$positions{helical_fwd} && $positions{helical_fwd} <$positions{variable_fwd}) {
           $type = "bicyc4varlib";
           $found_four_varlib_bi++;
           if ($var_fwd_size < 10) {
              if (!defined($bicyclized4_var_lib_final_lengths{'00'.$var_fwd_size})) {
                 $bicyclized4_var_lib_final_lengths{'00'.$var_fwd_size} = 1;
              } else {
                 $bicyclized4_var_lib_final_lengths{'00'.$var_fwd_size}++;
              }
           } else {
              if (!defined($bicyclized4_var_lib_final_lengths{'0'.$var_fwd_size})) {
                 $bicyclized4_var_lib_final_lengths{'0'.$var_fwd_size} = 1;
              } else {
                 $bicyclized4_var_lib_final_lengths{'0'.$var_fwd_size}++;
              }
           }
           if ($var_rev_size < 10) {
              if (!defined($bicyclized4_var_lib_final_lengths{'00'.$var_rev_size})) {
                 $bicyclized4_var_lib_final_lengths{'00'.$var_rev_size} = 1;
              } else {
                 $bicyclized4_var_lib_final_lengths{'00'.$var_rev_size}++;
              }
           } else {
              if (!defined($bicyclized4_var_lib_final_lengths{'0'.$var_rev_size})) {
                 $bicyclized4_var_lib_final_lengths{'0'.$var_rev_size} = 1;
              } else {
                 $bicyclized4_var_lib_final_lengths{'0'.$var_rev_size}++;
              }
           }
           if
(!defined($bicyclized4_var_lib_full_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd
}})) {

$bicyclized4_var_lib_full_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}} = 1;
           } else {

$bicyclized4_var_lib_full_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}}++;
           }
```

```
            if
(!defined($bicyclized4_var_lib_full_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}}
)) {

$bicyclized4_var_lib_full_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}} = 1;
            } else {

$bicyclized4_var_lib_full_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}}++;
            }
            $comment .= "fwd size: $var_fwd_size rev size: $var_rev_size ";
          } else {
            $type = "unknown4hit";
            $found_four_unknown++;
          }
          $comment .= "$type: ${type} ";
        }
        if ($observed_indices == 4 && $observe{stepsynth_fwd} > 0 && $observe{stepsynth_rev} > 0
&& $observe{stepcyc_fwd} > 0 && $observe{stepcyc_fwd} > 0 && $numbers{stepsynth_fwd} ==
$numbers{stepcyc_fwd} && $numbers{stepsynth_rev} == $numbers{stepcyc_rev}) {
          my @pieces = split(/\s+/, $comment);
          for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'stepsynth' && $dir eq 'fwd') {
              $stepsynthfwd = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
              $stepsynthrev = $name;
            } elsif ($piece eq 'stepcyc' && $dir eq 'fwd') {
              $stepcycfwd = $name;
            } elsif ($piece eq 'stepcyc' && $dir eq 'rev') {
              $stepcycrev = $name;
            }
          }
          my $step_fwd_size = $stepsynthfwd;
          my $step_rev_size = $stepsynthrev;
          if ($positions{stepsynth_fwd} < $positions{stepcyc_fwd} && $positions{stepcyc_fwd} <
$positions{stepcyc_rev} && $positions{stepcyc_rev} <$positions{stepsynth_rev}) {
            $type = "bicyc4steplib";
            $found_four_steplib_bi++;
            if (!defined($bicyclized4_step_lib_final_lengths{$step_fwd_size})) {
              $bicyclized4_step_lib_final_lengths{$step_fwd_size} = 1;
            } else {
              $bicyclized4_step_lib_final_lengths{$step_fwd_size}++;
            }
            if (!defined($bicyclized4_step_lib_final_lengths{$step_rev_size})) {
              $bicyclized4_step_lib_final_lengths{$step_rev_size} = 1;
            } else {
              $bicyclized4_step_lib_final_lengths{$step_rev_size}++;
            }
            if
(!defined($bicyclized4_step_lib_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{stepsynth
_rev}})) {

$bicyclized4_step_lib_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{stepsynth_rev}} =
1;
            } else {
```

```
$bicyclized4_step_lib_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{stepsynth_rev}}++;
            }
            $comment .= "fwd size: $step_fwd_size rev size: $step_rev_size ";
         } else {
            $type = "unknown4hit";
            $found_four_unknown++;
         }
         $comment .= "$type: ${type} ";
      }
      if ($observed_indices == 3 && $observe{helical_fwd} > 0 && $observe{variable_fwd} > 0 &&
$observe{stepsynth_fwd} > 0) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'fwd') {
               $helicalfwd = $name;
            } elsif ($piece eq 'variable' && $dir eq 'fwd') {
               $variablefwd = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'fwd') {
               $stepsynthfwd = $name;
            }
         }
         my $final_size = $options{spacer} + $helicalfwd + $stepsynthfwd + $variablefwd;
         $comment .= "final size: $final_size ";
         if ($positions{helical_fwd} < $positions{variable_fwd} &&
            $positions{variable_fwd} < $positions{stepsynth_fwd}) {
            $found_three_lin++;
            $type = "linear-3";
            if (!defined($linear3_final_lengths{$final_size})) {
               $linear3_final_lengths{$final_size} = 1;
            } else {
               $linear3_final_lengths{$final_size}++;
            }
            if
(!defined($linear3_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numb
ers{helical_fwd}})) {

$linear3_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{helica
l_fwd}} = 1;
            } else {

$linear3_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{helica
l_fwd}}++;
            }
         } else {
            $type = "unknown3hit";
            $found_three_unknown++;
         }
         $comment .= "type: ${type} ";
      }
      if ($observed_indices == 3 && $observe{helical_rev} > 0 && $observe{variable_rev} > 0 &&
$observe{stepsynth_rev} > 0) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
```

```
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'rev') {
               $helicalrev = $name;
            } elsif ($piece eq 'variable' && $dir eq 'rev') {
               $variablerev = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
               $stepsynthrev = $name;
            }
         }
         my $final_size = $options{spacer} + $helicalrev + $stepsynthrev + $variablerev;
         $comment .= "final size: $final_size ";
         if ($positions{helical_rev} > $positions{variable_rev} &&
            $positions{variable_rev} > $positions{stepsynth_rev}) {
            $found_three_lin++;
            $type = "linear-3";
            if (!defined($linear3_final_lengths{$final_size})) {
               $linear3_final_lengths{$final_size} = 1;
            } else {
               $linear3_final_lengths{$final_size}++;
            }
            if
(!defined($linear3_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbe
rs{helical_rev}})) {

$linear3_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{helical
_rev}} = 1;
            } else {

$linear3_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{helical
_rev}}++;
            }
         } else {
            $type = "unknown3hit";
            $found_three_unknown++;
         }
         $comment .= "type: ${type} ";
      }
      if ($observed_indices == 1 && $observe{stepcyc_fwd} > 0) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'stepcyc' && $dir eq 'fwd') {
               $stepcycfwd = $name;
            }
         }
         my $final_size = $stepcycfwd;
         $comment .= "final size: $final_size ";
         $found_one_lin_cyc++;
         $type = "linear-1-cyc";
         if (!defined($linear1_final_lengths_cyc{$final_size})) {
            $linear1_final_lengths_cyc{$final_size} = 1;
         } else {
            $linear1_final_lengths_cyc{$final_size}++;
         }
         $comment .= "type: ${type} ";
```

```
}
if ($observed_indices == 1 && $observe{stepcyc_rev} > 0) {
    my @pieces = split(/\s+/, $comment);
    for my $p (@pieces) {
        my ($position, $piece, $name, $dir) = split(/:/, $p);
        if ($piece eq 'stepcyc' && $dir eq 'rev') {
            $stepcycrev = $name;
        }
    }
    my $final_size = $stepcycrev;
    $comment .= "final size: $final_size ";
    $found_one_lin_cyc++;
    $type = "linear-1-cyc";
    if (!defined($linear1_final_lengths_cyc{$final_size})) {
        $linear1_final_lengths_cyc{$final_size} = 1;
    } else {
        $linear1_final_lengths_cyc{$final_size}++;
    }
    $comment .= "type: ${type} ";
}
if ($observed_indices == 1 && $observe{stepsynth_fwd} > 0) {
    my @pieces = split(/\s+/, $comment);
    for my $p (@pieces) {
        my ($position, $piece, $name, $dir) = split(/:/, $p);
        if ($piece eq 'stepsynth' && $dir eq 'fwd') {
            $stepsynthfwd = $name;
        }
    }
    my $final_size = $stepsynthfwd;
    $comment .= "final size: $final_size ";
    $found_one_lin_synth++;
    $type = "linear-1-synth";
    if (!defined($linear1_final_lengths_synth{$final_size})) {
        $linear1_final_lengths_synth{$final_size} = 1;
    } else {
        $linear1_final_lengths_synth{$final_size}++;
    }
    $comment .= "type: ${type} ";
}
if ($observed_indices == 1 && $observe{stepsynth_rev} > 0) {
    my @pieces = split(/\s+/, $comment);
    for my $p (@pieces) {
        my ($position, $piece, $name, $dir) = split(/:/, $p);
        if ($piece eq 'stepsynth' && $dir eq 'rev') {
            $stepsynthrev = $name;
        }
    }
    my $final_size = $stepsynthrev;
    $comment .= "final size: $final_size ";
    $found_one_lin_synth++;
    $type = "linear-1-synth";
    if (!defined($linear1_final_lengths_synth{$final_size})) {
        $linear1_final_lengths_synth{$final_size} = 1;
    } else {
        $linear1_final_lengths_synth{$final_size}++;
```

```
        }
        $comment .= "type: ${type} ";
    }
    ## for 5 I am looking at a set with three and a fragment this can be:
    ## stepsynth_rev < variable_rev < helical_rev < helical_fwd < variable_fwd
    ##               variable_rev < helical_rev < helical_fwd < variable_fwd < stepsynth_fwd
    ## these should only exist in the event that a library variable fragment
    ## found a full molecule and multimerized
    if ($observed_indices == 5 && $observe{stepsynth_fwd} > 0 && $observe{helical_fwd} > 0
&&
        $observe{variable_fwd} > 0 && $observe{helical_rev} > 0 && $observe{variable_rev} >
0) {
        my @pieces = split(/\s+/, $comment);
        for my $p (@pieces) {
            my ($positions, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'fwd') {
                $helicalfwd = $name;
            } elsif ($piece eq 'helical' && $dir eq 'rev') {
                $helicalrev = $name;
            } elsif ($piece eq 'variable' && $dir eq 'fwd') {
                $variablefwd = $name;
            } elsif ($piece eq 'variable' && $dir eq 'rev') {
                $variablerev = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'fwd') {
                $stepsynthfwd = $name;
            }
        }
        my $final_bi5_fwd_size = $options{spacer} + $stepsynthfwd + $variablefwd + $helicalfwd;
        ## the reverse size is variable plus helical, but this
        ## gives a minimum "size" of "0" up to "40" - this is
        ## not technically the size, the size is actually 072-112
        # I have chosen to leave this as the size shown because it
        # reflects a fragment and I want it to be clear that these
        # fragments should not be capable of cyclization., in theory
        # only appearing in a dead linear multimer (so the size is
        # less relevant so much as knowing that a molecule participated
        # in generation of a multimer) and which particular molecule it was.
        my $final_bi5_rev_size = $variablerev + $helicalrev;
        if ($positions{variable_rev} < $positions{helical_rev} &&
            $positions{helical_rev} < $positions{helical_fwd} &&
            $positions{helical_fwd} < $positions{variable_fwd} &&
            $positions{variable_fwd} < $positions{stepsynth_fwd} ) {
            $found_five_bi++;
            $type = "bimolecular-5";
            $comment .= "fwd size: $final_bi5_fwd_size rev size: $final_bi5_rev_size ";
            if (!defined($bicyclized5_final_lengths{$final_bi5_fwd_size})) {
                $bicyclized5_final_lengths{$final_bi5_fwd_size} = 1;
            } else {
                $bicyclized5_final_lengths{$final_bi5_fwd_size}++;
            }
            # for the fragmented size I need it to have three
            # digits like the rest of the library, for values
            # 0..9 prepend two leading 0s, else 10..40 prepend
            # one leading zero.
            if ($final_bi5_rev_size < 10) {
```

```
            if (!defined($bicyclized5_final_lengths{'00'.$final_bi5_rev_size})) {
               $bicyclized5_final_lengths{'00'.$final_bi5_rev_size} = 1;
            } else {
               $bicyclized5_final_lengths{'00'.$final_bi5_rev_size}++;
            }
         } else {
            if (!defined($bicyclized5_final_lengths{'0'.$final_bi5_rev_size})) {
               $bicyclized5_final_lengths{'0'.$final_bi5_rev_size} = 1;
            } else {
               $bicyclized5_final_lengths{'0'.$final_bi5_rev_size}++;
            }
         }
         if
(!defined($bicyclized5_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$n
umbers{helical_fwd}})) {

$bicyclized5_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}} = 1;
         } else {

$bicyclized5_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}}++;
         }
         if
(!defined($bicyclized5_frag_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}})) {
            $bicyclized5_frag_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}} = 1;
         } else {
            $bicyclized5_frag_final_lengths{$numbers{variable_rev}.','.$numbers{helical_rev}}++;
         }
      } else {
         $type = "unknown5hit";
         $found_five_unknown++;
      }
      $comment .= "type: ${type} ";
   }
   if ($observed_indices == 5 && $observe{stepsynth_rev} > 0 && $observe{helical_fwd} > 0 &&
      $observe{variable_fwd} > 0 && $observe{helical_rev} > 0 && $observe{variable_rev} >
0) {
      my @pieces = split(/\s+/, $comment);
      for my $p (@pieces) {
         my ($positions, $piece, $name, $dir) = split(/:/, $p);
         if ($piece eq 'helical' && $dir eq 'fwd') {
            $helicalfwd = $name;
         } elsif ($piece eq 'helical' && $dir eq 'rev') {
            $helicalrev = $name;
         } elsif ($piece eq 'variable' && $dir eq 'fwd') {
            $variablefwd = $name;
         } elsif ($piece eq 'variable' && $dir eq 'rev') {
            $variablerev = $name;
         } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
            $stepsynthrev = $name;
         }
      }
      my $final_bi5_fwd_size = $variablefwd + $helicalfwd;
      my $final_bi5_rev_size = $options{spacer} + $variablerev + $helicalrev + $stepsynthrev;
```

```
        if ($positions{stepsynth_rev} < $positions{variable_rev} &&
            $positions{variable_rev} < $positions{helical_rev} &&
            $positions{helical_rev} < $positions{helical_fwd} &&
            $positions{helical_fwd} < $positions{variable_fwd}) {
          $found_five_bi++;
          $type = "bimolecular-5";
          $comment .= "fwd size: $final_bi5_fwd_size rev size: $final_bi5_rev_size ";
          if ($final_bi5_fwd_size < 10) {
            if (!defined($bicyclized5_final_lengths{'00'.$final_bi5_fwd_size})) {
              $bicyclized5_final_lengths{'00'.$final_bi5_fwd_size} = 1;
            } else {
              $bicyclized5_final_lengths{'00'.$final_bi5_fwd_size}++;
            }
          } else {
            if (!defined($bicyclized5_final_lengths{'0'.$final_bi5_fwd_size})) {
              $bicyclized5_final_lengths{'0'.$final_bi5_fwd_size} = 1;
            } else {
              $bicyclized5_final_lengths{'0'.$final_bi5_fwd_size}++;
            }
          }
          if (!defined($bicyclized5_final_lengths{$final_bi5_rev_size})) {
            $bicyclized5_final_lengths{$final_bi5_rev_size} = 1;
          } else {
            $bicyclized5_final_lengths{$final_bi5_rev_size}++;
          }
          if
(!defined($bicyclized5_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$nu
mbers{helical_rev}})) {

$bicyclized5_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}} = 1;
          } else {

$bicyclized5_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}}++;
          }
          if
(!defined($bicyclized5_frag_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}})) {
            $bicyclized5_frag_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}} =
1;
          } else {
            $bicyclized5_frag_final_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}}++;
          }
        } else {
          $type = "unknown5hit";
          $found_five_unknown++;
        }
        $comment .= "type: ${type} ";
      }
      ##
      if ($observed_indices == 6 && $observe{stepsynth_fwd} > 0 && $observe{helical_fwd} > 0
&&
          $observe{variable_fwd} > 0 && $observe{stepsynth_rev} > 0 && $observe{helical_rev} >
0 &&
          $observe{variable_rev} > 0) {
```

```
            my @pieces = split(/\s+/, $comment);
            for my $p (@pieces) {
               my ($position, $piece, $name, $dir) = split(/:/, $p);
               if ($piece eq 'helical' && $dir eq 'fwd') {
                  $helicalfwd = $name;
               } elsif ($piece eq 'helical' && $dir eq 'rev') {
                  $helicalrev = $name
               } elsif ($piece eq 'variable' && $dir eq 'fwd') {
                  $variablefwd = $name
               } elsif ($piece eq 'variable' && $dir eq 'rev') {
                  $variablerev = $name
               } elsif ($piece eq 'stepsynth' && $dir eq 'fwd') {
                  $stepsynthfwd = $name
               } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
                  $stepsynthrev = $name
               }
            }
            my $final_bi6_fwd_size = $options{spacer} + $stepsynthfwd + $variablefwd + $helicalfwd;
            my $final_bi6_rev_size = $options{spacer} + $stepsynthrev + $variablerev + $helicalrev;
            if ($positions{stepsynth_fwd} > $positions{variable_fwd} &&
                  $positions{variable_fwd} > $positions{helical_fwd} &&
                  $positions{helical_fwd} > $positions{helical_rev} &&
                  $positions{helical_rev} > $positions{variable_rev} &&
                  $positions{variable_rev} > $positions{stepsynth_rev} ) {
               $found_six_bi++;
               $type = "bimolecular-6";
               $comment .= "fwd size: $final_bi6_fwd_size rev size: $final_bi6_rev_size ";
               if (!defined($bicyclized6_final_lengths{$final_bi6_fwd_size})) {
                  $bicyclized6_final_lengths{$final_bi6_fwd_size} = 1;
               } else {
                  $bicyclized6_final_lengths{$final_bi6_fwd_size}++;
               }
               if (!defined($bicyclized6_final_lengths{$final_bi6_rev_size})) {
                  $bicyclized6_final_lengths{$final_bi6_rev_size} = 1;
               } else {
                  $bicyclized6_final_lengths{$final_bi6_rev_size}++;
               }
               if
(!defined($bicyclized6_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$nu
mbers{helical_rev}})) {

$bicyclized6_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}} = 1;
               } else {

$bicyclized6_full_final_lengths{$numbers{stepsynth_rev}.','.$numbers{variable_rev}.','.$numbers{he
lical_rev}}++;
               }
               if
(!defined($bicyclized6_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$n
umbers{helical_fwd}})) {

$bicyclized6_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}} = 1;
               } else {
```

```
$bicyclized6_full_final_lengths{$numbers{stepsynth_fwd}.','.$numbers{variable_fwd}.','.$numbers{h
elical_fwd}}++;
            }
         } else {
            $type = "unknown6hit";
            $found_six_unknown++;
         }
         $comment .= "type: ${type} ";
      }
      ##
      if ($observed_indices == 2 && $observe{stepcyc_fwd} > 0 && $observe{stepcyc_rev} > 0) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'stepcyc' && $dir eq 'fwd') {
               $stepcycfwd = $name;
            } elsif ($piece eq 'stepcyc' && $dir eq 'rev') {
               $stepcycrev = $name;
            }
         }
         my $final_bi2_size = $stepcycfwd + $stepcycrev;
         if ($positions{stepcyc_fwd} < $positions{stepcyc_rev}) {
            $found_two_bi++;
            $type = "bimolecular-2";
            $comment .= "final size: $final_bi2_size ";
            # truthfully the final size here is useless for
            # our current library size given the separated
            # component counting below. I am leaving it
            # in case it becomes useful for future projects
            if (!defined($bicyclized2_final_lengths{$final_bi2_size})) {
               $bicyclized2_final_lengths{$final_bi2_size} = 1;
            } else {
               $bicyclized2_final_lengths{$final_bi2_size}++;
            }
            if
(!defined($bicyclized2_full_final_lengths{$numbers{stepcyc_fwd}.','.$numbers{stepcyc_rev}})) {
               $bicyclized2_full_final_lengths{$numbers{stepcyc_fwd}.','.$numbers{stepcyc_rev}} = 1;
            } else {
               $bicyclized2_full_final_lengths{$numbers{stepcyc_fwd}.','.$numbers{stepcyc_rev}}++;
            }
         } else {
            $type = "unknown2hit";
         }
         $comment .= "type: ${type} ";
      }
      if ($observed_indices == 2 && $observe{stepcyc_fwd} > 0 && $observe{stepsynth_fwd} > 0
&& $numbers{stepcyc_fwd} == $numbers{stepsynth_fwd} ) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'stepcyc' && $dir eq 'fwd') {
               $stepcycfwd = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'fwd') {
               $stepsynthfwd = $name;
```

```
            }
        }
        my $final_step_component = $stepcycfwd;
        if ($positions{stepcyc_fwd} > $positions{stepsynth_fwd}) {
            $found_two_lib_step++;
            $type = "Step-Component-2";
            $comment .= "component size: $final_step_component ";
            if (!defined($bi2_full_step_lengths{$numbers{stepcyc_fwd}})) {
                $bi2_full_step_lengths{$numbers{stepcyc_fwd}} = 1;
            } else {
                $bi2_full_step_lengths{$numbers{stepcyc_fwd}}++;
            }
        } else {
            $type = "unknown2hit";
        }
        $comment .= "type: ${type} ";
    }
    if ($observed_indices == 2 && $observe{stepcyc_rev} > 0 && $observe{stepsynth_rev} > 0 &&
$numbers{stepcyc_rev} == $numbers{stepsynth_rev} ) {
        my @pieces = split(/\s+/, $comment);
        for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'stepcyc' && $dir eq 'rev') {
                $stepcycrev = $name;
            } elsif ($piece eq 'stepsynth' && $dir eq 'rev') {
                $stepsynthrev = $name;
            }
        }
        my $final_step_component = $stepcycrev;
        if ($positions{stepcyc_rev} > $positions{stepsynth_rev}) {
            $found_two_lib_step++;
            $type = "Step-Component-2";
            $comment .= "component size: $final_step_component ";
            if (!defined($bi2_full_step_lengths{$numbers{stepcyc_rev}})) {
                $bi2_full_step_lengths{$numbers{stepcyc_rev}} = 1;
            } else {
                $bi2_full_step_lengths{$numbers{stepcyc_rev}}++;
            }
        } else {
            $type = "unknown2hit";
        }
        $comment .= "type: ${type} ";
    }
    if ($observed_indices == 2 && $observe{helical_fwd} > 0 && $observe{variable_fwd} > 0) {
        my @pieces = split(/\s+/, $comment);
        for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'fwd') {
                $helicalfwd = $name;
            } elsif ($piece eq 'variable' && $dir eq 'fwd') {
                $variablefwd = $name;
            }
        }
        my $final_bi2_lib_size = $helicalfwd + $variablefwd;
        if ($positions{helical_fwd} < $positions{variable_fwd}) {
```

```
            $found_two_lib_var++;
            $type = "Variable-Component-2";
            $comment .= "component size: $final_bi2_lib_size ";
            if (!defined($bi2_full_lib_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}})) {
               $bi2_full_lib_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}} = 1;
            } else {
               $bi2_full_lib_lengths{$numbers{variable_fwd}.','.$numbers{helical_fwd}}++;
            }
         } else {
            $type = "unknown2hit";
         }
         $comment .= "type: ${type} ";
      }
      if ($observed_indices == 2 && $observe{helical_rev} > 0 && $observe{variable_rev} > 0) {
         my @pieces = split(/\s+/, $comment);
         for my $p (@pieces) {
            my ($position, $piece, $name, $dir) = split(/:/, $p);
            if ($piece eq 'helical' && $dir eq 'rev') {
               $helicalrev = $name;
            } elsif ($piece eq 'variable' && $dir eq 'rev') {
               $variablerev = $name;
            }
         }
         my $final_bi2_lib_size = $helicalrev + $variablerev;
         if ($positions{helical_rev} > $positions{variable_rev}) {
            $found_two_lib_var++;
            $type = "Variable-Component-2";
            $comment .= "component size: $final_bi2_lib_size ";
            if (!defined($bi2_full_lib_lengths{$numbers{variable_rev}.','.$numbers{helical_fwd}})) {
               $bi2_full_lib_lengths{$numbers{variable_rev}.','.$numbers{helical_fwd}} = 1;
            } else{
               $bi2_full_lib_lengths{$numbers{variable_rev}.','.$numbers{helical_fwd}}++;
            }
         } else {
            $type = "unknown2hit";
         }
         $comment .= "type: ${type} ";
      }
      $comment .= "Count: $count  hits: ${observed_indices} ";
      if ($observed_indices == 1) {
         $singles{sum}++;
         foreach my $k (keys %observe) {
            $singles{$k} += $observe{$k};
         }
      } elsif ($observed_indices == 2) {
         $doubles{sum}++;
         foreach my $k (keys %observe) {
            $doubles{$k} += $observe{$k};
         }
      } elsif ($observed_indices == 3) {
         $triples{sum}++;
         foreach my $k (keys %observe) {
            $triples{$k} += $observe{$k};
         }
      } elsif ($observed_indices == 4) {
```

```
            $quads{sum}++;
            foreach my $k (keys %observe) {
               $quads{$k} += $observe{$k};
            }
         } elsif ($observed_indices == 5) {
            $fives{sum}++;
            foreach my $k (keys %observe) {
               $fives{$k} += $observe{$k};
            }
         } elsif ($observed_indices == 6) {
            $sixes{sum}++;
            foreach my $k (keys %observe) {
               $sixes{$k} += $observe{$k};
            }
         } elsif ($observed_indices >= 7) {
            $sevenup{sum}++;
            foreach my $k (keys %observe) {
               $sevenup{$k} += $observe{$k};
            }
         }
         my $fastq_string = qq"\@${id}
${sequence}
+${comment}
${qual}
";
         print $out $fastq_string;
      }
      return($data);
   }
   sub Sort_Dir {
      my %args = @_;
      my $cwd_dir = getcwd();
      my $searchdir = qq"$args{indir}";
      my $files = 0;
      unless ($searchdir =~ /^\//) {
         $searchdir = qq"${cwd_dir}/${searchdir}";
      }
      my @directory = ($searchdir);
      my @file_list = ();
      find(sub { push(@file_list, $File::Find::name)
                  if ($File::Find::name =~ /\.fastq\.gz/ and
                     $File::Find::name !~ /$args{outdir}/); }, @directory);
      my @approxes = ();
      foreach my $file (@file_list) {
         $files = $files++;
         next if ($file =~ /$options{outdir}/);
         $file =~ s/\/\.\/\//;
         my $approx = Sort_File_Approx(
            input => $file,
            outdir => $args{outdir},
            index_hash => $args{index_hash},
         );
         push(@approxes, $approx);
      }
      return(@approxes);
```

```
}

sub Read_indices {
   my %args = @_;

   my $indices = {
      total => {},
      unknown => {},
      possibilities => [],
   };

   $indices->{total} = {
      read => 0, written => 0, };
   $indices->{unknown} = {
      name => 'start_unknown', written => 0, };

   my $index_file = FileHandle->new("<$options{indices}");
   while (my $line = <$index_file>) {
      chomp $line;
      next if ($line =~ /^#/);
      next unless ($line =~ /^A|T|G|C|a|t|g|c/);

      my ($index_sequence, $phrase) = split(/\s+|\,|;/, $line);
      my ($name, $number, $direction) = split(/_/, $phrase);
      $indices->{$index_sequence} = {
         name => $name,
         direction => $direction,
         # I am formating each number to include three digits
         # this is because my greatest size is three digits and
         # I want everything to include all three (even if all zeros).
         number => $number,
         # original number input converted number to integer
         # number => int($number),
         total_observed => 0,
         ambiguous_observed => 0,
         unique_observed => 0
      };

      my @pos = @{$indices->{possibilities}};
      push(@pos, $index_sequence);
      $indices->{possibilities} = \@pos;
   }
   $index_file->close();
   return($indices);
}
## Here we create the overall summary file
sub End_Handler {
   print $log "Index used: $options{indices}\n";
   print $log "Input file: $options{input}\n";
   print $log "Date and time run: $datestring\n";
   if ( !defined $options{insertion} && !defined $options{deletion} &&
         !defined $options{substitution} ) {
      print $log "Direct match used";
   } else {
```

```
        print $log "String::Approx matching used:
I$options{insertion},D$options{deletion},S$options{substitution}\n";
    }
    print $log "Summary of index hits and full matches below:\n";
    print $log "${observed_reads} reads were observed in total, of these:\n";
    foreach my $k (sort keys %observations) {
        if ($observations{$k} > 0 and $k ne 'sum') {
            print $log "     The read type: ${k} was observed: $observations{$k} times.\n";
        }
    }
    if ($singles{sum} > 0) {
        print $log "$singles{sum} single-index reads were observed, including:\n";
        foreach my $k (sort keys %singles) {
            if ($singles{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $singles{$k} times.\n";
            }
        }
    }
    if ($doubles{sum} > 0) {
        print $log "$doubles{sum} double-index reads were observed, including:\n";
        foreach my $k (sort keys %doubles) {
            if ($doubles{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $doubles{$k} times.\n";
            }
        }
    }
    if ($triples{sum} > 0) {
        print $log "$triples{sum} triple-index reads were observed, including:\n";
        foreach my $k (sort keys %triples) {
            if ($triples{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $triples{$k} times.\n";
            }
        }
    }
    if ($quads{sum} > 0) {
        print $log "$quads{sum} 4-index reads were observed, including:\n";
        foreach my $k (sort keys %quads) {
            if ($quads{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $quads{$k} times.\n";
            }
        }
    }
    if ($fives{sum} > 0) {
        print $log "$fives{sum} 5-index reads were observed, including:\n";
        foreach my $k (sort keys %fives) {
            if ($fives{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $fives{$k} times.\n";
            }
        }
    }
    if ($sixes{sum} > 0) {
        print $log "$sixes{sum} 6-index reads were observed, including:\n";
        foreach my $k (sort keys %sixes) {
            if ($sixes{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $sixes{$k} times.\n";
```

```
            }
        }
    }
    if ($sevenup{sum} > 0) {
        print $log "$sevenup{sum} 7 or more-index reads were observed, including:\n";
        foreach my $k (sort keys %sevenup) {
            if ($sevenup{$k} > 0 and $k ne 'sum') {
                print $log "     The read type: ${k} was observed: $sevenup{$k} times.\n";
            }
        }
    }
    print $log "\n";
    print $log "${found_one_lin_cyc} reads had a single hit and were stepcyc linear fragments\n";
    foreach my $k (sort keys %linear1_final_lengths_cyc) {
        print $onehitlincyc_csv "$k,$linear1_final_lengths_cyc{$k}\n";
    }
    print $log "${found_one_lin_synth} reads had a single hit and were stepsynth linear fragments\n";
    foreach my $k (sort keys %linear1_final_lengths_synth) {
        print $onehitlinsynth_csv "$k,$linear1_final_lengths_synth{$k}\n";
    }
    print $log "\n";
    print $log "${found_two_bi} reads had two hits and were bimolecular B-B\n";
    foreach my $k (sort keys %bicyclized2_final_lengths) {
        print $bicyc2_csv "$k,$bicyclized2_final_lengths{$k}\n";
    }
    foreach my $k (sort keys %bicyclized2_full_final_lengths) {
        print $bicyc2full_csv "$k,$bicyclized2_full_final_lengths{$k}\n";
    }
    print $log "${found_two_lib_var} reads had two hits and were library variable fragments\n";
    foreach my $k (sort keys %bi2_full_lib_lengths) {
        print $varlibfull_csv "$k,$bi2_full_lib_lengths{$k}\n";
    }
    print $log "${found_two_lib_step} reads had two hits and were library step fragments\n";
    foreach my $k (sort keys %bi2_full_step_lengths) {
        print $steplibfull_csv "$k,$bi2_full_step_lengths{$k}\n";
    }
    print $log "\n";
    print $log "${found_three_lin} reads had three hits and were linear fragments
stepsynth+variable+helical\n";
    foreach my $k (sort keys %linear3_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $threehitlin_csv "$k,$linear3_final_lengths{$k}\n";
        }
    }
    foreach my $k (sort keys %linear3_full_final_lengths) {
        print $threehitlinfull_csv "$k,$linear3_full_final_lengths{$k}\n";
    }
    print $log "\n";
    print $log "${found_four_uni} reads had four hits and were cyclized unimolecular\n";
    foreach my $k (sort keys %unicyclized4_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $unicyc_csv "$k,$unicyclized4_final_lengths{$k}\n";
        }
    }
    foreach my $k (sort keys %unicyclized4_full_final_lengths) {
```

```
        print $unicycfull_csv "$k,$unicyclized4_full_final_lengths{$k}\n";
    }
    print $log "${found_four_lin} reads had four hits and were linear library molecules\n";
    foreach my $k (sort keys %linear4_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $fourhitlin_csv "$k,$linear4_final_lengths{$k}\n";
        }
    }
    foreach my $k (sort keys %linear4_full_final_lengths) {
        print $fourhitlinfull_csv "$k,$linear4_full_final_lengths{$k}\n";
    }
    print $log "${found_four_bi} reads had four hits and were biomolecular A-B\n";
    foreach my $k (sort keys %bicyclized4_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $bicyc4_csv "$k,$bicyclized4_final_lengths{$k}\n";
        }
    }
    foreach my $k (sort keys %bicyclized4_full_final_lengths) {
        print $bicyc4full_csv "$k,$bicyclized4_full_final_lengths{$k}\n";
    }
    print $log "${found_four_varlib_bi} reads had four hits and were bimolecular A-A library
fragments\n";
    foreach my $k (sort keys %bicyclized4_var_lib_final_lengths) {
        print $bicyc4varlib_csv "$k,$bicyclized4_var_lib_final_lengths{$k}\n";
    }
    foreach my $k (sort keys %bicyclized4_var_lib_full_final_lengths) {
        print $bicyc4varlibfull_csv "$k,$bicyclized4_var_lib_full_final_lengths{$k}\n";
    }
    print $log "${found_four_steplib_bi} reads had four hits and were bimolecular B-B library
fragments\n";
    foreach my $k (sort keys %bicyclized4_step_lib_final_lengths) {
        print $bicyc4steplib_csv "$k,$bicyclized4_step_lib_final_lengths{$k}\n";
    }
    foreach my $k (sort keys %bicyclized4_step_lib_full_final_lengths) {
        print $bicyc4steplibfull_csv "$k,$bicyclized4_step_lib_full_final_lengths{$k}\n";
    }
    print $log "\n";
    print $log "${found_five_bi} reads had five hits and were bimolecular A-A\n";
    foreach my $k (sort keys %bicyclized5_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $bicyc5_csv "$k,$bicyclized5_final_lengths{$k}\n";
        }
    }
    foreach my $k (sort keys %bicyclized5_full_final_lengths) {
        print $bicyc5full_csv "$k,$bicyclized5_full_final_lengths{$k}\n";
    }
    foreach my $k (sort keys %bicyclized5_frag_final_lengths) {
        print $bicyc5frag_csv "$k,$bicyclized5_frag_final_lengths{$k}\n";
    }
    print $log "\n";
    print $log "${found_six_bi} reads had six hits and were bimolecular A-A\n";
    foreach my $k (sort keys %bicyclized6_final_lengths) {
        if ($k ne "$options{spacer}") {
            print $bicyc6_csv "$k,$bicyclized6_final_lengths{$k}\n";
        }
```

```
    }
    foreach my $k (sort keys %bicyclized6_full_final_lengths) {
        print $bicyc6full_csv "$k,$bicyclized6_full_final_lengths{$k}\n";
    }
    $log->close();
    $unicyc_csv->close();
    $unicycfull_csv->close();
    $fourhitlin_csv->close();
    $fourhitlinfull_csv->close();
    $threehitlin_csv->close();
    $threehitlinfull_csv->close();
    $onehitlincyc_csv->close();
    $onehitlinsynth_csv->close();
    $bicyc4_csv->close();
    $bicyc4full_csv->close();
    $bicyc4varlib_csv->close();
    $bicyc4varlibfull_csv->close();
    $bicyc4steplib_csv->close();
    $bicyc4steplibfull_csv->close();
    $bicyc6_csv->close();
    $bicyc6full_csv->close();
    $bicyc5_csv->close();
    $bicyc5full_csv->close();
    $bicyc5frag_csv->close();
    $bicyc2_csv->close();
    $bicyc2full_csv->close();
    $varlibfull_csv->close();
    $steplibfull_csv->close();
    $out->close();
    exit(0);
}

sub aindexes {
    my ($index, $params, $sequence) = @_;
    my @return = ();
    my $continue = 1;
    while ($continue) {
        my @ind = aindex($index, $params, ($sequence));
        my $res = $ind[0];
        if ($res == -1) {
            $continue = 0;
        } else {
            push(@return, $res);
            my $end_position = $res + length($index);
            $sequence = substr($sequence, $end_position);
        }
    }
    return(@return);
}
```

Appendix 5.2: Index Sequences

Each index is a sequence followed by a 3 word phrase: region_size_direction

These indices are pulled into the code from appendix 5.1. The indices printed here were used at the time final assignments for this dissertation were performed. The index file is part of the scripts full GitHub package and any updates can be found at:

https://github.com/jhustedt/kahn-ngs

Appendix 5.2.1: Variable Sequence Side 0-10 Indices

```
CTCGGAATGGTAGCTCTCGATGATGC  helical_10_fwd
CTCGGAATGGTACCTTCGATGATGC  helical_09_fwd
CTCGGAATGGTAACTCGATGATGC  helical_08_fwd
CTCGGAATGCTATTCGATGATGC  helical_07_fwd
CTCGGACTGGTATCGATGATGC  helical_06_fwd
CTCGGACTGCTTCGATGATGC  helical_05_fwd
CTCGGACTGATCGATGATGC  helical_04_fwd
CTCGGATGTTCGATGATGC  helical_03_fwd
CTCGGAGTTCGATGATGC  helical_02_fwd
CTCGGACTCGATGATGC  helical_01_fwd
CTCGGATCGATGATGC  helical_00_fwd
GCATCATCGAGAGCTACCATTCCGAG  helical_10_rev
GCATCATCGAAGGTACCATTCCGAG  helical_09_rev
GCATCATCGAGTTACCATTCCGAG  helical_08_rev
GCATCATCGAATAGCATTCCGAG  helical_07_rev
GCATCATCGATACCAGTCCGAG  helical_06_rev
GCATCATCGAAGCAGTCCGAG  helical_05_rev
GCATCATCGATCAGTCCGAG  helical_04_rev
GCATCATCGAACATCCGAG  helical_03_rev
GCATCATCGAACTCCGAG  helical_02_rev
GCATCATCGAGTCCGAG  helical_01_rev
GCATCATCGATCCGAG  helical_00_rev
```

Appendix 5.2.2: Variable Sequence Side 0-30 Indices

```
ACCCATTAGGTCCAGCAGGCATTGCAGTCAGACGCAGTTGA  variable_30_rev
ACCCATTAGCTCCAGCAGGCATTGCAGTCAGAGCAGTTGA  variable_29_rev
ACCCATTAGATCCAGCAGGCATTGCAGTCAGGCAGTTGA  variable_28_rev
ACCCATGAGTTCCAGCAGGCATTGCAGTCAGCAGTTGA  variable_27_rev
ACCCATGAGGTCCAGCAGGCATTGCAGTCGCAGTTGA  variable_26_rev
ACCCATGAGCTCCAGCAGGCATTGCAGTGCAGTTGA  variable_25_rev
ACCCATGAGATCCAGCAGGCATTGCAGGCAGTTGA  variable_24_rev
ACCCATCAGTTCCAGCAGGCATTGCAGCAGTTGA  variable_23_rev
ACCCATCAGGTCCAGCAGGCATTGCGCAGTTGA  variable_22_rev
ACCCATCAGCTCCAGCAGGCATTGGCAGTTGA  variable_21_rev
ACCCATCAGATCCAGCAGGCATTGCAGTTGA  variable_20_rev
ACCCATAAGTTCCAGCAGGCATGCAGTTGA  variable_19_rev
ACCCATAAGGTCCAGCAGGCAGCAGTTGA  variable_18_rev
ACCCATAAGCTCCAGCAGGCGCAGTTGA  variable_17_rev
ACCCATAAGATCCAGCAGGGCAGTTGA  variable_16_rev
ACCAATTAGTTCCAGCAGGCAGTTGA  variable_15_rev
ACCAATTAGGTCCAGCAGCAGTTGA  variable_14_rev
```

```
ACCAATTAGCTCCAGCGCAGTTGA  variable_13_rev
ACCAATTAGATCCAGGCAGTTGA  variable_12_rev
ACCAATGAGTTCCAGCAGTTGA  variable_11_rev
ACCAATGAGGTCCGCAGTTGA  variable_10_rev
ACCAATGAGCTCGCAGTTGAG  variable_09_rev
ACCAATGAGATGCAGTTGAGG  variable_08_rev
ACCAATCAGTGCAGTTGAGGT  variable_07_rev
ACCCATGAGGCAGTTGAGGTG  variable_06_rev
CACCCATCAGCAGTTGAGGTG  variable_05_rev
TCACCCATAGCAGTTGAGGTG  variable_04_rev
GTCACCTATGCAGTTGAGGTG  variable_03_rev
GTCACCGAGCAGTTGAGGTG  variable_02_rev
GTCACCCGCAGTTGAGGTG  variable_01_rev
GTCACCGCAGTTGAGGTG  variable_00_rev
TCAACTGCGTCTGACTGCAATGCCTGCTGGACCTAATGGGT  variable_30_fwd
TCAACTGCTCTGACTGCAATGCCTGCTGGAGCTAATGGGT  variable_29_fwd
TCAACTGCCTGACTGCAATGCCTGCTGGATCTAATGGGT  variable_28_fwd
TCAACTGCTGACTGCAATGCCTGCTGGAACTCATGGGT  variable_27_fwd
TCAACTGCGACTGCAATGCCTGCTGGACCTCATGGGT  variable_26_fwd
TCAACTGCACTGCAATGCCTGCTGGAGCTCATGGGT  variable_25_fwd
TCAACTGCCTGCAATGCCTGCTGGATCTCATGGGT  variable_24_fwd
TCAACTGCTGCAATGCCTGCTGGAACTGATGGGT  variable_23_fwd
TCAACTGCGCAATGCCTGCTGGACCTGATGGGT  variable_22_fwd
TCAACTGCCAATGCCTGCTGGAGCTGATGGGT  variable_21_fwd
TCAACTGCAATGCCTGCTGGATCTGATGGGT  variable_20_fwd
TCAACTGCATGCCTGCTGGAACTTATGGGT  variable_19_fwd
TCAACTGCTGCCTGCTGGACCTTATGGGT  variable_18_fwd
TCAACTGCGCCTGCTGGAGCTTATGGGT  variable_17_fwd
TCAACTGCCCTGCTGGATCTTATGGGT  variable_16_fwd
TCAACTGCCTGCTGGAACTAATTGGT  variable_15_fwd
TCAACTGCTGCTGGACCTAATTGGT  variable_14_fwd
TCAACTGCGCTGGAGCTAATTGGT  variable_13_fwd
TCAACTGCCTGGATCTAATTGGT  variable_12_fwd
TCAACTGCTGGAACTCATTGGT  variable_11_fwd
TCAACTGCGGACCTCATTGGT  variable_10_fwd
CTCAACTGCGAGCTCATTGGT  variable_09_fwd
CCTCAACTGCATCTCATTGGT  variable_08_fwd
ACCTCAACTGCACTGATTGGT  variable_07_fwd
CACCTCAACTGCCTCATGGGT  variable_06_fwd
CACCTCAACTGCTGATGGGTG  variable_05_fwd
CACCTCAACTGCTATGGGTGA  variable_04_fwd
CACCTCAACTGCATAGGTGAC  variable_03_fwd
CACCTCAACTGCTCGGTGAC  variable_02_fwd
CACCTCAACTGCGGGTGAC  variable_01_fwd
CACCTCAACTGCGGTGAC  variable_00_fwd
```

Appendix 5.2.3: Size Class Indices

Cyclization side (XhoI) indices

```
GAGATATACGAGGAG  stepcyc_047_fwd
CGAAGTGGTCTGCAA  stepcyc_077_fwd
CTCTTACTGTCATGC  stepcyc_107_fwd
CTCCTCGTATATCTC  stepcyc_047_rev
TTGCAGACCACTTCG  stepcyc_077_rev
GCATGACAGTAAGAG  stepcyc_107_rev
```

Synthesis side (BstEII) indices

```
CCTGACTCCCCGTCG  stepsynth_047_fwd
CAGCCGGAAGGCCGA  stepsynth_077_fwd
TTATGGCAGCACTGC  stepsynth_107_fwd
CGACGGGGAGTCAGG  stepsynth_047_rev
TCGGCCTTCCGGCTG  stepsynth_077_rev
GCAGTGCTGCCATAA  stepsynth_107_rev
```

Appendix 5.3: compete_cyc.m

The following code is written in MATLAB. Simulations in chapter 4 were performed using this code and MATLAB version R2019a.

```
function [t,concs] = compete_cyc(end_time,M_input_vec,kc,kb,varargin)
% Calculate amts of products in ligation reactions given the parameters and
%    monomer input concentrations
% The optional args in varargin are kcd_vec,dfA_vec,dfB_vec
% Time should be just an end time
% M_input_vec is a vector of input monomer concentrations
% kc is a single number or else a vector of the same length as M_input_vec
%    for unimolecular  cyclization rate constants
% kb is a single bimolecular ligation rate constant for all (I suppose one could elaborate to have
different
%    values for AA, AB, and BB ligations
% kcd_vec,dfA_vec,dfB_vec can be left blank, or they can be single numbers,
%    or vectors of the same length as M_input_vec
% kcd_vec is a vector of cyclization rate constants for dimer circles
% kcd_multi is the max of the kcd_vec
% dfA_vec is  a vector of the dead end fractions for the A end for each class
% similarly dfB_vec
% Returns a time vector and a matrix with the each column being the amt of
%    each monomer and product
% Example of use:
% time = (0:120)'
%df = 0;
%kc = 0.04;
%kb = 0.0008;
%M0 = 5;
%[products] = bicycle_deadends(time,kc,kb,df,M0);
%figure
%plot(time,products,'r','LineWidth',2)
%defaults
maxnumopts = 3;
num_sets = length(M_input_vec);
% Set defaults, but recognize that the kcd is probably strongly length
% dependent in interesting length ranges
kcd_num = 3*max(kc);
dfA_num = 0;
dfB_num = dfA_num;
optargs = {kcd_num,dfA_num,dfB_num};
%
numvarargs = length(varargin);
if numvarargs > maxnumopts
```

```
    error('compete_cyc:TooManyInputs', ...
        'compete_cyc allows at most %d optional inputs',maxnumopts);
end
%
% Now put the new input values from varargin into the optargs cell array,
% overwriting the default values.
aretheythere = cellfun(@isempty,varargin);
for ivarg = 1:length(varargin)
  if ~aretheythere(ivarg)
      optargs(ivarg) = varargin(ivarg);
  end
end
%optargs(1:numvarargs) = varargin;
[kcd_num,dfA_num,dfB_num] = optargs{:};
% Allow for all inputs to be either single numbers or vectors of the
% correct length.
if (length(kc) == num_sets)
    kc_vec = kc;
elseif (length(kc) == 1)
    kc_vec = ones(1,num_sets).*kc;
else
    error('compete_cyc: kc should have length 1 or %d = number of monomer sets',num_sets);
end
% In theory there are kcd's for each pair of sets, with the number of
% interclass pairs given by the pascal triangle number below
pascalp = [0 1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 155];
% etc. inter-pairings are triangular numbers from the Pascal triangle,
% Provide the kcd's in order 1-1 1-2 1-3 1-4...2-2 2-3 2-4...
% So we can be sloppy and just iterate through the kcd's in the loop below
if (length(kcd_num) == num_sets + pascalp(num_sets))
    kcd_vec = kcd_num;
elseif (length(kcd_num) == 1)
    kcd_vec = ones(1,(num_sets + pascalp(num_sets))).*kcd_num;
else
    error('compete_cyc: kcd should have length 1 or %d',num_sets + pascalp(num_sets));
end
kc_multi = max(kcd_vec);
if (length(dfA_num) == num_sets)
    dfA_vec = dfA_num;
    dfB_vec = dfA_num;
elseif (length(dfA_num) == 1)
    dfA_vec = ones(1,num_sets).*dfA_num;
    dfB_vec = dfA_vec;
else
    error('compete_cyc: dfA should have length 1 or %d',num_sets);
end
if numvarargs == 3
    if (length(dfB_num) == num_sets)
        dfB_vec = dfB_num;
    elseif (length(dfB_num) == 1)
        dfB_vec = ones(1,num_sets).*dfB_num;
else
    error('compete_cyc: dfB should have length 1 or %d',num_sets);
    end
end
```

```
%dfA_vec
%dfB_vec
% Mvec is a vector of monomer concentrations of length 3*num_sets
% MDD is a vector of dead-dead monomers, of length num_sets
% M_LiveA_LiveB
MLL = (1-dfA_vec) .* (1-dfB_vec) .* M_input_vec;
% M_LiveA_DeadB
MLD = (1-dfA_vec) .* (dfB_vec) .* M_input_vec;
% M_DeadA_LiveB
MDL = (dfA_vec) .* (1-dfB_vec) .* M_input_vec;
% Don't need to track dead-dead since concentration is constant
MDD = (dfA_vec) .* (dfB_vec) .* M_input_vec;
% Then reassort the monomers by class again

for isn = 1:num_sets
   Mvec((isn-1)*3+1) = MLL(isn);
   Mvec((isn-1)*3+2) = MLD(isn);
   Mvec((isn-1)*3+3) = MDL(isn);
end
% There will be three monomers per set that can do anything
num_monos = length(Mvec);
num_bimols = num_sets*(10) + pascalp(num_sets)*(16);
num_unicirc = num_sets;
% There are ABAB and ABBA double circles and each can be formed from
% interclass pairs
num_doublec = num_sets*2 + pascalp(num_sets)*2;
% But we will at least assume that double circles within a size class have the same kcd
% The multimers are split into LL, LD, and DD clumps
num_multi = 4;
num_species = num_monos + num_unicirc + num_doublec + num_bimols + num_multi; %
Conc_blank allows us to have an element of the concentration matrix that
% is always zero, for convenience so we don't have to check logic every
% time on whether an element exists (i.e. as opposed to having a "0" in an
% index column). Ensure that conc_vec(conc_blank) is set to and remains 0.
conc_blank = num_species + 1;
conc0_vec = zeros(num_species+1,1);
conc0_vec(1:num_monos) = Mvec;
% The parameter matrix is constructed to describe the formation of each individual species.
% Each row describes one species:
% monomers, then bimolecalar products, then uni circles, then double
% circles, then 3 kinds of multimers.
% At this point we can also set up the cyclization of all double-ended
% molecules, since there is only one partner in a cyclization.
%
% The nine columns are:
% 1: Species number that converts to a unimolecular circle. Number of nonzero
%    entries is 2 * the number of LL monomers, conc_blank otherwise
% 2: unimolecular cyclization rate constant for species indexed in column 1; + in the row
%    for the circle being formed, - in the row for the linear monomer
% 3: Species number #1 that converts to a double circle. There is just one
%    for an ABAB double circle.
% 4: Species number #2 that converts to a double circle, there are two for
%    ABBA and BAAB bimolecular products that give the same circle. Set this
%    to conc_blank for an ABAB circle
% 5: cyclization rate constant for the DC species; + in the row
```

```
%    for the circle being formed, - in the row for the linear monomer, assumed
%    the same for ABAB and ABBA/BAAB. If there were bending this would not be
%    true.
% 6: Species number #1 for the monomer that is incorporated into the bimolecular product for
%    the row, conc_blank otherwise
% 7: The other species that is in the bimolecular product
% 8: The number of ways the bimol product can be formed (= 2 for LA_isn_LB +
%    LA_isn_LB -> LA_isn_LB_LA_isn_LB, = 1 otherwise).
% 9: The number of live ends for each species -- this does not change, no
%    need to have a separate vector

param_matrix = zeros(num_species+1,9);

% Set up the parameter matrix by iterating through monomer sets
% First do intra-set and cyclization, then pairwise interactions
% Each set of 3 monomers is LL LD DL (not including DD, which is stuck at a constant
%  concentration determined above); only LL can yield circles
for mono_num = 1:num_monos
   set_num = floor((mono_num+2)/3); % 111 222 333 444
   param_matrix(mono_num,9) = 1 + ~mod(mono_num+2,3); % 211 211 211 211
   conc0_vec(mono_num) = Mvec(mono_num);
% This enters kc in the appropriate cyclization formation matrix
% Should not need to iterate through circles separately
   if ~mod(mono_num+2,3) % 100 100 100 100
      % There is no way to form a monomer, and the decay is set up
      % separately
      param_matrix(mono_num,:) = [ mono_num -1*kc_vec(set_num) conc_blank conc_blank 0
conc_blank conc_blank 0 2];
      param_matrix(num_monos+num_bimols+set_num,:) = [ mono_num kc_vec(set_num)
conc_blank conc_blank 0 conc_blank conc_blank 0 0];
% Next four lines are redundant here; written for parallel structure and
% completeness
%       conc0_vec(num_monos+num_bimols+set_num) = 0;
%       live_ends_vec(num_monos+num_bimols+set_num) = 0;
%    else
%       param_matrix(mono_num,:) = [ 0 0 0 0 0 0 0];
   else
      % Enter conc_blank in appropriate places so deltaconc program can
      % be written more simply
      param_matrix(mono_num,:) = [ conc_blank 0 conc_blank conc_blank 0 conc_blank conc_blank 0
1];
   end
end
% Any monomer can react with any other species, we take care of dead ends
% being dead later
% We step through the bimolecular products set by set "triangle-wise" ie.
% 1+1 1+2 1+3 1+4 ... then 2+2 2+3 2+4 ... then 3+3 3+4 ...
bimol_base = num_monos;
kcd_index = 0;
dc_base = num_monos + num_bimols + num_sets;
for isn0 = 1:num_sets
   bi_inter_abab{isn0} = [];
end
for isn1 = 1:num_sets
% intra-class
```

```
% just increment bimol_num to keep track instead of doing it smarter
   if isn1 > 1
      bimol_base = bimol_base + 10 + (num_sets-isn1+1)*16;
      dc_base = dc_base + (num_sets-isn1+2)*2;
   end
   kcd_index = kcd_index + 1;
   bimol_num = bimol_base + 1; %1
% First the ABAB dimers
% The first intraclass bimol is LA-isn1-LB_LA_isn1_LB so it can cyclize to an ABAB dimer
% circle, which is the only way to get that circle. The bimol  is formed from a single monomer in two
separate ways
   param_matrix(bimol_num,:) = [conc_blank 0 bimol_num conc_blank -kcd_vec(kcd_index) 3*(isn1-
1)+1 3*(isn1-1)+1 2 2];
   bi_intra_abab{isn1} = [bimol_num:bimol_num+3];
   param_matrix(dc_base+1,:) = [ conc_blank 0 bimol_num conc_blank kcd_vec(kcd_index)
conc_blank conc_blank 0 0];
   dc_abab(isn1) = dc_base+1;
   bimol_num = bimol_num+1; %2
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn1-1)+2 1
1];
   bimol_num = bimol_num+1; %3
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn1-1)+3 1
1];
% The fourth intraclass bimol has two dead ends, so it connot react further
   bimol_num = bimol_num+1; %4
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+2 3*(isn1-1)+3 1
0];
% And the ABBA
% First the LA-isn1-LB_LB_isn1_LA dimer
   bimol_num = bimol_num+1; %5
   dimer_circ_input1 = bimol_num;
   param_matrix(bimol_num,:) = [conc_blank 0 dimer_circ_input1 conc_blank -kcd_vec(kcd_index)
3*(isn1-1)+1 3*(isn1-1)+1 1 2];
   bimol_num = bimol_num+1; %6
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn1-1)+3 1
1];
   bimol_num = bimol_num+1; %7
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+3 3*(isn1-1)+3 1
0];
% And the BAAB
   bimol_num = bimol_num+1; %8
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank bimol_num -kcd_vec(kcd_index) 3*(isn1-
1)+1 3*(isn1-1)+1 1 2];
%   Now we can set up the second dimer circle
   dc_num = dc_base+2;
   param_matrix(dc_num,:) = [ conc_blank 0 dimer_circ_input1 bimol_num kcd_vec(kcd_index)
conc_blank conc_blank 0 0 ];
   bimol_num = bimol_num+1; %9
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn1-1)+2 1
1];
   bimol_num = bimol_num+1; %10
   param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+2 3*(isn1-1)+2 1
0];
% inter-class. Without loss of generality we list the ABAB products in that orientation, and
% specify the lower-numbered class on the left for ABBA and BAAB
```

```
% List inter-class dimers
% The four entries in the first column will all form at the same rate but
% in theory the DC's could form differently so we keep track separately.
   for isn2 = isn1+1:num_sets
      kcd_index = kcd_index + 1; % So these must be provided in the proper order
% First the ABAB dimers
% First four have lower isn to the left
      bi_inter_abab{isn1} = [ bi_inter_abab{isn1} [bimol_num+5:bimol_num+8]];
      bi_inter_abab{isn2} = [ bi_inter_abab{isn2} [bimol_num+1:bimol_num+4]];
      bimol_num = bimol_num+1; %1
      param_matrix(bimol_num,:) = [conc_blank 0 bimol_num conc_blank -kcd_vec(kcd_index)
3*(isn1-1)+1 3*(isn2-1)+1 1 2];
      dimer_circ_input1 = bimol_num;
      bimol_num = bimol_num+1; %2
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+3 3*(isn2-
1)+1 1 1];
      bimol_num = bimol_num+1; %3
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn2-
1)+2 1 1];
      bimol_num = bimol_num+1; %4
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+3 3*(isn2-
1)+2 1 1];
%       second set of four have lower number on the right. They are different.
      bimol_num = bimol_num+1; %5
      param_matrix(bimol_num,:) = [conc_blank 0 bimol_num conc_blank -kcd_vec(kcd_index)
3*(isn2-1)+1 3*(isn1-1)+1 1 2];
      dc_num = dc_num + 1;
      param_matrix(dc_num,:) = [ conc_blank 0 dimer_circ_input1 bimol_num kcd_vec(kcd_index)
conc_blank conc_blank 0 0];
      bimol_num = bimol_num+1; %6
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn2-1)+3 3*(isn1-
1)+1 1 1];
      bimol_num = bimol_num+1; %7
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn2-1)+1 3*(isn1-
1)+2 1 1];
      bimol_num = bimol_num+1; %8
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn2-1)+3 3*(isn1-
1)+2 1 1];
% Then the ABBA dimers. Constructed so the lower isn is always to the left
      bimol_num = bimol_num+1; %9
      param_matrix(bimol_num,:) = [conc_blank 0 bimol_num conc_blank -kcd_vec(kcd_index)
3*(isn1-1)+1 3*(isn2-1)+1 1 2];
      dimer_circ_input1 = bimol_num;
      bimol_num = bimol_num+1; %10
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+3 3*(isn2-
1)+1 1 1];
      bimol_num = bimol_num+1; %11
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn2-
1)+3 1 1];
      bimol_num = bimol_num+1; %12
      param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+3 3*(isn2-
1)+3 1 1];
% Then the BAAB dimers. Constructed so the lower isn is always to the left
      bimol_num = bimol_num+1; %13
```

```
        param_matrix(bimol_num,:) = [conc_blank 0 bimol_num conc_blank -kcd_vec(kcd_index)
3*(isn1-1)+1 3*(isn2-1)+1 1 2];
        dc_num = dc_num + 1;
        param_matrix(dc_num,:) = [ conc_blank 0 dimer_circ_input1 bimol_num kcd_vec(kcd_index)
conc_blank conc_blank 0 0];
        bimol_num = bimol_num+1; %14
        param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+2 3*(isn2-
1)+1 1 1];
        bimol_num = bimol_num+1; %15
        param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+1 3*(isn2-
1)+2 1 1];
        bimol_num = bimol_num+1; %16
        param_matrix(bimol_num,:) = [conc_blank 0 conc_blank conc_blank 0 3*(isn1-1)+2 3*(isn2-
1)+2 1 1];

% bimol_form_instraset_mat = ...
%     [ 2 0 0 ; ... % First four lines are the ABAB dimers
%       1 1 0 ; ...
%       1 0 1 ; ...
%       0 1 1 ; ...
%       1 0 0 ; ... % Then the ABBA
%       1 0 1 ; ...
%       0 0 1 ; ...
%       1 0 0 ; ... % Then the BAAB
%       1 1 0 ; ...
%       0 1 0 ]
% bimol_form_interset_mat_half =  ...
%     [ 1 0 0 ; ... % First four lines are the ABAB dimers
%       0 0 1 ; ...
%       1 0 0 ; ...
%       0 0 1 ; ...
%       1 0 0 ; ... % rows 5-8 switches 1 and 2
%       1 0 0 ; ...
%       0 1 0 ; ...
%       0 1 0 ; ...
%       1 0 0 ; ... % ABBA dimers
%       0 0 1 ;
%       1 0 0 ;
%
    end
end

% There is no mechanism for making monomers so the top num_monos row remain zeros
% Then any pair of monomers can make a dimer

% For example, for two sets the concs vector should contain the six types of monomer above that we
% track, plus
%  LA_1_LB,DA_1_LB,LA_1_DB,LA_2_LB,DA_1_LB,LA_1_DB
%
%  LA_1_LB_LA_1_LB, DA_1_LB_LA_1_LB, LA_1_LB_LA_1_DB, DA_1_LB_LA_1_DB,
%  LA_2_LB_LA_2_LB, DA_2_LB_LA_2_LB, LA_2_LB_LA_2_DB, DA_2_LB_LA_2_DB,
%  LA_1_LB_LA_2_LB, DA_1_LB_LA_2_LB, LA_1_LB_LA_2_DB, DA_1_LB_LA_2_DB,
%  LA_2_LB_LA_1_LB, DA_2_LB_LA_1_LB, LA_2_LB_LA_1_DB, DA_2_LB_LA_1_DB,
%
%  LA_1_LB_LB_1_LA, DA_1_LB_LB_1_LA, DA_1_LB_LB_1_DA,
```

```
%  LA_2_LB_LB_2_LA, DA_2_LB_LB_2_LA, DA_2_LB_LB_2_DA,
%  LA_1_LB_LB_2_LA, DA_1_LB_LB_2_LA, LA_1_LB_LB_2_DA, DA_1_LB_LB_2_DA,
%
%  LB_1_LA_LA_1_LB, DB_1_LA_LA_1_LB, DB_1_LA_LA_1_DB,
%  LB_2_LA_LA_2_LB, DB_2_LA_LA_2_LB, DB_2_LA_LA_2_DB,
%  LB_1_LA_LA_2_LB, DB_1_LA_LA_2_LB, LB_1_LA_LA_2_DB, DB_1_LA_LA_2_DB,
%  C1, C2,
%  DC_11_ABAB, DC_11_ABBA, DC_12_ABAB, DC_12_AABB, DC_22_ABAB, DC_22_AABB,
%  and lump all multimers

% For debugging send the param matrix to the screen
%param_matrix;

% 12dimer, 11DC, 22DC, 12DC,multimers
%concs0 =
[M1_ActiveA_ActiveB,M1_ActiveA_DeadB,M1_DeadA_ActiveB,M2_ActiveA_ActiveB,M2_Active
A_DeadB,M2_DeadA_ActiveB,...
%    zeros(1,45)];
% Can use ode23 or ode15s. ode15s seems to be about 40% faster.
% Inputs to deltaconcs that are not in the param_matrix are...kb,num_monos,
% num_bimols, num_unicirc, num_doublec
num_vec = [num_monos num_bimols num_unicirc num_doublec num_multi num_species]
% Try to pre-calculate everything that does not change
monos_indices = 1:num_vec(1);
bb = (num_vec(1)+1):(num_vec(1)+num_vec(2));
unic_indices = (num_vec(1)+num_vec(2)+1):(num_vec(1)+num_vec(2)+num_vec(3));
dc_indices =
(num_vec(1)+num_vec(2)+num_vec(3)+1):(num_vec(1)+num_vec(2)+num_vec(3)+num_vec(4));
multi_indices =
(num_vec(1)+num_vec(2)+num_vec(3)+num_vec(4))+1:(num_vec(1)+num_vec(2)+num_vec(3)+num
_vec(4)+num_vec(5)) ;
% Multimers can come from monomer + bimol or bimol + bimol or
% multimer+monomer or multimer + bimol or multimer + multimer
% How to keep track? This can be done in the dconcs routine by just adding
% everything.
% We do need to separately have a line for cyclization of the double-live
% ended multimer to give the fourth multimer population

param_matrix(multi_indices(1),:) = [multi_indices(1) -kc_multi conc_blank conc_blank 0 conc_blank
conc_blank 1 2];
param_matrix(multi_indices(4),:) = [multi_indices(1) kc_multi conc_blank conc_blank 0 conc_blank
conc_blank 1 0];

% For the multi-calc we need to have lists of monomers with 2 active ends,
% bimols with 2 ends, monos with 1 end, bimols with 1 end.
 [temp_ind_2,~] = find(param_matrix(:,9) == 2) ;
[temp_ind_1,~] = find(param_matrix(:,9) == 1) ;
monos_two_active = temp_ind_2(find(temp_ind_2 <= num_monos)) ;
bimols_two_active = temp_ind_2(find(temp_ind_2 > num_monos & temp_ind_2 < (num_monos +
num_bimols))) ;
monos_one_active = temp_ind_1(find(temp_ind_1 <= num_monos)) ;
bimols_one_active = temp_ind_1(find(temp_ind_1 > num_monos & temp_ind_1 < (num_monos +
num_bimols))) ;
```

```
options = odeset('RelTol',1e-7,'AbsTol',1e-9,'NonNegative',1:num_species);
 [t,concs] = ode15s(@(t,concs) deltaconcs(t,concs,param_matrix,...

monos_indices,bb,unic_indices,dc_indices,multi_indices,monos_two_active,monos_one_active,bimols
_two_active,bimols_one_active,num_species,kb),...
    [0 end_time],conc0_vec',options);
%concs
% For plotting we want to see the monomers, the unimolecular circles, the intra-class
% ABAB dimers that end up indistinguishable from circles, and the dimer circles that end up
indistinguishable
figure
hold on
conc_linear_multis = sum(concs(:,multi_indices(1:3)),2);
conc_circ_multis = sum(concs(:,multi_indices(4)),2);
for kplot = 1:num_sets
    conc_monomers(:,kplot) = sum(concs(:,[3*kplot-2:3*kplot]),2) + MDD(kplot);
    conc_bimol_intra_ABAB(:,kplot) = sum(concs(:,bi_intra_abab{kplot}),2);
    conc_bimol_inter_ABAB(:,kplot) = sum(concs(:,bi_inter_abab{kplot}),2);
    conc_unicircle(:,kplot) = concs(:,unic_indices(kplot));
    conc_dc_ABAB(:,kplot) = concs(:,dc_abab(kplot));
%    all_else (kplot) =
    plot(t,conc_monomers(:,kplot),'Color',[.25+.5*kplot/num_sets .25+.5*kplot/num_sets
.25+.5*kplot/num_sets],'LineWidth',2); % gray
    plot(t,conc_bimol_intra_ABAB(:,kplot),'Color',[.25+.75*kplot/num_sets 0 0],'LineWidth',2); % red
    %plot(t,conc_bimol_inter_ABAB(:,kplot),'Color',[.25+.75*kplot/num_sets 0
.25+.75*kplot/num_sets],'LineWidth',1); % red
    plot(t,conc_unicircle(:,kplot),'Color',[0 .25+.75*kplot/num_sets 0],'LineWidth',2); % green
    plot(t,conc_dc_ABAB(:,kplot),'Color',[0 0 .25+.75*kplot/num_sets],'LineWidth',2);% blue
end
plot(t,conc_linear_multis,'Color',[.25+.75*kplot/num_sets 0 .25+.75*kplot/num_sets],'LineWidth',2);%
pink
plot(t,conc_circ_multis,'Color',[ 0 .25+.75*kplot/num_sets .25+.75*kplot/num_sets],'LineWidth',2);%
teal
axis([0 1 0 0.0065]);
ax = axis;
box on

% The second plot just shows everything
figure
hold on
plot(t,concs(:,monos_indices),'Color',[0.5 0.5 0.5]); % gray
plot(t,concs(:,bb),'Color',[1 0 0]); % red
plot(t,concs(:,unic_indices),'Color',[0 1 0]); % green
plot(t,concs(:,dc_indices),'Color',[0 0 1]); % blue
plot(t,concs(:,multi_indices),'Color',[0.75 0.25 0]); % brown
legend
axis(ax);
%products = concs;
end

function dconcs = deltaconcs(~,concs,param_matrix,...

monos_indices,bb,unic_indices,dc_indices,multi_indices,monos_two_active,monos_one_active,bimols
_two_active,bimols_one_active,num_species,kb)
% here is where all the cleverness with the param_matrix comes into play
```

```
%
% the dot product below is essentially the concentration of all reactive
% ends in the mixture multiplied by kb. It is a scalar.
react_ends_rate = kb*concs'*(param_matrix(:,9));
M_2_conc_sum = sum(concs(monos_two_active));
B_2_conc_sum = sum(concs(bimols_two_active));
M_1_conc_sum = sum(concs(monos_one_active));
B_1_conc_sum = sum(concs(bimols_one_active));
dconcs = zeros(num_species+1,1);
%for ii = monos_indices -> looks like it can be done with an array
%statement.
% We assume that the param matrix entries have already identified which
% monomers make uni circles so we don't have to check the live ends
% number in the first line
dconcs(monos_indices) = concs(param_matrix(monos_indices,1)).*param_matrix(monos_indices,2)...
      - (concs(monos_indices).*param_matrix(monos_indices,9))*react_ends_rate ...
      - kb*concs(monos_indices).*concs(monos_indices).*param_matrix(monos_indices,9).^2;
dconcs(bb) = (concs(param_matrix(bb,3))+concs(param_matrix(bb,4))).*param_matrix(bb,5) ...
   + kb*param_matrix(bb,8).*concs(param_matrix(bb,6)).*concs(param_matrix(bb,7)) ...
   - (concs(bb).*param_matrix(bb,9))*react_ends_rate ...
   - kb*concs(bb).*concs(bb).*param_matrix(bb,9).^2 ;
dconcs(unic_indices) = concs(param_matrix(unic_indices,1)).*param_matrix(unic_indices,2);
dconcs(dc_indices) =
(concs(param_matrix(dc_indices,3))+concs(param_matrix(dc_indices,4))).*param_matrix(dc_indices,
5);
% Third line below -- when two double-live multis ligate together, the
% product is one double-ended multi so the concentration decreases
% The last line represents cyclization of multimer
dconcs(multi_indices(1)) = 4*kb*(M_2_conc_sum + 0.5*B_2_conc_sum)*B_2_conc_sum +  ...
                    + 4*kb*(M_2_conc_sum + B_2_conc_sum)*concs(multi_indices(1)) ...
                    - 4*kb*concs(multi_indices(1))*concs(multi_indices(1)) ...
                    - 2*kb*(M_1_conc_sum + B_1_conc_sum)*concs(multi_indices(1)) ...
                    - 2*kb*concs(multi_indices(1))*concs(multi_indices(2)) ...
                    + concs(param_matrix(multi_indices(1),1))*param_matrix(multi_indices(1),2);
dconcs(multi_indices(2)) = 2*kb*(M_2_conc_sum + B_2_conc_sum)*B_1_conc_sum +  ...
                    + 2*kb*(M_2_conc_sum + B_2_conc_sum)*concs(multi_indices(2)) ...
                    + 2*kb*(M_1_conc_sum + B_1_conc_sum)*concs(multi_indices(1)) ...
                    + 2*kb*concs(multi_indices(1))*concs(multi_indices(2)) ...
                    - kb*(M_1_conc_sum + B_1_conc_sum)*concs(multi_indices(2)) ...
                    - kb*concs(multi_indices(2))*concs(multi_indices(2));
dconcs(multi_indices(3)) = kb*(M_1_conc_sum + 0.5*B_1_conc_sum)*B_1_conc_sum +  ...
                    + kb*(M_1_conc_sum + B_1_conc_sum)*concs(multi_indices(2)) ...
                    + kb*concs(multi_indices(2))*concs(multi_indices(2));
dconcs(multi_indices(4)) =
concs(param_matrix(multi_indices(4),1))*param_matrix(multi_indices(4),2);
%sum(dconcs)
end
```

Appendix 5.4: List of Tracked Species in compete_cyc.m

12 Monomers:
47-Live A-Live B
47-Live A-Dead B
47-Dead A-Live B
47-Dead A-Dead B

77-Live A-Live B
77-Live A-Dead B
77-Dead A-Live B
77-Dead A-Dead B
107-Live A-Live B
107-Live A-Dead B
107-Dead A-Live B
107-Dead A-Dead B

3 Monomer Circles:

47-Circle
77-Circle
107-Circle

78 Linear Dimers:

47-Live B-Live A + 47-Live A-Live B
47-Live B-Live A + 47-Live A-Dead B
47-Live A-Live B + 47-Live B-Live A
47-Live A-Live B + 47-Live B-Dead A
47-Live A-Live B + 47-Live A-Live B
47-Live A-Live B + 47-Live A-Dead B
47-Dead B-Live A + 47-Live A-Dead B
47-Dead A-Live B + 47-Live B-Dead A
47-Dead A-Live B + 47-Live A-Live B
47-Dead A-Live B + 47-Live A-Dead B
47-Live B-Live A + 77-Live B-Live A
47-Live B-Live A + 77-Live B-Dead A
47-Live B-Live A + 77-Live A-Live B
47-Live B-Live A + 77-Live A-Dead B
47-Live A-Live B + 77-Live B-Live A
47-Live A-Live B + 77-Live B-Dead A
47-Live A-Live B + 77-Live A-Live B
47-Live A-Live B + 77-Live A-Dead B
47-Dead B-Live A + 77-Live B-Live A
47-Dead B-Live A + 77-Live B-Dead A
47-Dead B-Live A + 77-Live A-Live B
47-Dead B-Live A + 77-Live A-Dead B
47-Dead A-Live B + 77-Live B-Live A
47-Dead A-Live B + 77-Live B-Dead A
47-Dead A-Live B + 77-Live A-Live B
47-Dead A-Live B + 77-Live A-Dead B
47-Live B-Live A + 107-Live B-Live A
47-Live B-Live A + 107-Live B-Dead A
47-Live B-Live A + 107-Live A-Live B
47-Live B-Live A + 107-Live A-Dead B
47-Live A-Live B + 107-Live B-Live A
47-Live A-Live B + 107-Live B-Dead A
47-Live A-Live B + 107-Live A-Live B
47-Live A-Live B + 107-Live A-Dead B
47-Dead B-Live A + 107-Live B-Live A
47-Dead B-Live A + 107-Live B-Dead A
47-Dead B-Live A + 107-Live A-Live B
47-Dead B-Live A + 107-Live A-Dead B
47-Dead A-Live B + 107-Live B-Live A
47-Dead A-Live B + 107-Live B-Dead A
47-Dead A-Live B + 107-Live A-Live B
47-Dead A-Live B + 107-Live A-Dead B

77-Live B-Live A + 77-Live A-Live B
77-Live B-Live A + 77-Live A-Dead B
77-Live A-Live B + 77-Live B-Live A
77-Live A-Live B + 77-Live B-Dead A
77-Live A-Live B + 77-Live A-Live B
77-Live A-Live B + 77-Live A-Dead B
77-Dead B-Live A + 77-Live A-Dead B
77-Dead A-Live B + 77-Live B-Dead A
77-Dead A-Live B + 77-Live A-Live B
77-Dead A-Live B + 77-Live A-Dead B
77-Live B-Live A + 107-Live B-Live A
77-Live B-Live A + 107-Live B-Dead A
77-Live B-Live A + 107-Live A-Live B
77-Live B-Live A + 107-Live A-Dead B
77-Live A-Live B + 107-Live B-Live A
77-Live A-Live B + 107-Live B-Dead A
77-Live A-Live B + 107-Live A-Live B
77-Live A-Live B + 107-Live A-Dead B
77-Dead B-Live A + 107-Live B-Live A
77-Dead B-Live A + 107-Live B-Dead A
77-Dead B-Live A + 107-Live A-Live B
77-Dead B-Live A + 107-Live A-Dead B
77-Dead A-Live B + 107-Live B-Live A
77-Dead A-Live B + 107-Live B-Dead A
77-Dead A-Live B + 107-Live A-Live B
77-Dead A-Live B + 107-Live A-Dead B
107-Live B-Live A + 107-Live A-Live B
107-Live B-Live A + 107-Live A-Dead B
107-Live A-Live B + 107-Live B-Live A
107-Live A-Live B + 107-Live B-Dead A
107-Live A-Live B + 107-Live A-Live B
107-Live A-Live B + 107-Live A-Dead B
107-Dead B-Live A + 107-Live A-Dead B
107-Dead A-Live B + 107-Live B-Dead A
107-Dead A-Live B + 107-Live A-Live B
107-Dead A-Live B + 107-Live A-Dead B

12 Dimer Circles:

47-Live B-Live A + 47-Live A-Live B
47-Live A-Live B + 47-Live A-Live B
47-Live B-Live A + 77-Live B-Live A
47-Live B-Live A + 77-Live A-Live B
47-Live B-Live A + 107-Live B-Live A
47-Live B-Live A + 107-Live A-Live B
77-Live B-Live A + 77-Live A-Live B
77-Live A-Live B + 77-Live A-Live B
77-Live B-Live A + 107-Live B-Live A
77-Live B-Live A + 107-Live A-Live B
107-Live B-Live A + 107-Live A-Live B
107-Live A-Live B + 107-Live A-Live B

4 Multimers (anything with 3+ monomers contributing):

Multimer with two live ends
Multimer with one live end and one dead end
Multimer with two dead ends
Multimer circles

Appendix 5.5: FPLC MBP-trap

Variables:
InjectionVolume: 350 {CV}
Main Method:

Main
0.00 Base CV 0.96 {ml}HiTrap_Chelating_HP_1_ml
0.00 Block starting_conditions
0.00 Base SameAsMain
0.00 AutozeroUV
0.00 Alarm_Pressure Enabled 0.400 {MPa} 0.000 {MPa}
0.00 Flow 1.00 {ml/min}
0.00 ColumnPosition Position1Bypass
0.00 InjectionValve Load
0.00 Message "Move Pump A to EQ and Pump B to Elution"
Screen "Default sound"
0.00 Pause INFINITE {Minutes}
0.00 Message "Verify Column is attached at position 3"
Screen "No sound"
0.00 Pause INFINITE {Minutes}
0.00 Message "Execute manual pump wash on A & B"
Screen "No sound"
0.00 Pause INFINITE {Minutes}
0.00 Flow 2.00 {ml/min}
5.00 InjectionValve Inject
7.00 InjectionValve Load
10.00 ColumnPosition Position3
13.00 End_Block
0.00 Block column_equilibration
0.00 Base SameAsMain
0.00 Flow 1.00 {ml/min}
0.00 ColumnPosition Position3
8.00 AutozeroUV
8.00 End_Block
0.00 Block sample_inject
0.00 Base SameAsMain
0.00 ColumnPosition Position3
0.00 Message "Move Pump A to Lysate"
Screen "Default sound"
0.00 Pause INFINITE {Minutes}
0.00 Fractionation 30mm 50.00 {ml} FirstTube Volume
0.00 Flow 1.00 {ml/min}
0.00 Set_Mark "sample_inject"
(350.00) #InjectionVolume End_Block
0.00 Block pump_clear
0.00 Base SameAsMain
0.00 Message "Add 50 mL EQ Buffer to Lysate Bottle"
Screen "Default sound"
0.00 Pause INFINITE {Minutes}
0.00 Fractionation 30mm 50.00 {ml} NextTube Volume
0.00 Flow 1.00 {ml/min}
0.00 Set_Mark "pump_clear"
40.00 FractionationStop
40.00 End_Block
0.00 Block pump_wash

0.00 Base SameAsMain
0.00 ColumnPosition Position1Bypass
0.00 Message “Move Pump A to EQ Buffer”
Screen “Default sound”
0.00 Pause INFINITE {Minutes}
0.00 Message “Execute manual pump wash on Pump A”
Screen “No sound”
0.00 Pause INFINITE {Minutes}
0.00 Flow 5.00 {ml/min}
10.00 Flow 0.00 {ml/min}
10.00 ColumnPosition Position3
10.00 End_Block

0.00 Block column_wash

0.00 Base SameAsMain
0.00 ColumnPosition Position3
0.00 Flow 1.00 {ml/min}
0.00 Set_Mark “column_wash”
0.00 Fractionation 18mm 10 {ml} FirstTube Volume
10.00 FractionationStop
10.00 End_Block

0.00 Block elution

0.00 Base SameAsMain
0.00 Fractionation 18mm 1.000 {ml} TubeNumber[A.2] Volume
0.00 Gradient 50 {%B} 15 {base}
15.00 Gradient 100 {%B} 1.00 {base}
20.00 FractionationStop
22.00 End_Block

Appendix 6: Common Buffers

Gel Extraction Buffer pH 7 (200 mL)
0.82 g NaOAc (final 50 mM)
400 μL 500 mM EDTA (final 1 mM)
Adjust final pH to 7

Autoinduction A (250 mL; filter sterilized)
12.5 g lactose
3.125 g glucose
31.25 mL glycerol

Autoinduction B (250 mL; filter sterilized)
19.8 g $(NH_4)SO_4$
42.5 g KH_2PO_4
44.375 g Na_2HPO_4

Autoinduction solution for culture
1 L LB, 40 mL Autoinduction A, 40 mL Autoinduction B, 1 mL 100 mg/mL Amp

IMAC Equilibration Buffer (500 mL; filtered)
1.97g Tris HCl pH 7.5 (final 25 mM)
14.61 g NaCl (final 0.5 M)
1.02 g Imidazole (or 1.57 g Imidazole HCl; final 30 mM)

IMAC Elution Buffer (500 mL; filtered)
1.97 g Tris HCl pH 7.5 (final 25 mM)
14.61 g NaCl (final 0.5 M)
13.62 g Imidazole (or 20.91 g Imidazole HCl; 400 mM)

Maltose Equilibration Buffer (500 mL; filtered)
1.576 g Tris HCl pH 7.5 (final 20 mM)
5.844 g NaCl (final 200 mM)
0.146 g EDTA (final 1 mM)

Maltose Elution Buffer (500 mL; filtered)
1.576 g Tris Hcl pH 7.5 (final 20 mM)
5.844 g NaCl (final 200 mM)
0.146 g EDTA (final 1 mM)
1.802 g Maltose (final 10 mM)

TEV Reaction Buffer (200 mL; filter sterilized)
200 μL 0.5 M EDTA pH 8 (final 500 μM)
0.031 g DTT (final 1 mM)
1.576 g Tris HCl pH 8 (or 5 mL 2 M Tris pH 8; final 50 mM)

TEV Storage Buffer (50 mL; filter sterilized)
100 μL 0.5 M EDTA pH 8 (final 1 mM)
0.0385 g DTT (final 5 mM)
1.25 mL 2M Tris pH 7.5 (final 50 mM)
25 mL glycerol (final 50% v/v)
0.05 g Triton X-100 (final 0.1% w/v)

10X TE pH 7.5 (50 mL)
2.5 mL 2 M Tris pH 7.5 (10x is 100 mM, 1x is 10 mM)
1 mL 500 mM EDTA pH 8 (10x is 10 mM, 1x is 1 mM)

5x TBE (500 mL)
27 g Tris base (5x is 445 mM, 1x is 89 mM)
13.75 g Boric acid (5x is 445 mM, 1x is 89 mM)
20 mL 0.5 M EDTA pH 8 (5x is 10 mM, 1x is 2 mM)

NEB CutSmart Buffer (1x, pH 7.9)
50 mM Potassium Acetate
20 mM Tris Acetate
10 mM Magnesium Acetate
100 μg/mL BSA

NEB T4 Ligase Buffer (1x, pH 7.5)
50 mM Tris HCl
10 mM $MgCl_2$
10 mM DTT
1 mM ATP

NEB T4 Polynucleotide Kinase Buffer (1x, pH 7.6)
70 mM Tris HCl
10 mM $MgCl_2$
5 mM DTT
ATP MUST BE ADDED

Bibliography

1. Wang, J. Helical repeat of DNA in solution. *Proc National Acad Sci* **76,** 200–203 (1979).

2. Watson, J. & Crick, F. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature* **171,** 737–738 (1953).

3. Malik, M., Bensaid, A., Rouviere-Yaniv, J. & Drlica, K. Histone-like Protein HU and Bacterial DNA Topology: Suppression of an HU Deficiency by Gyrase Mutations. *J Mol Biol* **256,** 66–76 (1996).

4. Narayana, N. & Weiss, M. A. Crystallographic Analysis of a Sex-Specific Enhancer Element: Sequence-Dependent DNA Structure, Hydration, and Dynamics. *J Mol Biol* **385,** 469–490 (2009).

5. Marini, J., Levene, S., Crothers, D. & Englund, P. A Bent Helix in Kinetoplast DNA. *Cold Spring Harb Sym* **47,** 279–283 (1983).

6. Schultz, S., Shields, G. & Steitz, T. Crystal structure of a CAP-DNA complex: the DNA is bent by 90 degrees. *Science* **253,** 1001 (1991).

7. Gartenberg, M. R. & Crothers, D. M. DNA sequence determinants of CAP-induced bending and protein binding affinity. *Nature* **333,** 824–829 (2001).

8. Starr, B. D., Hoopes, B. C. & Hawley, D. K. DNA Bending is an Important Component of Site-specific Recognition by the TATA Binding Protein. *J Mol Biol* **250,** 434–446 (1995).

9. Haeusler, A. R., Goodson, K. A., Lillian, T. D., Wang, X., Goyal, S., Perkins, N. C. & Kahn, J. D. FRET studies of a landscape of Lac repressor-mediated DNA loops.

Nucleic Acids Res **40,** 4432 4445 (2012).

10. Koudelka, G., Harbury, P., Harrison, S. & Ptashne, M. DNA twisting and the affinity of bacteriophage 434 operator for bacteriophage 434 repressor. *Proc National Acad Sci* **85,** 4633–4637 (1988).

11. Mitchell, J. S., Glowacki, J., Grandchamp, A. E., Manning, R. S. & Maddocks, J. H. Sequence-dependent persistence lengths of DNA. *J Chem Theory Comput* (2016). doi:10.1021/acs.jctc.6b00904

12. Jacobson, H. & Stockmayer, W. H. Intramolecular Reaction in Polycondensations. I. The Theory of Linear Systems. *J Chem Phys* **18,** 1600–1606 (1950).

13. Shore, D., Langowski, J. & Baldwin, R. DNA flexibility studied by covalent closure of short fragments into circles. *Proc National Acad Sci* **78,** 4833–4837 (1981).

14. Shore, D. & Baldwin, R. L. Energetics of DNA twisting I. Relation between twist and cyclization probability. *J Mol Biol* **170,** 957–981 (1983).

15. Shimada, J. & Yamakawa, H. Ring-closure probabilities for twisted wormlike chains. Application to DNA. *Macromolecules* **17,** 689–698 (1984).

16. Lu, X. & Olson, W. K. 3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. *Nucleic Acids Res* **31,** 5108–5121 (2003).

17. Humphrey, W., Dalke, A. & Schulten, K. VMD: Visual molecular dynamics. *J Mol Graphics* **14,** 33–38 (1996).

18. Wang, J. C. & Davidson, N. On the probability of ring closure of lambda DNA. *J*

Mol Biol **19,** 469–482 (1966).

19. Studier, W. F. Sedimentation studies of the size and shape of DNA. *J Mol Biol* **11,** 373–390 (1965).

20. Vinograd, J., Bruner, R., Kent, R. & Weigle, J. Band-Centrifugation of Macromolecules and Viruses in Self-Generating Density Gradients. *Proc National Acad Sci* **49,** 902–910 (1963).

21. Crothers, D. M., Drak, J., Kahn, J. D. & Levene, S. D. DNA bending, flexibility, and helical repeat by cyclization kinetics. *Methods Enzymol* **212,** 3–29 (1992).

22. Ticau, S., Friedman, L. J., Champasa, K., Jr, I. R., Gelles, J. & Bell, S. P. Mechanism and timing of Mcm2-7 ring closure during DNA replication origin licensing. *Nat Struct Mol Biol* **24,** 309–315 (2017).

23. Vafabakhsh, R. & Ha, T. Extreme bendability of DNA less than 100 base pairs long revealed by single-molecule cyclization. *Science* **337,** 1097 1101 (2012).

24. Zhang, Y. & Crothers, D. M. Statistical Mechanics of Sequence-Dependent Circular DNA and Its Application For DNA Cyclization. *Biophys J* **84,** 136–153 (2003).

25. Du, Q., Vologodskaia, M., Kuhn, H., Frank-Kamenetskii, M. & Vologodskii, A. Gapped DNA and Cyclization of Short DNA Fragments. *Biophys J* **88,** 4137–4145 (2005).

26. Du, Q., Kotlyar, A. & Vologodskii, A. Kinking the double helix by bending deformation. *Nucleic Acids Res* **36,** 1120–1128 (2008).

27. Cloutier, T. & Widom, J. DNA twisting flexibility and the formation of sharply looped protein–DNA complexes. *P Natl Acad Sci Usa* **102,** 3645–3650 (2005).

28. Ha, T., Rasnik, I., Cheng, W., Babcock, H. P., Gauss, G. H., Lohman, T. M. & Chu, S. Initiation and re-initiation of DNA unwinding by the Escherichia coli Rep helicase. *Nature* **419,** 638 641 (2002).

29. Mehta, R. & Kahn, J. Designed hyperstable Lac repressor.DNA loop topologies suggest alternative loop geometries. *J Mol Biol* **294,** 67 77 (1999).

30. Brenowitz, M., Pickar, A. & Jamison, E. Stability of a Lac repressor mediated 'looped complex'. *Biochemistry-us* **30,** 5986–5998 (1991).

31. Medler, S., Husini, N., Raghunayakula, S., Mukundan, B., Aldea, A. & Ansari, A. Evidence for a Complex of Transcription Factor IIB (TFIIB) with Poly(A) Polymerase and Cleavage Factor 1 Subunits Required for Gene Looping. *J Biol Chem* **286,** 33709–33718 (2011).

32. Stenger, J., Tegtmeyer, P., Mayr, G., Reed, M., Wang, Y., Wang, P., Hough, P. & Mastrangelo, I. p53 oligomerization and DNA looping are linked with transcriptional activation. *Embo J* **13,** 6011–6020 (1994).

33. Saiz, L. The physics of protein–DNA interaction networks in the control of gene expression. *Journal of Physics: Condensed Matter* **24,** 193102 (2012).

34. Gowetski, D. B., Kodis, E. J. & Kahn, J. D. Rationally designed coiled-coil DNA looping peptides control DNA topology. *Nucleic Acids Res* **41,** 8253 8265 (2013).

35. Kovari, D. T., Yan, Y., Finzi, L. & Dunlap, D. Single Molecule Analysis, Methods and Protocols. 317–340 (2017). doi:10.1007/978-1-4939-7271-5_17

36. Nelson, P. C., Zurla, C., Brogioli, D., Beausang, J. F., Finzi, L. & Dunlap, D. Tethered Particle Motion as a Diagnostic of DNA Tether Length. *J Phys Chem B* **110,** 17260–17267 (2006).

37. Du, Q., Smith, C., Shiffeldrim, N., Vologodskaia, M. & Vologodskii, A. Cyclization of short DNA fragments and bending fluctuations of the double helix. *P Natl Acad Sci Usa* **102,** 5397–5402 (2005).

38. Kahn, J. D. Topological Effects of the TATA Box Binding Protein on Minicircle DNA and a Possible Thermodynamic Linkage to Chromatin Remodeling†. *Biochemistry-us* **39,** 3520–3524 (2000).

39. Sanger, F., Nicklen, S. & Coulson, A. DNA sequencing with chain-terminating inhibitors. *Proc National Acad Sci* **74,** 5463–5467 (1977).

40. Collins, F. S., Morgan, M. & Patrinos, A. The Human Genome Project: Lessons from Large-Scale Biology. *Science* **300,** 286–290 (2003).

41. Luo, C., Tsementzi, D., Kyrpides, N., Read, T. & Konstantinidis, K. T. Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *Plos One* **7,** e30087 (2012).

42. Illumina. An Introduction to Next-Generation Sequencing Technology. Accessed 2019-05-13. http://www.illumina.com/technology/next-generation-sequencing.html

43. Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., Boutell, J. M., Bryant, J., Carter, R. J., Cheetham, K. R., Cox, A. J., Ellis, D. J., Flatbush, M. R., Gormley, N. A., Humphray, S. J., Irving, L. J., Karbelashvili, M. S., Kirk, S. M., Li, H., Liu, X., Maisinger, K. S., Murray, L. J., Obradovic, B., Ost, T., Parkinson, M. L., Pratt, M. R., Rasolonjatovo, I. M., Reed, M. T., Rigatti, R., Rodighiero, C., Ross, M. T., Sabot, A., Sankar, S. V., Scally, A., Schroth, G. P., Smith, M. E., Smith, V. P., Spiridou, A., Torrance, P. E., Tzonev, S. S., Vermaas, E. H., Walter, K., Wu, X.,

Zhang, L., Alam, M. D., Anastasi, C., Aniebo, I. C., Bailey, D. M., Bancarz, I. R., Banerjee, S., Barbour, S. G., Baybayan, P. A., Benoit, V. A., Benson, K. F., Bevis, C., Black, P. J., Boodhun, A., Brennan, J. S., Bridgham, J. A., Brown, R. C., Brown, A. A., Buermann, D. H., Bundu, A. A., Burrows, J. C., Carter, N. P., Castillo, N., Catenazzi, M. E., Chang, S., Cooley, N. R., Crake, N. R., Dada, O. O., Diakoumakos, K. D., Dominguez-Fernandez, B., Earnshaw, D. J., Egbujor, U. C., Elmore, D. W., Etchin, S. S., Ewan, M. R., Fedurco, M., Fraser, L. J., Fajardo, K. V., Furey, S. W., George, D., Gietzen, K. J., Goddard, C. P., Golda, G. S., Granieri, P. A., Green, D. E., Gustafson, D. L., Hansen, N. F., Harnish, K., Haudenschild, C. D., Heyer, N. I., Hims, M. M., Ho, J. T., Horgan, A. M., Hoschler, K., Hurwitz, S., Ivanov, D. V., Johnson, M. Q., James, T., Jones, H. T., Kang, G.-D., Kerelska, T. H., Kersey, A. D., Khrebtukova, I., Kindwall, A. P., Kingsbury, Z., Kokko-Gonzales, P. I., Kumar, A., Laurent, M. A., Lawley, C. T., Lee, S. E., Lee, X., Liao, A. K., Loch, J. A., Lok, M., Luo, S., Mammen, R. M., Martin, J. W., McCauley, P. G., McNitt, P., Mehta, P., Moon, K. W., Mullens, J. W., Newington, T., Ning, Z., Ng, B., Novo, S. M., O'Neill, M. J., Osborne, M. A., Osnowski, A., Ostadan, O., Paraschos, L. L., Pickering, L., Pike, A. C., Pike, A. C., Pinkard, C. D., Pliskin, D. P., Podhasky, J., Quijano, V. J., Raczy, C., Rae, V. H., Rawlings, S. R., Rodriguez, A., Roe, P. M., Rogers, J., Bacigalupo, M. C., Romanov, N., Romieu, A., Roth, R. K., Rourke, N. J., Ruediger, S. T., Rusman, E., Sanches-Kuiper, R. M., Schenker, M. R., Seoane, J. M., Shaw, R. J., Shiver, M. K., Short, S. W., Sizto, N. L., Sluis, J. P., Smith, M. A., Sohna, J., Spence, E. J., Stevens, K., Sutton, N., Szajkowski, L., Tregidgo, C. L., Turcatti, G., vandeVondele, S., Verhovsky, Y., Virk, S. M., Wakelin, S., Walcott, G. C., Wang, J.,

Worsley, G. J., Yan, J., Yau, L., Zuerlein, M., Rogers, J., Mullikin, J. C., Hurles, M. E., McCooke, N. J., West, J. S., Oaks, F. L., Lundberg, P. L., Klenerman, D., Durbin, R. & Smith, A. J. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* **456,** 53 (2008).

44. Wingett, S. W. & Andrews, S. FastQ Screen: A tool for multi-genome mapping and quality control. *F1000research* **7,** 1338 (2018).

45. Magoč, T. & Salzberg, S. L. FLASH: fast length adjustment of short reads to improve genome assemblies. *Bioinformatics* **27,** 2957–2963 (2011).

46. Aird, D., Ross, M. G., Chen, W.-S., Danielsson, M., Fennell, T., Russ, C., Jaffe, D. B., Nusbaum, C. & Gnirke, A. Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome Biol* **12,** R18 (2011).

47. Reuter, J. S. & Mathews, D. H. RNAstructure: software for RNA secondary structure prediction and analysis. *Bmc Bioinformatics* **11,** 129 (2010).

48. Xayaphoummine, A., Bucher, T. & Isambert, H. Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. *Nucleic Acids Res* **33,** W605–W610 (2005).

49. Kircher, M., Sawyer, S. & Meyer, M. Double indexing overcomes inaccuracies in multiplex sequencing on the Illumina platform. *Nucleic Acids Res* **40,** e3–e3 (2012).

50. Chen, J. & Seeman, N. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature* **350,** 631 633 (1991).

51. Saccà, B. & Niemeyer, C. M. Functionalization of DNA nanostructures with proteins. *Chem Soc Rev* **40,** 5910 5921 (2011).

52. Lund, K., Manzo, A. J., Dabby, N., Michelotti, N., Johnson-Buck, A., Nangreave,

J., Taylor, S., Pei, R., Stojanovic, M. N., Walter, N. G., Winfree, E. & Yan, H. Molecular robots guided by prescriptive landscapes. *Nature* **465,** 206 210 (2010).

53. Saghatelian, A., Guckian, K. M., Thayer, D. A. & Ghadiri, R. M. DNA detection and signal amplification via an engineered allosteric enzyme. *J Am Chem Soc* **125,** 344 345 (2003).

54. Shin, J.-S. & Pierce, N. A. A synthetic DNA walker for molecular transport. *J Am Chem Soc* **126,** 10834 10835 (2004).

55. Williams, D. D., Benedek, O. & Turnbough, C. L. Species-specific peptide ligands for the detection of Bacillus anthracis spores. *Appl Environ Microb* **69,** 6288 6293 (2003).

56. Fruk, L. & Niemeyer, C. M. Covalent hemin-DNA adducts for generating a novel class of artificial heme enzymes. *Angewandte Chemie Int Ed* **44,** 2603 2606 (2005).

57. Gollmick, F. A., Lorenz, M., Dornberger, U., von Langen, J., Diekmann, S. & Fritzsche, H. Solution structure of dAATAA and dAAUAA DNA bulges. *Nucleic Acids Res* **30,** 2669–2677 (2002).

58. Leng, Q. & Mixson, A. Small interfering RNA targeting Raf-1 inhibits tumor growth in vitro and in vivo. *Cancer Gene Ther* **12,** 7700831 (2005).

59. Leng, Q. & Mixson, J. A. Modified branched peptides with a histidine-rich tail enhance in vitro gene transfection. *Nucleic Acids Res* **33,** e40–e40 (2005).

60. Leng, Q., Scaria, P., Zhu, J., Ambulos, N., Campbell, P. & Mixson, J. A. Highly branched HK peptides are effective carriers of siRNA. *J Gene Medicine* **7,** 977 986 (2005).

61. Lee, A., Karcz, A., Akman, R., Zheng, T., Kwon, S., Chou, S., Sucayan, S.,

Tricoli, L. J., Hustedt, J. M., Leng, Q., Kahn, J. D., Mixson, J. A. & Seog, J. Direct Observation of Dynamic Mechanical Regulation of DNA Condensation by Environmental Stimuli. *Angewandte Chemie Int Ed* **53,** 10631–10635 (2014).

62. Leng, Q., Goldgeier, L., Zhu, J., Cambell, P., Ambulos, N. & Mixson, A. Histidine-lysine peptides as carriers of nucleic acids. *Drug News Perspect* **20,** 77 (2007).

63. Leng, Q., Scaria, P., Lu, P., Woodle, M. & Mixson, A. Systemic delivery of HK Raf-1 siRNA polyplexes inhibits MDA-MB-435 xenografts. *Cancer Gene Ther* **15,** cgt200829 (2008).

64. Chou, S.-T., Hom, K., Zhang, D., Leng, Q., Tricoli, L. J., Hustedt, J. M., Lee, A., Shapiro, M. J., Seog, J., Kahn, J. D. & Mixson, J. A. Enhanced silencing and stabilization of siRNA polyplexes by histidine-mediated hydrogen bonds. *Biomaterials* **35,** 846–855 (2014).

65. Guo, S. & Akhremitchev, B. Investigation of mechanical properties of insulin crystals by atomic force microscopy. *Langmuir* **24,** 880 887 (2008).