2016-12-01

Analysis of Nucleosome Isolation and Recovery: From *In Silico* Invitrosomes to *In Vivo* Nucleosomes

Collin Brendan Skousen
Brigham Young University

Analysis of Nucleosome Isolation and Recovery: From *In Silico* Invitrosomes to *In Vivo* Nucleosomes

Collin Brendan Skousen

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Steven M. Johnson, Chair
Richard Robison
Joshua A. Udall

Department of Microbiology and Molecular Biology

Brigham Young University

Abstract

Analysis of Nucleosome Isolation and Recovery: From *In Silico* Invitrosomes to *In Vivo* Nucleosomes

Collin Brendan Skousen
Department of Microbiology and Molecular Biology, BYU
Master of Science

There are a vast number of factors that influence nucleosome formation, and consequently gene regulation. These factors include histone modifications, nucleotide composition, transcriptional region elements, and specific nucleotide motifs, among others. Although the amount we know now is limited, we are creating new techniques and discoveries to assist us in continued understanding of chromatin. To make a significant contribution to the field of chromatin, I conducted two hypothesis driven sets of experiments that address the topic of chromatin structure. First, I created a technique for tissue specific nucleosome isolation with the goal of observing the effect of single nucleotide polymorphisms (SNPs) on nucleosome formation. Second, I created and tested a method to recover lost *in vitro* nucleosome reconstitution data, which can improve this type of data, commonly used for observing nucleosome positioning. The first experiment needs a more specific antibody to complete the last step and function as designed. The second experiment shows that our nucleosome recovery method, when applied conservatively, can recover 90% of the lost nucleosome data.

Keywords: nucleosomes, tissue specific nucleosome isolation, in vitro nucleosome reconstitution

Acknowledgements

The last two years have been an incredible period of growth. It was difficult beyond belief between failed experiments, different data from expected, and rewriting our program into a different language after failure. However, it was all made doable and manageable through the fantastic support system and the amazing people with whom I was able to work.

First and foremost is my advisor, Dr. Steven Johnson, for his ability to find humor in science, for his valuable insights into my projects, and for his continued guidance beginning in undergraduate and continuing through my graduate career. I could not have been successful without his support and guidance. Dr. Joshua Udall, for his knowledge in bioinformatics. I obtained much of my coding experience through his classes and teaching. Dr. Richard Robison, for his knowledge of molecular biology. I started in genetics, but through his guidance and classes, I learned much of Immunology and Molecular Biology.

Additionally, I would like to thank Jordon Ritchie, who wrote the Java script used for the nucleosome recovery. I would also like to thank him for helping me learn programming and scripting. Colton Kempton, David Bates, Kevin Adams and the other members of the Johnson lab all helped with their knowledge, support and assistance in my projects.

Table of Contents

List of Tables

List of Figures

Chapter 1. Background

Adenine, Thymine, Guanine and Cystosine make up the composition of the most basic part of life, DNA. Strung together, these bases become instructions for the cell in its protein making process. The human genome contains three billion of these bases, each paired with its complement, and organized into chromosomes [1]. Aside from gamete cells, every cell contains two copies of the genome, for a total of 12 billion nucleotides. With each base pair at 0.34 nanometers long, six billion base pairs adds up to two meters of DNA when placed end to end! An estimate of 37.2 trillion cells are in the human body, equating to ~75 trillion meters of DNA within a human body [18]. How can we relate this into terms we can better understand? The earth has a circumference of 40 million meters [2]. DNA from a single human can stretch and wrap around the earth almost 1.9 million times! It seems implausible that such a large amount of important biological information can be contained within such a small container. How does two meters of DNA fit into a cell's nucleus, which only has a diameter of five micrometers? [3]

We look into a class of proteins called histones for the answer. Histones were discovered in 1884 by Albrecht Kossel, and later defined as DNA compaction proteins by Roger Kornberg in 1974 [4]. There are four types of core histones that when two each of the H2A, H2B, H3 and H4 histone proteins combine create an octamer of histone proteins. The octamer takes on a globular or wheel like structure, which is perfect for DNA compaction. DNA has a negative charge, and is attracted to the histones, which are positively charged. Grooves along the octamer give the DNA an optimal fit as 147 bp wraps 1.7 times around the histone octamer [5-7]. The whole unit of DNA wrapped around a histone octamer is called a nucleosome. A final histone protein, H1, acts

as a linker, and wraps up the additional DNA and increases the compaction between nucleosomes [8].

Recent studies have shown that there are many factors that contribute to octamer affinity. A recent paper by Voulouev et al. looks at the DNA composition of bound octamers [30]. Using high throughput sequencing, they looked at almost 1.3 billion reads of nucleosome positions and found similarities between octamer binding regions. While looking at nucleosomes *in vivo*, they observed that nucleosomes prefer to sit in close relation to the next nucleosome. They documented that at certain locations, a new nucleosome read would start every 193 bp, creating periodicity, or a phasing of nucleosomes. This is important when looking at key transcription landmarks. Polymerase and other proteins and enzymes bind to important sites in the genome, then position and phase nucleosomes extending out around them [25]. Without these factors, nucleosome positioning due to phasing is non-existent. Within the same study, the *In vitro* nucleosome data showed no phasing positioning whatsoever.

Another interesting experiment from Valouev et al. looks at histone repelling and attracting elements [30]. They found that histones prefer binding to high G/C content, and shy away from A/T content. This creates the theory of container sites, where a G/C-rich region is flanked by an A/T-rich region. The repelling elements funnel the histone octamer onto the favorable binding site, and create a well-established location for nucleosome formation. With a well-established formation site, important binding sites are kept open for polymerase and transcription factors [23,24].

Compaction of nucleosomes into higher order structures regulates protein synthesis and creates what we know as chromosomes. The first level of compaction starts with the "beads on a string" 10 nm formation where nucleosomes bind to DNA. After clumping together, a larger 30 nm fiber is created. This structure has been a topic of debate for many years, as crystal structure images show the impossibility of the 30 nm fiber and it has never been directly observed. Many other structures have been seen, from clutches of nucleosomes to Magnesium-dependent nucleosome globs with sizes ranging from 50 nm – 1000 nm [5]. These structures undergo a final level of compaction to create the chromosome structures seen in cells [8]. Nucleosome compaction can be alternatively changed through histone modifications. Attaching different molecules to histones, from methyl groups to glucose molecules, modifies their binding capabilities and preferences [45].

There are a vast number of factors that influence nucleosome formation, and consequently gene regulation. Although the amount we know now is limited, we are creating new techniques and discoveries to assist us in continued understanding of chromatin. To make a significant contribution to the field of chromatin, I conducted two hypothesis driven sets of experiments that address the topic of chromatin structure: 1) I created a technique for tissue specific nucleosome isolation and 2) I tested the validity of *in vitro* nucleosome reconstitution techniques commonly used for observing nucleosome positioning.

Chapter 2. Tissue Specific Nucleosome Isolation

2.1 Introduction

Red blood cells are the only cells within our bodies without a nucleus [15]. Every other somatic cell contains all the genetic information needed to create any protein that every other cell is capable of producing. Why then are only select proteins created from specific cell types? Why do some cells, like cardiac cells, look different than others, like neurons? Transcription factors and regulators play the largest role, but a unique part of differential gene expression stems from nucleosome occupancy and histone modification. Histone octamers wrap up the DNA in different patterns depending on cell type and tissue, and change as the organism ages or experiences environmental stimuli. This creates open transcription for important genes and transcriptional inaccessibility for non-important genes. These same genes may be necessary in other types of cells, and in those cells, differential nucleosome occupancy will keep them transcriptionally active. An example of this is of the genes encoding myosin. Myosin is a motor protein that helps with muscle contraction. It is present in vast quantities in muscle tissues, but almost non-existent in other tissue types. Through various factors, these genes are bound tightly by nucleosomes in tissues that do not need muscle contraction, but kept open and highly active in muscle tissues [16].

If we look at nucleosome occupancy across all tissues in an organism using pan-cellular bulk analysis, the nucleosome positions look uniform when mapped together. However, a 2012 study on human tissue culture lines, nucleosome occupancy analysis reveals differing underlying nucleosome patterns. Kundaje *et al*. created a Clustered AGgretation Tool (CAGT), which attempts unsupervised pattern discovery on nucleosome positioning [17]. It takes into

account “inherent heterogeneity in signal magnitude, shape, and implicit strand orientation of chromatin marks” to separate the uniform nucleosome occupancy shape into smaller groups of distinct patterns. Why is this important to experimental design in nucleosome research? A complex tool was used to separate nucleosome occupancy patterns from a uniform pattern in only one type of tissue culture cells. There are about 200 cell types in humans, all with different gene expression and varying nucleosome occupancy patterns [18]. Even smaller organisms, like *C. elegans*, which has 20 cell types (Sulston), are difficult to use in this manner because of the inability to separate cell types [19]. At only 1 mm long, it is impossible to manually isolate certain cell types from *C. elegans* with enough material for experimentation. Instead of increasing the computational complexity of the CAGT, we have created a Chromatin Immunoprecipitation (ChIP) protocol to isolate tissue specific nucleosomes. Using an antibody specific to Green Fluorescent Protein (GFP) attached to the H2B histone protein, we can pull down nucleosomes harboring GFP-labeled histones from organisms that are uniquely modified to express GFP in only one specific cell type (CGC). After the pull down of the nucleosomes, it is possible to sequence the attached DNA and use CAGT to analyze the nucleosome occupancy of that tissue type.

2.2 Materials and Methods

Worm Growth

The AZ212 strain of *C. elegans* was used for the immunoprecipitation protocol. This strain of worms expresses GFP in the gonadal tissue. As the worms hatch and age, GFP labeling can be seen in the embryo, but subsequently goes away and does not return until the young

adult stage [20]. To compensate for age variation, we synchronized the age of the worms and only used young adults for the pull-down DNA. Mixed-stage worms were grown on rich nematode growth media (RNGM) and washed off using a non-tris buffer, M9(5g NaCl; 3g KH2PO4; 6g Na2HPO4; up to 1L in H2O; Autoclave; 1ml of MgSO4). Bleach was used to kill all living worms, leaving only eggs from gravid adults. These eggs were then plated to hatch and grow starting at the same life stage. This process was done multiple times, consecutively, to closely synchronize the ages. The worms were flash frozen in M9 buffer using liquid nitrogen and left at -80^{o} until needed.

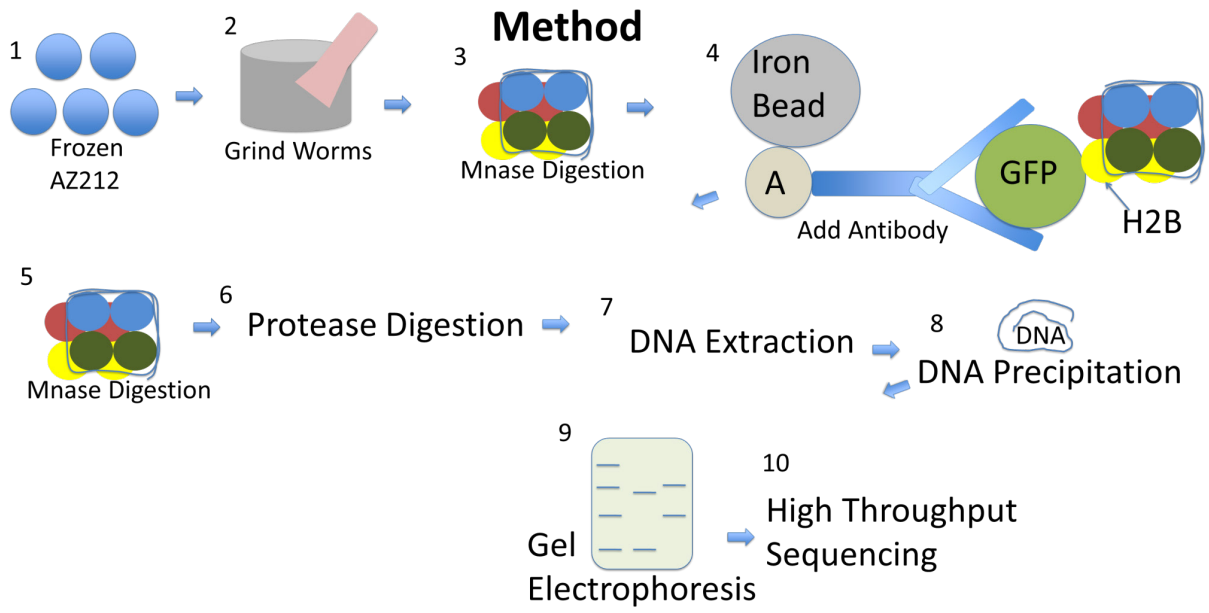


Figure 2.1 Workflow for Immunoprecipitation of Tissue Specific Nucleosomes. The C. elegans are first crushed using Liquid Nitrogen, then undergo an MNase digestion. The nucleosomes are conjugated with an anti-GFP antibody and isolated with magnetic Thermo Fisher Dynabeads. The nucleosomes undergo another MNase digestion followed by a protease digestion to digest the histone proteins. Phenol/Chloroform and Chloroform are used to extract the naked DNA and ethanol is used to precipitate it out of solution. The Sample is run on a 2% agarose gel and the mononucleosome band is cut out for High Throughput Sequencing.

Digestion and pull down

The frozen AZ212 worms were placed in liquid nitrogen and crushed into a fine powder using a mortar and pestle. The powder was added to 1mL 1x Reaction Buffer (100mM Hepes pH 7.4; 50mM MgCl2; 50mM CaCl2). Micrococcal nuclease, resuspended at 150 U/uL, was added for a final concentration of 0.75 U/uL, followed by incubation at 16^{o} for 12 min to digest DNA down to nucleosome cores. The digestion was halted by the addition of 50 uL EGTA (0.5M). Rabbit anti-GFP antibody was added to the sample and incubated overnight to ensure antibody binding. After antibody incubation, Thermo Fisher Dynabeads (30mg/ml) were added to sample and incubated together for one hour. In the sample, the Dynabeads attached to the antibody, which attached to the nucleosomes. A strong magnet (Neodymium 140 lb pull force) was used to pull down the beads (along with the string of GFP-harboring nucleosomes attached) and wash them three times using LiCl (0.8M). Reaction Buffer was added along with micrococcal nuclease in conditions seen above to digest the nucleosomes a second time. The reaction was halted with EGTA and washed again with LiCl. The beads were pulled down with a magnet, and resuspended in 400 uL Worm Lysis Buffer (0.1M Tris HCl pH 8.5, 0.1M NaCl, 1% SDS) along with 4 uL proteinase K (20 ug/mL). The solution was incubated at 65 ^{o} for 2 hours to digest the histone cores, leaving naked DNA. The DNA was extracted using a standard phenol/chloroform, chloroform extraction and precipitated using ethanol. The DNA was resuspended in TE Buffer (composition). The resulting sample was visualized on a 2% agarose gel using ethidium bromide.

2.3 Results and Discussion

The current protocol is close to completion. There is still some work to be done for antibody specification, however. The immunoprecipitation protocol is successful, but the antibody is not as specific as desired (Figure 2.2). Both samples, those with GFP, and those without, have proteins pulled down by the GFP-antibody. The only sample that should pull down any protein should be the sample with GFP-expressing histone proteins (AZ212 strain). Limiting the immunoprecipitation to only pull down the desired proteins should be the last step in developing an immunoprecipitation protocol for tissue specific histones.

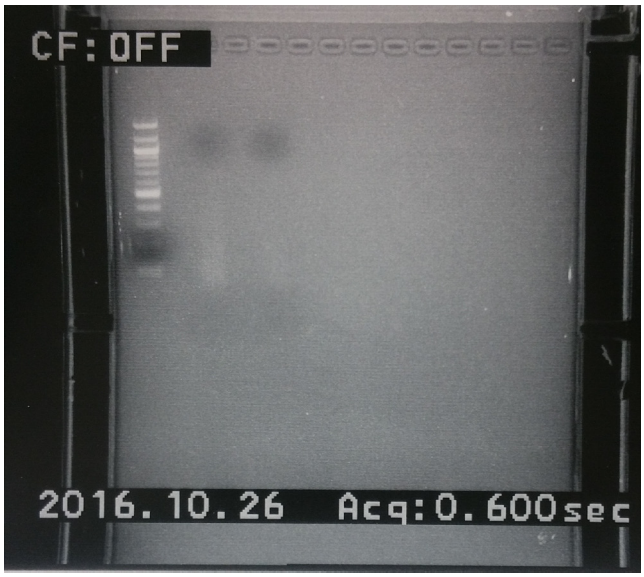


Figure 2.2 Gel Image of DNA extract from Immunoprecipitation Protocol. Lane 1: 150 bp Ladder Lane 2: N2 wild-type (no expressed GFP) C. Elegans with anti-GFP antibody, normal protocol Lane 3: AZ212 C. elegans with anti-GFP antibody, normal protocol. Lane 2 worms do not express GFP, and thus should have no DNA extract with this protocol. Only Lane 3, with GFP expressing C. elegans should have any DNA extract

It is proven that the non-specific binding can be inhibited through different blockers. There are different types of blockers used in immunoprecipitation to inhibit the antibody binding to non-specified proteins including detergent blockers (Tween-20, Triton X-100, etc.),

protein blockers (Bovine serum albumin, casein, etc.), and polymer based blockers (Polyethylene glycol, Polyvinyl alcohol, etc.) [21,22]. By adding one of these blockers, the antibody should be inhibited from non-specific binding, and only pull out GFP-labeled histones. The protocol should be complete once the blocker can be used for better antibody binding.

2.4 Future Directions

This protocol can be used for many different applications. The first planned application of this technology is for research into single nucleotide polymorphisms (SNPs) and their effect on histone placement. There are two strains of *C. elegans*, Bristol and Hawaiian, which have ~100,000 SNP differences between them. Our lab has stable mutant lines, both Bristol and Hawaiian, which express GFP on the histone H2B subunit, exclusively in gonadal tissue. The immunoprecipitation protocol could be used on both strains to look at histone placement between the two. This will look at nucleosome placement only in the gonad for both strains. Without this protocol, the data would become messy with nucleosomes from many different tissue types and occupancy profiles. If there is any difference in histone occupancy rates, it can be attributed to SNPs. Other applications of this technology may include investigation into differences in histone occupancy between same-organism tissues or investigation into gene silencing.

Chapter 3. Efficient Recovery of Lost Invitrosomes Through Comparative Defined-End Analysis

3.1 Introduction

Access to the nucleotide sequence by trans-acting factors is primarily determined by nucleosome positions within the immediate chromatin architecture. Several things can direct and regulate nucleosome positions, including the underlying nucleotide sequence itself [23,24]. Whole-genome approaches, looking at nucleosome formation and positioning, have shown that DNA sequences with high affinity for nucleosome formation often contain dinucleotide motifs with a 10-11 base pair (bp) periodic repetition that can have a significant influence on the rotational setting and positioning of nucleosomes [25-27]. It is thought that this periodicity helps to form and stabilize a favorable conformation of DNA around the histone octamer core by minimizing the energetic costs of bending the DNA to form nucleosomes [28]. Additionally, more recent studies have shown that high G/C-content DNA fragments are also favorable to nucleosome formation while homopolymeric runs of A's and T's and high A/T content in general are recalcitrant to nucleosome formation and are often found in the linker region between well-positioned nucleosomes [29,30].

A commonly used method for determining high-nucleosome-affinity DNA sequences is through the use of *in vitro* nucleosome reconstitutions. Whole-genome applications of this method begin with isolation of naked genomic DNA followed by generation of smaller DNA fragments primarily through sonic sheering or restriction enzyme digestion of the high-molecular-weight DNA. Recombinant or isolated histone octamers and DNA fragments are then added together in high-salt solution in a stoichiometric ratio such that on average a single

nucleosome will form on each individual DNA fragment. The salts in the solution are then dialyzed away, allowing the formation of nucleosomes [31,32]. Nucleosome positions from the *in vitro* reconstituted assemblies can be compared to their *in vivo* genomic equivalents, allowing for the identification of high-nucleosome-affinity sequences determined exclusively by intrinsic DNA sequences. The assemblies can also determine the amount of *in vivo* remodeling that occurs within individual cell or tissue types. Such an approach was used by Locke et al. to demonstrate the extent of nucleosome remodeling that occurs *in vivo* to the *C. elegans* genome [24].

While *in vitro* nucleosome reconstitutions provide valuable information, the technique contains at least one inherent bias that must be overcome to fully use the derived data. It has been demonstrated that DNA-fragment ends can influence nucleosome formation, encouraging end-proximal nucleosome formation relative to the remainder of the DNA fragment [33,34]. This preference is termed fragment-end bias and can introduce a major hurdle when attempting to identify high-nucleosome-affinity DNA sequences. Because of this major bias, in any *in vitro* nucleosome reconstitution experiment, it becomes impossible to determine if *in vitro* nucleosome (invitrosome) [35] formation was due to fragment-end bias or an actual affinity for the underlying nucleotide sequence.

It is thought that this end bias can be overcome by using sonication to generate the needed DNA fragments. In theory, if DNA fragmentation by sonication were random and invitrosome sequencing were sufficiently deep, the bias from fragment-end affinity would be eliminated. Sonication would produce random fragment ends in excess across the entire

sample, and sufficient deep sequencing of invitrosomes would create a uniform background coverage of end-biased nucleosome reads (Figure 3.1). True positive enrichment on invitrosomes due to high-nucleosome-affinity DNA sequences would be seen above this uniform background (Figure 3.1 A). If sequencing depth were not sufficient, end-biased nucleosome reads would not provide a uniform background that could be subtracted out, and end-biased reads could produce false positive peaks or hide true peaks (Figure 3.1 B). Additionally, recent papers have revealed that sonication produces DNA breaks preferably at AT-rich regions and specific di- and tetra-nucleotide sequences [38]. This can create a fragmentation bias, meaning sonication may not be the solution to the fragment-end-bias dilemma.

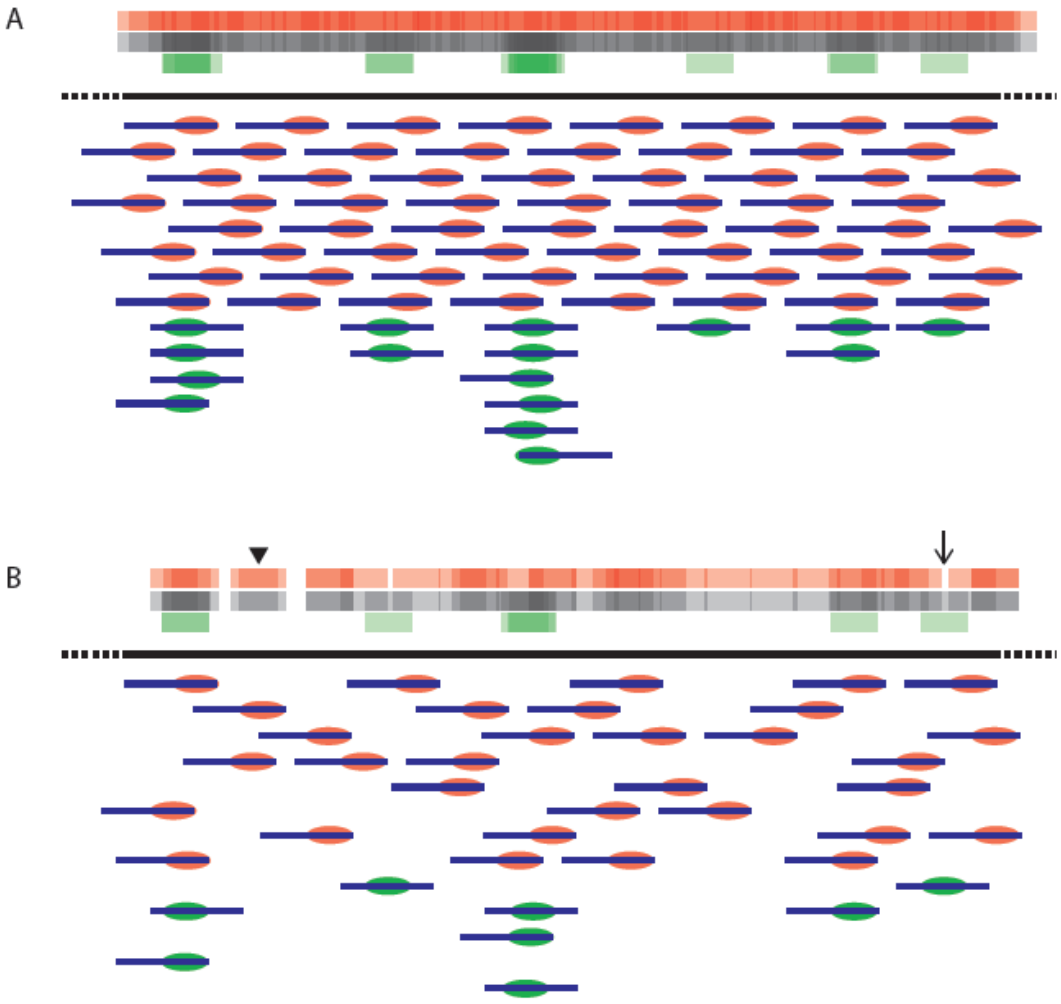


Figure 3.1 Effect of end bias on detection of nucleosome positioning and occupancy. Both A and B depict the same hypothetical genomic locus (black line) where random, unbiased DNA fragments (blue lines) are reconstituted into invitrosomes that either have end-bias (red ovals) or are unbiased/formed due to DNA sequence affinity (green ovals). The total occupancy and inferred positioning of nucleosomes are depicted by the red bars over the genomic locus (end-biased invitrosomes), the green bars over the locus (unbiased invitrosomes), or the grey bars, a combination of the red and green bars (all invitrosomes; both end-biased and unbiased combined). A is an example of random shearing and sufficiently deep coverage resulting in relatively uniform end-biased-nucleosome coverage (red bar). Much of the pattern from the green bars, representing invitrosome formation due to sequence affinity, can still be detected above the background in the total invitrosome data (grey bars). B is the same as figure A except with half of the reads removed, resulting in non-uniform coverage of the end-biased-nucleosome (red bars). The combination of end-biased and unbiased invitrosome coverage results in the obfuscation of some of the green pattern (arrow) and conflation of end-biased (arrow head) and sequence-positioned invitrosome occupancy. In both A and B, "random" DNA fragments are depicted by a uniform size for simplicity and end-bias is only show on one fragment end.

One option to overcome this hurdle is to discard nucleosome positions that fall near DNA fragment ends. This is only possible if the ends of the DNA fragments used in the reconstitution experiments are known. In this case, the amount of data discarded is normally a significant portion of the potentially meaningful data. This presents a major limitation to nucleosome reconstitution, as it requires an excessive amount of time and materials and sequencing to generate enough usable data once end-proximal nucleosome positions are discarded. Such an approach to overcome potential end-bias was used by Locke et al. in their analysis [24].

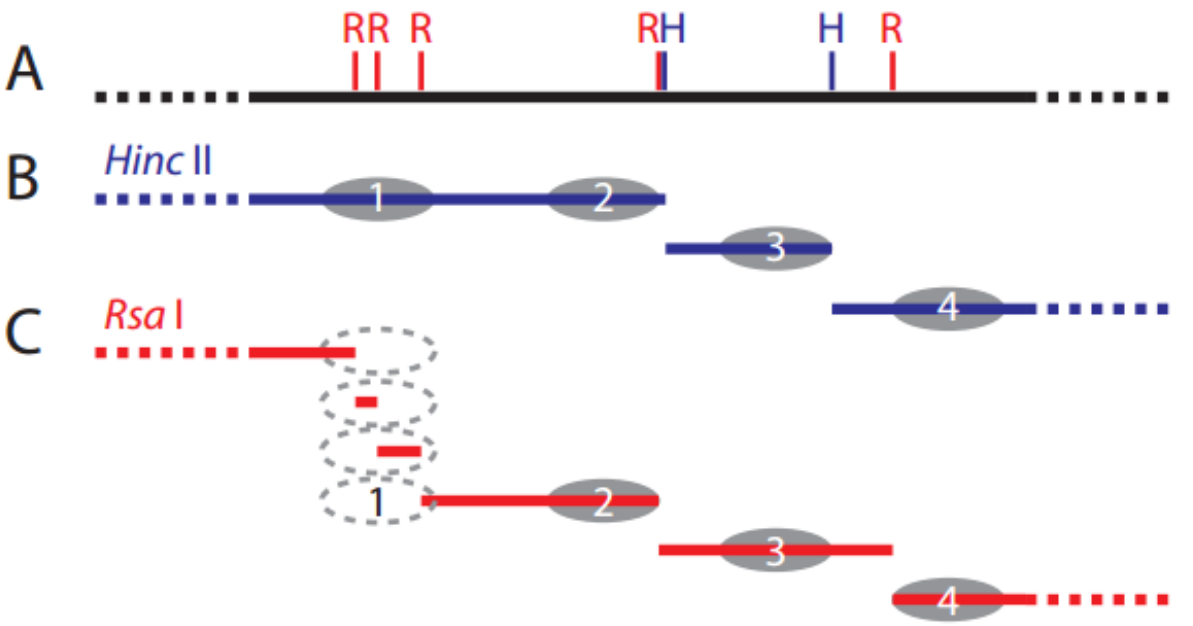
In the following study we present a novel approach for addressing fragment-end bias that eliminates the need to discard large portions of the data produced in these types of experiments. We apply our approach to the Locke et al. data set and show that we can recover up to 90.8% of the discarded data.

3.2 Approach

Currently using conventional approaches, two classes of DNA loci are typically excluded from invitrosome analyses or have invitrosomes discarded in order to eliminate potential end bias. When DNA-fragment ends are defined, 1) any invitrosome found to map within a defined number of nucleotides from a fragment end is classified as suspect of fragment-end bias and is discarded. 2) DNA fragments digested to sizes too small for reconstitution (<147 bp) are lost from invitrosome analyses.

Both of these classes of excluded loci can potentially be recovered and analyzed by performing nucleosome reconstitutions on two DNA samples digested by two different

restriction endonucleases. Our approach is such that each individually-digested DNA sample is used for separate nucleosome reconstitutions. Invitrosome positions from the two experiments are identified by mapping sequenced mononucleosome core DNAs back to the original source of DNA. For each individual reconstitution experiment, invitrosomes that may suffer from end-effect bias can be identified by defining a specific number of bases from restriction-enzyme cut sites as “too close” to the end of the DNA fragment (the suspect range). Invitrosomes that map within suspect-range regions are considered theoretically subject to fragment-end bias and so are defined as “suspect” nucleosomes. Invitrosomes that do not fall within the suspect-range regions are assumed to not be affected by fragment-end bias and are classified as “passed” nucleosomes.



D Nucleosome Positions Passed (✓) Suspect (?) or Lost (X)

Position	1	2	3	4
Hinc II only	✓	?	?	✓
Rsa I only	X	?	✓	?
Recovered	✓	?	✓	✓

Figure 3.2 Recovery of lost or suspect invitrosomes. A, a genomic locus (black bar) with Rsa I (R) or Hinc II (H) restriction sites depicted in red and blue respectively. B, Hinc II-digested DNA fragments (blue) harboring theoretical invitrosomes (grey ovals). C, Rsa I-digested DNA fragments (red) harboring theoretical invitrosomes (grey ovals). D, table summarizing the status of each theoretical invitrosome after comparative defined-end analysis, and the contribution of the Hinc II and Rsa I experiments to the final status. Each invitrosome is declared as passed, suspect, lost or recovered based on the combined results of both invitrosome sets.

The restriction sites of the two restriction endonucleases used will usually not be near one another on the DNA. Therefore, invitrosomes from one experiment that are defined as suspect and normally would be discarded (due to proximity to a fragment end) can be recovered if the same locus is found to be occupied by a passed invitrosome in the second experiment. This is demonstrated in Figure 3.2 with the example invitrosomes in position 3 and position 4. In contrast, invitrosomes in position 2 remain in doubt as this position is near a fragment end in both experiments and both invitrosomes are suspect.

Additionally, the positions where DNA fragments were generated that were too small to participate in reconstitution can be recovered. As the likelihood of this happening with both endonuclease digestions is small; a position lost in one experiment can be recovered if in the second experiment the fragment is of sufficient size to form a "passed" invitrosome (e.g. Figure 3.2 position 1).

In order to validate our approach, we have applied our recovery methods to the invitrosome datasets generated using the *Caenorhabditis elegans* genome described in Locke et al. The suspect range used by Locke et al was 200 bp, which is very large considering the *Rsa I* enzyme creates fragments at an average of ~490bp. Using the same strict suspect range of 200 bp from DNA-fragment ends used in the Locke analysis, we find that we can recover the vast majority of suspect, discarded invitrosome positions. As the suspect range is decreased, the recovery rate increases proportionately (Tables A.1/A,2). Percent recovery is also dependent on the number of total invitrosome position generated in both datasets. These results show that

our method is capable of preventing the massive loss of data by which current nucleosome reconstitution studies are limited.

3.3 Materials and Methods

Mapping and preprocessing of reads

The 9.5 million *Rsa I* and 5.3 million *Hinc II* raw 36-bp read libraries (up to 147 bp reads are practical) used in the Locke analysis were mapped to the WS190 version of the *C. elegans* genome using Bowtie (version 1.1.2) [44]. Parameters were set so that a maximum of one mismatch per read was allowed and the best match position was used. Unnecessary columns were suppressed to output only later used data, while all other parameters were set to default. 871,964 and 509,696 reads were eliminated from the *Rsa I* and the *Hinc II* libraries respectively at this point because they failed to map back to the template, had more than one maximal alignment of equal score or failed to meet the required stringency. The reads were then separated into individual data structures for each chromosome. This separation helped to keep the data structures small and speed up the processing time.

Defining Suspect regions

The first step in our recovery approach is the generation of a suspect-range size based as a user-defined variable. To generate a suspect-range region, the exact genomic positions of the restriction-enzyme-digested fragment starts and ends are required. We were able to define fragment starts and ends by use the fragment-end tables generated by Locke et al. and available at http://nucleosome.rutgers.edu/nucenergen/celegansnuc/xfer [24]. These tables

contain the start, end and fragment size of all hypothetical fragments generated across all chromosomes by both the *Rsa I* and *Hinc II* restriction endonucleases. However, the palindromic cuts sites are not included in the provided start/end positions. The size of the suspect range is determined by the number of bases from each fragment end that should be considered to be subject to fragment-end bias. In the Locke analysis this was defined to be 200 bp from a fragment end and 200 bp from the fragment start, for a total exclusion of 400 bps per restriction fragment. For the purpose of assessing our approach, we defined multiple suspect ranges beginning at a minimum suspect range of a single bp and increasing to 5 bp, 11 bp (one helical turn of DNA), and then by increments of 11 bp until a maximum suspect range of 200 bp was reached. Each nucleosome read was converted into a genomic position and the center of the nucleosome (dyad) position was saved. To generate each genomic suspect-range region, each dyad position had 73 bases added or subtracted to simulate ends, and then the defined number of base pairs added or subtracted to find the suspect region. The sets of positions found in the data structures define all possible suspect ranges and allowed us to separate suspect reads from the remainder of the fragments.

Defining Suspects and Passed nucleosomes

The second step in our approach is to define suspect and passed nucleosomes using the suspect-range regions defined in the previous step. All *Rsa I*- and *Hinc II*-invitrosome reads post processing were compared to and defined by their location relative to the suspect-range regions. The reads were classified as either suspect or passed. If a read was found to begin within the suspect-range region, it was classified as a suspect invitrosome. The sense of the

read was taken into account when this comparison was made as described above. All invitrosomes that do not receive this classification are considered passed because they did not fall within the suspect-range region. Underdigested invitrosome reads were also classified and separated at this point. Once defined, the three classifications were separated into six separate data structures: *Hinc II*-suspect invitrosomes, *Hinc II*-passed invitrosomes, *Hinc II*-InnerCutSite (underdigested reads), *Rsa I*-suspect invitrosomes, *Rsa I*-passed invitrosomes, and *Rsa I*-InnerCutSite.

Recovery of suspect nucleosomes

The final step in our approach is recovery of suspect invitrosomes by comparison to the alternate experiment's set of passed invitrosomes. Suspect-*Rsa I* invitrosomes were compared to passed-*Hinc II* invitrosomes and InnerCutSite-*Hinc II* invitrosomes, while suspect-*Hinc II* invitrosomes were compared to passed-*Rsa I* invitrosomes and InnerCutSite-*Rsa I* invitrosomes. Suspect invitrosomes that sit at the same position as passed invitrosomes or InnerCutSite invitrosomes in the alternate fragment set were classified as recovered. Those that do not receive this new classification were considered biased invitrosomes. The final result was a set of recovered, biased and InnerCutSite invitrosomes for each fragment set. The custom Java scripts used in these analyses are available upon request.

3.4 Results

Discarded Invitrosomes

The Locke et al. datasets we use in our analysis were derived from invitrosomes formed on *C. elegans* genomic DNA. In their analysis, two separate invitrosome data sets were made by reconstituting invitrosomes on *C. elegans* genomic DNA that had been digested with either *Rsa I* (a blunt, four-cutter) or with *Hinc II* (a blunt, five-cutter). Invitrosome-core DNA was isolated using micrococcal nuclease (MNase) to ensure unwrapped DNA was digested and then sequenced on the Illumina platform. The resulting *Rsa I*-reconstitution experiment produced a total of 9.5 million raw sequencing reads, while the resulting *Hinc II*-reconstitution experiment produced a total of 5.3 million raw sequencing reads. To control for invitrosomes positioned due to end effects, Locke et al. defined a 200-bp suspect range from each restriction enzyme cut site. In the *C. elegans* genome, *Rsa I* cuts on average once per 490 bp, and *Hinc II* cuts on average once every 2109 bp [24] Use of their 200-bp suspect range resulted in excluding 87.7% of the bps in the *C. elegans* genome for the *Rsa I* dataset and 19.0% of genomic bps for the *Hinc II* dataset [24], an alarmingly large portion of the genome.

We hypothesized that we could recover a significant portion of invitrosome positions lost to the Locke analysis by applying our recovery approach. Thus we used the 9.5 million *Rsa I* raw sequencing reads and the 5.3 million *Hinc II* raw sequencing reads from Locke et al. in our analysis.

Pre-processing of reads

Because we were using raw reads, it was necessary to eliminate poor-quality reads and reads that mapped to multiple loci before our approach could be applied. The raw reads were mapped back to the WS190 version of the *C. elegans* genome using Bowtie. Parameters were set to only use reads with very high alignment scores (maximum of one mismatch). A number of reads from both data sets mapped to multiple sites within the genome. As our approach assumes one position per read, these multiple alignments were processed so only the match with the highest alignment score was retained. Using these parameters, a total of 8.6 million (90.8%) of the original *Rsa I*-generated sequence reads mapped back to the genome, while a total of 4.8 million (90.5%) of the original *Hinc II* sequence reads mapped to the genome (Figure 3.3).

Application of Recovery Approach

Our recovery approach is composed of three steps. 1) a suspect range is generated as a user-defined variable, 2) invitrosomes are mapped and declared either passed or suspect, and 3) suspect invitrosomes are recovered by comparison to the alternate experiment's set of passed invitrosomes.

In applying the first step, generation of suspect-range regions is dependent on knowing precise fragment ends produced by restriction enzyme digestion. Because two different restriction endonucleases are used, the loci that fall into the suspect-range regions will be different for the two experiments and will depend on the restriction enzyme used to prepare the template DNA for reconstitution. In applying this step, we used the fragment-end list

generated by Locke et al., which defines the beginning and end of DNA fragments based on the presence of either a *Rsa I* or a *Hinc II* cut site. This list shows all hypothetical fragments generated across all chromosomes by digestion with these enzymes [24]. In the Locke analysis the suspect range was defined as 200 bp in either direction from a restriction digest site, for a total range of 400 bps per DNA fragment. We used a 200-bp suspect range to match the results of the Locke analysis. To generate each suspect-range region, the genomic position of each DNA-fragment start or DNA-fragment end (excluding the palindromic restriction enzyme cut site) had the suspect range-defined number of base pairs added to or subtracted from it respectively, producing suspect-range-defined starts or ends. This resulted in unique sets of suspect-range regions for each restriction enzyme.

We applied the second step of our approach by first mapping all the invitrosome sequence reads from both experiments to the WS190 version of the *C. elegans* genome. After mapping the sequence reads, each read was extended out to 147 bp to represent the entire footprint of the invitrosome from which it was derived, and the direct center, or dyad position, was recorded to produce sets of both *Hinc II*-invitrosome dyads and *Rsa I*-invitrosome dyads. During analysis, the dyad positions (exact center of the 147 bp nucleosome) were pushed out 73 bp in both directions to produce start and end positions for both sets and were then compared to their respective suspect-range regions. Depending on where each invitrosome fell relative to the suspect-range regions (within the suspect range or outside of the suspect range), it was defined as either “suspect” or “passed” respectively. Any invitrosome with a start that fell into suspect-range start region or any invitrosome with an end that fell into a suspect-range end region was defined as suspect. Passed invitrosome positions were separated from suspect

invitrosome positionsc and kept as good data for each experiment. Underdigested fragments which had invitrosome reads which tiled over cut sites were also considered passed but kept separate for statistical purposes (InnerCutSite). For each experiment the suspect-range size was kept the same between the *Rsa I* and the *Hinc II* datasets. This resulted in six invitrosome sets from the two experiments: passed-*Rsa I* invitrosomes, suspect-*Rsa I* invitrosomes, InnerCutSite-*Rsa I* invitrosomes, passed-*Hinc II* invitrosomes, suspect-*Hinc II* invitrosomes, and InnerCutSite-*Hinc II* invitrosomes.

The final step was to recover suspect invitrosomes from one experiment and reclassify them as free of end-effect bias through comparison with passed invitrosome reads from the alternate experiment.

Suspect-*Rsa I* invitrosomes were compared to passed-*Hinc II* invitrosomes and InnerCutSite-*Hinc II* invitrosomes, while suspect-*Hinc II* invitrosomes were compared to passed-RsaI invitrosomes and InnerCutSite-*Rsa I* invitrosomes. Suspect invitrosomes that sit at the same position as passed invitrosomes in the alternative fragment set were now reclassified as "recovered" invitrosomes. Those that did not receive this new classification are considered to be potentially affected by end bias and were reclassified as "biased" invitrosomes. The final result is a set of recovered and biased invitrosomes for each fragment set. The results generated by step two and this step were eight unique output files: passed-*Rsa I* invitrosomes, InnerCutSite-*Rsa I* invitrosomes, recovered-*Rsa I* invitrosomes, biased-*Rsa I* invitrosomes, passed-*Hinc II* invitrosomes, InnerCutSite-*Hinc II* invitrosomes, recovered-*Hinc II* invitrosomes, and biased-*Hinc II* invitrosomes (Tables A.1/A.2). The complete workflow is shown in Figure 3.3.

Recovery of *Rsa I* and *Hinc II* Invitrosomes

The mapped *Rsa I* dataset contained a total of 8,617,075 invitrosomes (Table A.1 and Figure 3.3). Using our 200-bp suspect range; 5,015,478 or 58.2% of the mapped *Rsa I* invitrosomes were declared suspect (Figure 3.3). Without our recovery method these suspect invitrosomes would be lost to further analysis. This is substantially lower than the number excluded from the Locke et al. analysis, but still a very large portion of the data. The possible reasons for this discrepancy will be discussed below (see Discussion).

In order to recover suspect-*Rsa I* invitrosomes we compared these invitrosomes to the passed-*Hinc II* invitrosomes that were analyzed at the *Hinc II* 200-bp suspect range. As described above, any suspect-*Rsa I* invitrosome that shared the same position with a passed-*Hinc II* invitrosome was assumed to be an invitrosome that formed at that particular locus due to preferable DNA sequence rather than end-position bias and was declared recovered. This comparison resulted in 2,492,674 (49.7%) of the suspect-*Rsa I* invitrosomes being reclassified as recovered through comparison, and 514,960 (10.3%) invitrosomes being reclassified as underdigested recovered (InnerCutSite). Thus using our recovery method we recovered 60.0% of the suspect-*Rsa I* invitrosomes resulting in a total of 6,609,231 passed- or recovered-*Rsa I* invitrosomes, or 76.7% of the original invitrosome set. This left 2,007,844 suspect invitrosomes that were reclassified as biased and unusable, 23.3% of the original *Rsa I* invitrosome set, instead of the 58.2% that would be unusable without our recovery procedure (Figure 3.3).

The same analysis was performed on the 4,848,298 mapped *Hinc II* invitrosomes, with recovery analysis being performed with the passed-*Rsa I* invitrosomes that were analyzed at

the *Rsa I* 200-bp suspect range. At the suspect range of 200 bp; 902,261 or 18.6% of the *Hinc II* invitrosomes were declared suspect (Figure 3.3). Using the passed-*Rsa I* invitrosomes, 386,545 suspect-*Hinc II* invitrosomes were recovered through comparison and 514,960 suspect-*Hinc II* invitrosomes were recovered through underdigested recovery. The remaining 476,550 (52.8%) suspect-*Hinc II* invitrosomes were labeled as biased. Thus we recovered 42.8% of the suspect-*Hinc II* invitrosomes through comparison and 4.3% of the suspect-*Hinc II* invitrosomes through underdigested recovery, for a total of 47.1% of the biased reads recovered. A total of 4,371,748 (90.2%) passed- or recovered-*Hinc II* invitrosomes, of the original invitrosome set (Figure 3.3) were usable after our recovery approach. The remaining 476,550 biased invitrosomes represent 9.8% of the original *Hinc II* invitrosome set that was still unusable (Figure 3.3). Despite the more modest size of this recovery, it still represents a substantial improvement over the 18.6% that would be unusable without our recovery procedure.

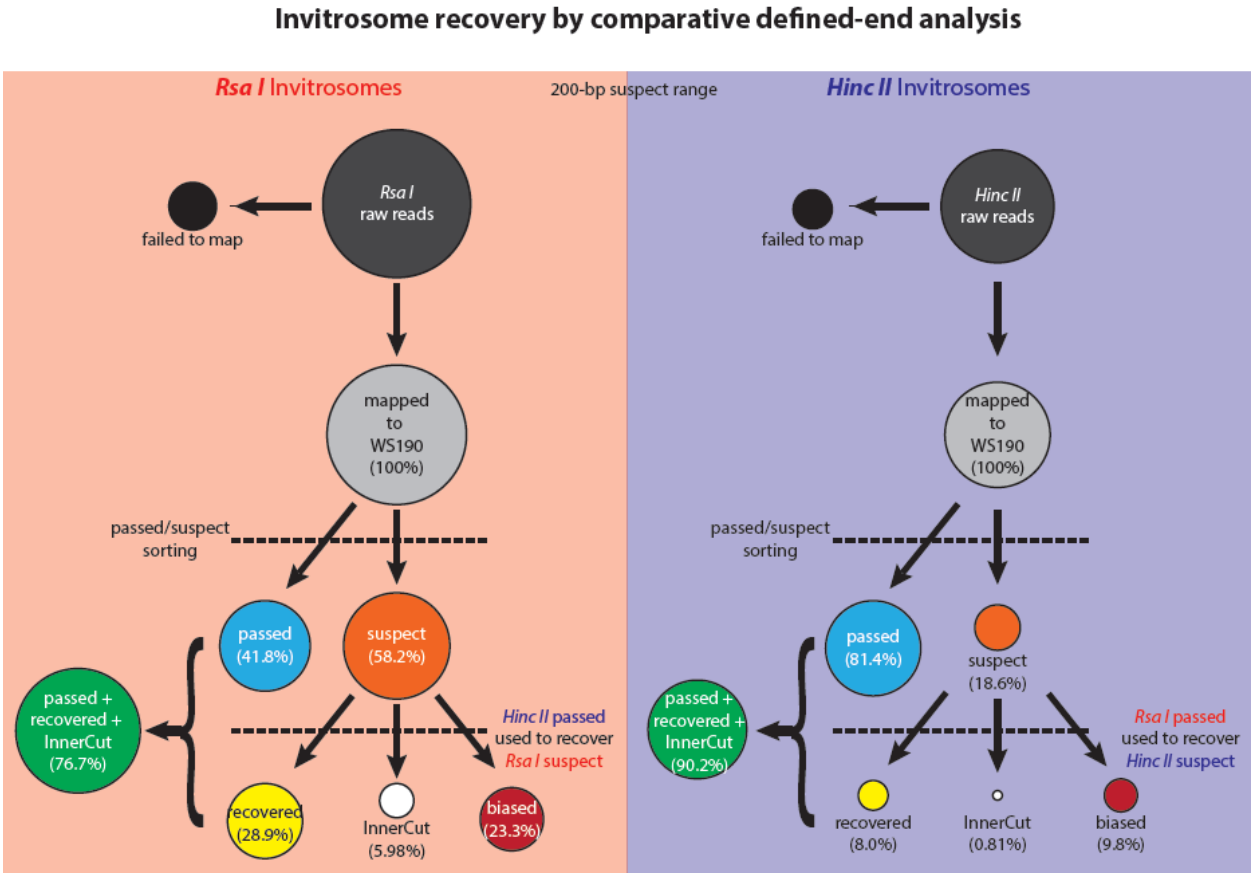


Figure 3.3 Invitrosome recovery by comparative defined-end analysis. The workflow for the recovery of invitrosomes with raw reads (dark grey circle) being mapped (light grey circle), and then declared passed (blue circle) or suspect (orange circle). Suspect invitrosomes are then declared recovered (yellow circle), InnerCut (white circle) or biased (dark read circle) using the passed invitrosomes from the alternative experiment. The total of useable invitrosomes (green circle) is the sum of the passed, recovered and InnerCut invitrosomes. Left, workflow applied to the Rsa I invitrosomes (light red background), and Right, workflow being applied to the Hinc II invitrosomes (light blue background). In both panels the size of the circles is proportional to the total number of invitrosomes at each step and the percent in each circle is the percent relative to the total number of mapped invitrosomes.

Varying the suspect range length

We wanted to test the effect of varying lengths of suspect ranges on the number of invitrosomes declared suspect and recovered by our approach. To this end, we applied 19 more suspect ranges beginning with 1 bp, 5 bp, 11 bp (one helical turn of DNA) and then increasing by 11 bp until reaching 187 bp. We compared the results of applying these additional 19 suspect ranges to the results from our maximum 200-bp suspect range. As expected, with decreased suspect range we see a decrease in the number of suspect invitrosomes. Specifically, we see the number of suspect invitrosomes decrease in relation to the length of the suspect range, and the lowest suspect range of a single base pair resulting in a low of only 737,064 (8.6%) of the *Rsa I* and 70,425 (1.5%) of the *Hinc II* invitrosomes being declared suspect respectively (Tables A.1 & A.2). It is interesting to note that for *Rsa I* invitrosomes, at the larger suspect ranges (≥154 bp) we observed that the number of suspect invitrosomes is actually greater than the passed invitrosomes. This is not the case for the *Hinc II* invitrosomes. All of these trends are demonstrated in Tables A.1 & A.2.

One suspect range is of particular interest. The 11-bp suspect range represents one full turn of the DNA helix. If invitrosomes were to be affected by end bias, but still try and retain a preferential rotational setting, it might be predicted that they would form between 1-11 bp from the end of the DNA fragment as this would cover all potential rotational settings. Previous studies have demonstrated that virtually all end-effect nucleosome positioning results in invitrosomes within about ±10 bp of the fragment end [34]. At the 11-bp suspect range

1,235,064 (14.3%) of *Rsa I* invitrosomes are suspect and 124,076 (2.6%) of *Hinc II* invitrosomes are suspect (Tables A.1 & A.2). At this same level, 999,958 (80.9%) of the suspect-*Rsa I* are recovered, with 484,998 saved through comparison and 514,960 saved through underdigested comparison (Table A.1). 105,377 (84.9%) of suspect-*Hinc II* invitrosomes are recovered, with 66,211 through comparison and 39,166 through underdigested comparison (Table A.2).

Having applied our approach, we find that a substantial number of suspect invitrosomes can be recovered within the *Rsa I* invitrosome set no matter what size the suspect range is. Within the maximum 200-bp suspect range we find that our approach is able to recover 60.0% or 3,007,634 of the suspect-*Rsa I* invitrosomes. However, with the smaller 11-bp suspect range, we are able to recover 80.9% or 999,958 of the suspect-*Rsa I* invitrosomes.

It should be noted that in the Locke analysis a 11-bp window was used when mapping the invitrosomes back to the genome. In all our previously described analyses we have used this same 11-bp allowance when recovering suspect invitrosomes. That is to say, we reclassified a suspect invitrosome as recovered if the footprint of the suspect invitrosome overlapped with the footprint of a passed invitrosome from the alternative invitrosome set, effectively mapping within 11 bp (one helical turn). When this allowance is removed and an exact overlap is required, all previous described trends remain the same. The observable difference in actual recovery rates decrease by 14.8% to 21.7% for the *Rsa I* analyses and 13.1% to 15.4% for the *Hinc II* analyses across all suspect ranges, with the exception of the 1-bp suspect ranges where the decrease is 9.9% and 8.9% for *Rsa I* and *Hinc II* invitrosomes respectively (Table A.3).

3.5 Discussion

Our findings can be summarized in the statement of a few observed trends. First, recovery is most efficient when the suspect range is minimized. However, even when a very large suspect range is set (e.g. 200 bp), recovery is still significant. Of the two datasets, the *Rsa I* dataset achieves the greatest amount of recovery, but this is as expected since it contained the larger number of suspect nucleosomes to begin with and included the largest portion of the genome within the suspect-range region. The *Hinc II* dataset, in contrast, had a much lower recovery rate, but also far fewer suspect invitrosomes. When the stringency of recovery was increased (i.e., when a perfect alignment was required for a suspect invitrosome to become a passed invitrosome), all observed trends in suspect nucleosome definition and recovery rate remain the same for both fragment sets.

As noted above, 5,015,478 or 58.2% of the *Rsa I* invitrosomes were declared suspect in our analysis at the 200-bp suspect range used by Locke et al., which is substantially lower than the number excluded due to suspected end bias from the Locke et al. analysis itself. Since at a suspect range of 200 bp, 87.7% of the genome falls into a suspect-range region due to its proximity to an *Rsa I* restriction site, it might be expected that at least 87.7% of the invitrosome reads would be declared suspect even ignoring any expected increase due to end bias. This assumes that invitrosomes are reconstituted evenly over the entire genome, which is supported by data from the Locke analysis [24]. Thus there are two possible explanations for the difference between our number of suspect invitrosomes and the Locke analysis.

First, in the Locke analysis, the even distribution of invitrosomes across the genome is

demonstrated with invitrosome data that has already excluded suspect invitrosomes and genomic regions close to fragment ends. It is possible that invitrosomes are actually disproportionately found on the middle of DNA fragments (fragment-middle bias) as compared to ends of DNA fragments, and the Locke analysis does not see this because of the exclusion of genomic regions near DNA-fragment ends. This would explain why our analysis results in fewer suspect invitrosomes than would be expected. This explanation seems unlikely due to results of other genome-wide *in vitro* nucleosome experiments where fragment ends are not considered or excluded yet this proposed fragment-middle bias has not been observed [8]. Despite this lack of support, the idea of fragment-middle bias is nonetheless an intriguing possibility in light of the differential *in vivo* nucleosome coverage over different parts of *C. elegans* chromosomes as shown by the Locke analysis itself [24].

The second, and more likely, explanation is that the methods of declaring suspect invitrosomes were different between our analysis and the Locke analysis. In the Locke analysis they removed any invitrosomes that fell into the filtered regions of the genome (87.7% and 19% of the genome for *Rsa I*- and *Hinc II*-digested genomes respectively) [24]. In our analysis, we declared invitrosome reads "suspect" if their starts fell within the start-suspect-range region of the genome or their ends fell within the end-suspect-range region. The Locke analysis filter would also remove invitrosomes which formed on DNA fragments that were underdigested and actually contained an *Rsa I* or *Hinc II* cut site within the nucleosome; whereas our analysis, because it compares the ends of the nucleosomes to restriction cut sites, would keep such reads (our InnerCutSite invitrosomes). For *Rsa I*-digested genomic fragments that were used for invitrosome formation, Locke et al. reported that the theoretical average size of an *Rsa I*

fragment should be 490 bp but the observed average size was ~850 bp [24]. This underdigestion would actually provide a significant number of genomic fragments that had intact *Rsa I* restriction sites that could form invitrosomes and produce invitrosomes harboring *Rsa I* sites. These *Rsa I*-harboring invitrosomes would be declared passed in our analysis as described above but would be excluded from the Locke analysis. Thus, because of these differences in defining suspect invitrosomes we find a significantly smaller percent of suspect-*Rsa I* invitrosomes compared to the Locke analysis. This subtle difference in declaring suspect invitrosomes between our method and that of the Locke analysis actually results in an extra 29.5% of the invitrosomes being usable data by itself (87.7% - 58.2% = 29.5%). This suggests that rather than being an undesirable experimental foible, underdigestion in comparative defined-end invitrosome reconstitutions experiments might actually be an optimal part of the procedure by creating more usable reads that span over normal cut sites.

This work represents a unique method of recovering *in vitro* nucleosome-reconstitution data lost to end bias. The Locke et al. data analyzed in this study were the first genome-wide invitrosome datasets with clearly defined ends of genomic-DNA fragments used for *in vitro* nucleosome reconstitution. Having fragment-end data allows detection of invitrosomes at DNA-fragment ends and thus throws these data into suspicion because of potential end bias; but at the same time, this same information allows for very significant recovery of these data using our novel recovery method. Since one of the major goals of *in vitro* nucleosome reconstitution experiments is to define the sequence preferences intrinsic to DNA and histone octamer interactions, it is imperative that results be free of potential end biases that could easily be interpreted as bona fide sequence preferences. Thus our method of using defined-end DNA

fragments on genome-wide analyses coupled with recovery of lost suspect data is one method that could rectify these problems and be used to better define the fundamental sequence preferences that influence nucleosome formation both *in vitro* and *in vivo*.

3.6 Future Directions

We have prepared this work for publication and a complete manuscript has been written. This work is the first of its kind and represents a unique method of recovering *in vitro* nucleosome reconstitution data. The data utilized in this study was the first to generate genome-wide invitrosome datasets with clearly defined fragment ends. While having defined ends does introduce known bias, this same information allows for very significant recovery of these types of datasets. We plan to submit this publication to a yet undecided journal by December 1st, 2016 and believe it will be a new standard for genome-wide invitrosome analysis.

Bibliography

1. Venter, J. C. *et al.* The sequence of the human genome. *Science (80-.).* 291, 1304+ (2001).

2. Blundell, D. J. The legacy of the European Geotraverse. *TECTONOPHYSICS* 314, 7–16 (1999).

3. CLEGG, J. S. PROPERTIES AND METABOLISM OF THE AQUEOUS CYTOPLASM AND ITS BOUNDARIES. *Am. J. Physiol.* 246, R133–R151 (1984).

4. KORNBERG, R. D. CHROMATIN STRUCTURE - REPEATING UNIT OF HISTONES AND DNA. *Science (80-.).* 184, 868–871 (1974).

5. Struhl, K. & Segal, E. Determinants of nucleosome positioning. *Nat. Struct. Mol. Biol.* 20, 267–273 (2013).

6. PERRY, C. A., DADD, C. A., ALLIS, C. D. & ANNUNZIATO, A. T. ANALYSIS OF NUCLEOSOME ASSEMBLY AND HISTONE EXCHANGE USING ANTIBODIES SPECIFIC FOR ACETYLATED H4. *Biochemistry* 32, 13605–13614 (1993).

7. Luger, K., Mader, A. W., Richmond, R. K., Sargent, D. F. & Richmond, T. J. Crystal structure of the nucleosome core particle at 2.8 angstrom resolution. *Nature* 389, 251–260 (1997).

8. Bednar, J. *et al.* Nucleosomes, linker DNA, and linker histone form a unique structural motif that directs the higher-order folding and compaction of chromatin. *Proc. Natl. Acad. Sci. U. S. A.* 95, 14173–14178 (1998).

9. Lee, J. Y. & Lee, T.-H. Effects of DNA Methylation on the Structure of Nucleosomes. *J. Am. Chem. Soc.* 134, 173–175 (2012).

10. PENNINGS, S., MUYLDERMANS, S., MEERSSEMAN, G. & WYNS, L. FORMATION, STABILITY AND CORE HISTONE POSITIONING OF NUCLEOSOMES REASSEMBLED ON BENT AND OTHER NUCLEOSOME-DERIVED DNA. *J. Mol. Biol.* 207, 183–192 (1989).

11. Collings, C. K., Waddell, P. J. & Anderson, J. N. Effects of DNA methylation on nucleosome stability. *Nucleic Acids Res.* 41, 2918–2931 (2013).

12. Gasch, A. P. *et al.* Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell* 11, 4241–4257 (2000).

13. Weiner, A., Hughes, A., Yassour, M., Rando, O. J. & Friedman, N. High-resolution nucleosome mapping reveals transcription-dependent promoter packaging. *GENOME Res.* 20, 90–100 (2010).

14. Mavrich, T. N. *et al.* A barrier nucleosome model for statistical positioning of nucleosomes throughout the yeast genome. *GENOME Res.* 18, 1073–1083 (2008).

15. O'Neill, J. S. & Reddy, A. B. Circadian clocks in human red blood cells. *Nature* 469, 498-U70 (2011).

16. van Rooij, E. *et al.* A Family of microRNAs Encoded by Myosin Genes Governs Myosin Expression and Muscle Performance. *Dev. Cell* 17, 662–673 (2009).

17. Kundaje, A. *et al.* Ubiquitous heterogeneity and asymmetry of the chromatin environment at regulatory elements. *GENOME Res.* 22, 1735–1747 (2012).

18. Bianconi, E. *et al.* An estimation of the number of cells in the human body. *Ann. Hum. Biol.* 40, 463–471 (2013).

19. SULSTON, J. E. & HORVITZ, H. R. POST-EMBRYONIC CELL LINEAGES OF NEMATODE, CAENORHABDITIS-ELEGANS. *Dev. Biol.* 56, 110–156 (1977).

20. Hannon, G. J. RNA interference. *Nature* 418, 244–251 (2002).

21. Robertson, G. *et al.* Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nat. Methods* 4, 651–657 (2007).

22. Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-wide mapping of in vivo protein-DNA interactions. *Science (80-.).* 316, 1497–1502 (2007).

23. Segal, E. *et al.* A genomic code for nucleosome positioning. *Nature* 442, 772–778 (2006).

24. Locke, G., Haberman, D., Johnson, S. M. & Morozov, A. V. Global remodeling of nucleosome positions in C. elegans. *BMC Genomics* 14, (2013).

25. Johnson, S. M., Tan, F. J., McCullough, H. L., Riordan, D. P. & Fire, A. Z. Flexibility and constraint in the nucleosome core landscape of Caenorhabditis elegans chromatin. *GENOME Res.* 16, 1505–1516 (2006).

26. Ioshikhes, I., Bolshoy, A., Derenshteyn, K., Borodovsky, M. & Trifonov, E. N. Nucleosome DNA sequence pattern revealed by multiple alignment of experimentally mapped sequences. *J. Mol. Biol.* 262, 129–139 (1996).

27. Valouev, A. *et al.* A high-resolution, nucleosome position map of C. elegans reveals a lack of universal sequence-dictated positioning. *GENOME Res.* 18, 1051–1063 (2008).

28. Widom, J. Role of DNA sequence in nucleosome stability and dynamics. *Q. Rev. Biophys.* 34, 269–324 (2001).

29. Field, Y. *et al.* Distinct Modes of Regulation by Chromatin Encoded through Nucleosome Positioning Signals. *PLOS Comput. Biol.* 4, (2008).

30. Valouev, A. *et al.* Determinants of nucleosome organization in primary human cells. *Nature* 474, 516-U148 (2011).

31. Dyer, P. N. *et al.* in *CHROMATIN AND CHROMATIN REMODELING ENZYMES, PT A* 375, 23–44 (ACADEMIC PRESS INC, 2004).

32. Luger, K., Rechsteiner, T. J. & Richmond, T. J. in *CHROMATIN* 304, 3–19 (ACADEMIC PRESS INC, 1999).

33. Sakaue, T., Yoshikawa, K., Yoshimura, S. H. & Takeyasu, K. Histone core slips along DNA and prefers positioning at the chain end. *Phys. Rev. Lett.* 87, (2001).

34. Flaus, A. & Richmond, T. J. Positioning and stability of nucleosomes on MMTV 3 ` LTR sequences. *J. Mol. Biol.* 275, 427–441 (1998).

35. Johnson, S. M. Painting a Perspective on the Landscape of Nucleosome Positioning. *J. Biomol. Struct. Dyn.* 27, 795–802 (2010).

36. Grokhovsky, S. L. *et al.* Sequence-Specific Ultrasonic Cleavage of DNA. *Biophys. J.* 100, 117–125 (2011).

37. Schwartz, S. L. & Farman, M. L. Systematic overrepresentation of DNA termini and underrepresentation of subterminal regions among sequencing templates prepared from hydrodynamically sheared linear DNA molecules. *BMC Genomics* 11, (2010).

38. Poptsova, M. S. *et al.* Non-random DNA fragmentation in next-generation sequencing. *Sci. Rep.* 4, (2014).

39. Tremethick, D. J. Higher-order structures of chromatin: The elusive 30 nm fiber. *Cell* 128, 651–654 (2007).

40. Thastrom, A. *et al.* Sequence motifs and free energies of selected natural and non-natural nucleosome positioning DNA sequences. *J. Mol. Biol.* 288, 213–229 (1999).

41. Praitis, V., Casey, E., Collar, D. & Austin, J. Creation of low-copy integrated transgenic lines in Caenorhabditis elegans. *Genetics* 157, 1217–1226 (2001).

42. Maeshima, K. *et al.* Nucleosomal arrays self-assemble into supramolecular globular structures lacking 30-nm fibers. *EMBO J.* 35, 1115–1132 (2016).

43. Li, B., Carey, M. & Workman, J. L. The role of chromatin during transcription. *Cell* 128, 707–719 (2007).

44. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* 9, 357-U54 (2012).

45. Kouzarides, T. Chromatin modifications and their function. *Cell* 128, 693–705 (2007).

46. Kempton, C. E., Heninger, J. R. & Johnson, S. M. Reproducibility and Consistency of In Vitro Nucleosome Reconstitutions Demonstrated by Invitrosome Isolation and Sequencing. *PLoS One* 9, (2014).

47. Jones, P. A. & Baylin, S. B. The epigenomics of cancer. *Cell* 128, 683–692 (2007).

48. Gracey, L. E. *et al.* An in vitro-identified high-affinity nucleosome-positioning signal is capable of transiently positioning a nucleosome in vivo. *Epigenetics Chromatin* 3, (2010).

49. Felsenfeld, G. & Groudine, M. Controlling the double helix. *Nature* 421, 448–453 (2003).

50. Fay, D. Genetic mapping and manipulation: Chapter 1-Introduction and basics. *WormBook* 17, 1–12 (2006).

Appendix A. Supplemental Tables and Figures

Table A 1 Rsa I-Invitrosome reads mapped, passed, suspect and recovered

Range	Total #	Passed	%	Suspect	%	InnerCut	%	Rec	%	% Rec + InnerCut	Biased	%	Usable Data	Rec/ suspect
1	8617075	7880011	91.4%	737064	8.6%	514960	5.98%	150947	1.8%	7.7%	71157	0.8%	99.2%	90.3%
5	8617075	7661723	88.9%	955352	11.1%	514960	5.98%	299089	3.5%	9.4%	141303	1.6%	98.4%	85.2%
11	8617075	7382011	85.7%	1235064	14.3%	514960	5.98%	484998	5.6%	11.6%	235106	2.7%	97.3%	81.0%
22	8617075	7000636	81.2%	1616439	18.8%	514960	5.98%	731608	8.5%	14.5%	369871	4.3%	95.7%	77.1%
33	8617075	6681192	77.5%	1935883	22.5%	514960	5.98%	933195	10.8%	16.8%	487728	5.7%	94.3%	74.8%
44	8617075	6389867	74.2%	2227208	25.8%	514960	5.98%	1111242	12.9%	18.9%	601006	7.0%	93.0%	73.0%
55	8617075	6119893	71.0%	2497182	29.0%	514960	5.98%	1271443	14.8%	20.7%	710779	8.2%	91.8%	71.5%
66	8617075	5865791	68.1%	2751284	31.9%	514960	5.98%	1418355	16.5%	22.4%	817969	9.5%	90.5%	70.3%
77	8617075	5628494	65.3%	2988581	34.7%	514960	5.98%	1551328	18.0%	24.0%	922293	10.7%	89.3%	69.1%
88	8617075	5400078	62.7%	3216997	37.3%	514960	5.98%	1675494	19.4%	25.4%	1026543	11.9%	88.1%	68.1%
99	8617075	5182750	60.1%	3434325	39.9%	514960	5.98%	1790331	20.8%	26.8%	1129034	13.1%	86.9%	67.1%
110	8617075	4972656	57.7%	3644419	42.3%	514960	5.98%	1899669	22.0%	28.0%	1229790	14.3%	85.7%	66.3%
121	8617075	4777977	55.4%	3839098	44.6%	514960	5.98%	1992602	23.1%	29.1%	1331536	15.5%	84.5%	65.3%
132	8617075	4591099	53.3%	4025976	46.7%	514960	5.98%	2081004	24.1%	30.1%	1430012	16.6%	83.4%	64.5%
143	8617075	4413970	51.2%	4203105	48.8%	514960	5.98%	2162445	25.1%	31.1%	1525700	17.7%	82.3%	63.7%
154	8617075	4243382	49.2%	4373693	50.8%	514960	5.98%	2238928	26.0%	32.0%	1619805	18.8%	81.2%	63.0%
165	8617075	4078640	47.3%	4538435	52.7%	514960	5.98%	2309095	26.8%	32.8%	1714380	19.9%	80.1%	62.2%
176	8617075	3921969	45.5%	4695106	54.5%	514960	5.98%	2371337	27.5%	33.5%	1808809	21.0%	79.0%	61.5%
187	8617075	3770628	43.8%	4846447	56.2%	514960	5.98%	2430618	28.2%	34.2%	1900869	22.1%	77.9%	60.8%
200	8617075	3601597	41.8%	5015478	58.2%	514960	5.98%	2492674	28.9%	34.9%	2007844	23.3%	76.7%	60.0%

Table A 2 Hinc II-Invitrosome reads mapped, passed, suspect and recovered

Range	Total #	Passed	%	Suspect	%	InnerCut	%	Rec	%	% Rec + InnerCut	Biased	%	Usable Data	Rec/ suspect
1	4848298	4777873	98.5%	70425	1.5%	39166	0.81%	24779	0.5%	1.3%	6480	0.1%	99.9%	90.8%
5	4848298	4754403	98.1%	93895	1.9%	39166	0.81%	42988	0.9%	1.7%	11741	0.2%	99.8%	87.5%
11	4848298	4724222	97.4%	124076	2.6%	39166	0.81%	66211	1.4%	2.2%	18699	0.4%	99.6%	84.9%
22	4848298	4670935	96.3%	177363	3.7%	39166	0.81%	105115	2.2%	3.0%	33082	0.7%	99.3%	81.3%
33	4848298	4619141	95.3%	229157	4.7%	39166	0.81%	140071	2.9%	3.7%	49920	1.0%	99.0%	78.2%
44	4848298	4568470	94.2%	279828	5.8%	39166	0.81%	171767	3.5%	4.4%	68895	1.4%	98.6%	75.4%
55	4848298	4521215	93.3%	327083	6.7%	39166	0.81%	197859	4.1%	4.9%	90058	1.9%	98.1%	72.5%
66	4848298	4474781	92.3%	373517	7.7%	39166	0.81%	222031	4.6%	5.4%	112320	2.3%	97.7%	69.9%
77	4848298	4428752	91.3%	419546	8.7%	39166	0.81%	244167	5.0%	5.8%	136213	2.8%	97.2%	67.5%
88	4848298	4383876	90.4%	464422	9.6%	39166	0.81%	265002	5.5%	6.3%	160254	3.3%	96.7%	65.5%
99	4848298	4339733	89.5%	508565	10.5%	39166	0.81%	282679	5.8%	6.6%	186720	3.9%	96.1%	63.3%
110	4848298	4296274	88.6%	552024	11.4%	39166	0.81%	297918	6.1%	7.0%	214940	4.4%	95.6%	61.1%
121	4848298	4252225	87.7%	596073	12.3%	39166	0.81%	313097	6.5%	7.3%	243810	5.0%	95.0%	59.1%
132	4848298	4208768	86.8%	639530	13.2%	39166	0.81%	327116	6.7%	7.6%	273248	5.6%	94.4%	57.3%
143	4848298	4165363	85.9%	682935	14.1%	39166	0.81%	341094	7.0%	7.8%	302675	6.2%	93.8%	55.7%
154	4848298	4121924	85.0%	726374	15.0%	39166	0.81%	352796	7.3%	8.1%	334412	6.9%	93.1%	54.0%
165	4848298	4078775	84.1%	769523	15.9%	39166	0.81%	363022	7.5%	8.3%	367335	7.6%	92.4%	52.3%
176	4848298	4035732	83.2%	812566	16.8%	39166	0.81%	371340	7.7%	8.5%	402060	8.3%	91.7%	50.5%
187	4848298	3995010	82.4%	853288	17.6%	39166	0.81%	378522	7.8%	8.6%	435600	9.0%	91.0%	49.0%
200	4848298	3946037	81.4%	902261	18.6%	39166	0.81%	386545	8.0%	8.8%	476550	9.8%	90.2%	47.2%

For Tables A.1 and A.2: Each percent is relative to the number of reads mapped except for Rec/suspect

Usable Data: passed + recovered + InnerCut

Table A 3 Effect of imperfect versus exact match on invitrosome recovery

Range	*Rsa I* Δ 11-bp (-) perfect	*Hinc II* Δ 11-bp (-) perfect
1	9.9%	8.9%
5	14.8%	13.1%
11	18.3%	15.7%
22	20.7%	18.1%
33	21.8%	18.9%
44	22.4%	19.2%
55	22.7%	19.2%
66	22.9%	19.0%
77	23.0%	18.7%
88	23.0%	18.6%
99	23.0%	18.6%
110	22.9%	18.1%
121	22.8%	17.8%
132	22.6%	17.4%
143	22.5%	17.1%
154	22.4%	16.8%
165	22.2%	16.5%
176	22.0%	16.0%
187	21.9%	15.7%
200	21.7%	15.4%

Figure A.1 Code for java program “Application.java” written by Jordon Ritchie and Collin Skousen

Application.java:

```
package com.rescue;

import com.rescue.dao.SaveBowtieReads;
import com.rescue.dto.BowtieOutput;
import com.rescue.dto.SavedReads;
import com.rescue.dto.SeparatedCutSites;
import org.apache.commons.cli.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.*;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
/**
 * Created by Jordon on 5/28/2016.
 */
public class Application {
   private static long start;
   private static long end;
   private static final Logger LOG = LoggerFactory.getLogger(Application.class);
   private static final Map<String, String> CHROM;
   static
   {
      CHROM = new HashMap<>();
      CHROM.put("CHROMOSOME_I", "chr1");
      CHROM.put("CHROMOSOME_II", "chr2");
      CHROM.put("CHROMOSOME_III", "chr3");
      CHROM.put("CHROMOSOME_IV", "chr4");
      CHROM.put("CHROMOSOME_V","chr5");
      CHROM.put("CHROMOSOME_X", "chrX");
      CHROM.put("CHROMOSOME_MtDNA", "chrM");
   }
   private static Options options;
   private static File hincBowtieProcessedReads;
   private static File rsaBowtieProcessedReads;
   private static Integer endOfReadsBuffer;
   private static Integer mismatchesAllowed;
   private static File hincCutSitesFh;
   private static File rsaCutSitesFh;
```

```
public static Integer marginOfError;

private static List<String> hincBowtieFh;
private static List<String> rsaBowtieFh;
private static BowtieOutput hincBowtieOutput;
private static BowtieOutput rsaBowtieOutput;

private static Map<String, Set<Integer>> hincCutSites;
private static Map<String, Set<Integer>> rsaCutSites;

private static SeparatedCutSites hinc;
private static SeparatedCutSites rsa;

private static Map<String, List<Integer>> hincSaved;
private static Map<String, List<Integer>> rsaSaved;

public static void rescue(){
   start = System.nanoTime();
   hincBowtieOutput = Application.processBowtie(hincBowtieFh);
   rsaBowtieOutput = Application.processBowtie(rsaBowtieFh);

   try {
      hincCutSites = Application.extractCutSites(hincCutSitesFh);
      rsaCutSites = Application.extractCutSites(rsaCutSitesFh);
   } catch (IOException e) {
      LOG.error("Failed to parse cut site files");
      e.printStackTrace();
   }
   hinc = Application.separateReads(hincBowtieOutput, hincCutSites);
   rsa = Application.separateReads(rsaBowtieOutput, rsaCutSites);

   hincSaved = Application.saveReads(hinc, rsa);
   rsaSaved = Application.saveReads(rsa, hinc);
   end = System.nanoTime();
   LOG.info("Time (seconds): " + (end - start) / 1.0e9);
}

private static BowtieOutput processBowtie(List<String> fh) {
   Map<String, List<Integer>> alignedReadsChrPos = new HashMap<>();
   Map<Integer, String> alignedReads = new HashMap<>();
   Set<String> failedAlignmentCounts = new HashSet<>();

   for (Map.Entry<String, String> c : CHROM.entrySet()){
      alignedReadsChrPos.put(c.getValue(), new ArrayList<>());
```

```
    }

    for(String r : fh) {
      String[] output = r.split("\\t");
      if(output[0].equals("")) {
        failedAlignmentCounts.add(r);
      } else {
        if (output[0].equals("-")) {
          Integer dyadPosition = Integer.parseInt(output[2]) - 37;
          alignedReadsChrPos.get(CHROM.get(output[1])).add(dyadPosition); //this is a
reverse read
          alignedReads.put(dyadPosition, output[3]); //right now we are using the bowtie
read output.
        }
        else if (output[0].equals("+")) {
          Integer dyadPosition = Integer.parseInt(output[2]) + 74;
          alignedReadsChrPos.get(CHROM.get(output[1])).add(dyadPosition); //this is a
forward read
          alignedReads.put(dyadPosition, output[3]); //right now we are using the bowtie
read output.
        }else{
          LOG.info("Mystery entry: " + output.length + " " + output[0]);
        }
      }
    }

    LOG.info("Process Bowtie size: " + Application.validate(alignedReadsChrPos));

    return new BowtieOutput(failedAlignmentCounts, alignedReads, alignedReadsChrPos);
  }

  public static Map<String, Set<Integer>> extractCutSites(File fh) throws IOException {
    LOG.debug("EXTRACT CUTS SITES");
    String line;
    String chr = "";
    Map<String, Set<Integer>> cutSites = new HashMap<>();
    for (Map.Entry<String, String> c : CHROM.entrySet()){
      cutSites.put(c.getValue(), new HashSet<>());
    }
    BufferedReader br = new BufferedReader(new FileReader(fh));
    while((line = br.readLine()) != null){
      if (line.startsWith(">")){
        chr = line.substring(1);
        LOG.debug("CHROM: " + chr);
```

```
        }else{
            String[] sites = line.split("\\s+");
            cutSites.get(chr).add(Integer.parseInt(sites[1]));
            LOG.debug("SITE: " + sites[1]);
        }
    }
    return cutSites;
  }

  public static SeparatedCutSites separateReads(BowtieOutput bowtieOutput, Map<String,
Set<Integer>> cutSites) {
    Map<String, List<Integer>> with = new HashMap<>();
    Map<String, List<Integer>> withOut = new HashMap<>();
    Map<String, List<Integer>> innerCutSiteRange = new HashMap<>();

    for (Map.Entry<String, String> c : CHROM.entrySet()){
        innerCutSiteRange.put(c.getValue(), new ArrayList<>());
        with.put(c.getValue(), new ArrayList<>());
        withOut.put(c.getValue(), new ArrayList<>());
    }

    for (Map.Entry<String, List<Integer>> read :
bowtieOutput.getAlignedReadsChrPos().entrySet()){
        for(Integer j : read.getValue()) {
            Set<Integer> range1Cuts = IntStream.rangeClosed(j + 74, j + 74 + endOfReadsBuffer)
                    .boxed().collect(Collectors.toSet()); //gets positive cuts
            Set<Integer> negativeCuts = IntStream.rangeClosed(j - 73 - endOfReadsBuffer, j - 73)
                    .boxed().collect(Collectors.toSet());
            range1Cuts.addAll(negativeCuts);

            Set<Integer> range2Cuts = IntStream.rangeClosed(j - 72, j + 73)
                    .boxed().collect(Collectors.toSet());


            Boolean found1 = Boolean.FALSE;
            Boolean found2 = Boolean.FALSE;
            for (Integer i : range1Cuts) {
                if (cutSites.get(read.getKey()).contains(i)) {
                    found1 = Boolean.TRUE;
                }
            }
            for (Integer k : range2Cuts) {
                if (cutSites.get(read.getKey()).contains(k)) {
                    found2 = Boolean.TRUE;
```

```
            }
          }

          if (found1 == Boolean.TRUE && found2 == Boolean.FALSE) {
            with.get(read.getKey()).add(j);
          } else if (found2 == Boolean.TRUE){
            innerCutSiteRange.get(read.getKey()).add(j);
          }
          else {
            withOut.get(read.getKey()).add(j);
          }
        }
      }
      SeparatedCutSites separatedCutSites = new SeparatedCutSites(with, withOut,
innerCutSiteRange);
      Application.removeSavedReads(separatedCutSites, innerCutSiteRange);
      LOG.info("Separate Reads size: " + (Application.validate(with) +
Application.validate(withOut) + Application.validate(innerCutSiteRange)));
      return separatedCutSites;
    }

    public static Map<String, List<Integer>> saveReads(SeparatedCutSites withSites,
SeparatedCutSites withOutSites){

      //original
//      SavedReads traditionalSave = save(withSites.getWith(), withOutSites.getWithOut());
//      Application.removeSavedReads(withSites, traditionalSave.getToRemove());
//      SavedReads innerSave = save(withSites.getWith(), withOutSites.getInnerCutSiteRange());
//      Application.removeSavedReads(withSites, innerSave.getToRemove());

      //multithreaded
      SaveBowtieReads saveBowtieReadsTraditional = new SaveBowtieReads();
      SavedReads traditionalSave = saveBowtieReadsTraditional.saveReads(withSites.getWith(),
withOutSites.getWithOut());
      Application.removeSavedReads(withSites, traditionalSave.getToRemove());

      SaveBowtieReads saveBowtieReadsInner = new SaveBowtieReads();
      SavedReads innerSave = saveBowtieReadsInner.saveReads(withSites.getWith(),
withOutSites.getInnerCutSiteRange());
      Application.removeSavedReads(withSites, innerSave.getToRemove());


      for(Map.Entry<String, List<Integer>> entry : traditionalSave.getSaved().entrySet()){
```

```
traditionalSave.getSaved().get(entry.getKey()).addAll(innerSave.getSaved().get(entry.getKey()));
      }


      LOG.info("Saved Reads size: " + (Application.validate(withSites.getWith()) +
            Application.validate(withSites.getWithOut()) +
            Application.validate(traditionalSave.getSaved()) +
            Application.validate(withSites.getInnerCutSiteRange())));
      return traditionalSave.getSaved();
   }

//    public static SavedReads save(Map<String, List<Integer>> withSites, Map<String,
List<Integer>> withOutSites){
//       Map<String, List<Integer>> saved = new HashMap<>();
//       Map<String, List<Integer>> toRemove = new HashMap<>();
//
//       for (Map.Entry<String, String> c : CHROM.entrySet()){
//          saved.put(c.getValue(), new ArrayList<>());
//          toRemove.put(c.getValue(), new ArrayList<>());
//       }
//
//       for (Map.Entry<String, List<Integer>> withSite : withSites.entrySet()) {
//          for (Integer dyad : withSite.getValue()) {
//             Set<Integer> allPossible = IntStream.rangeClosed(dyad - marginOfError, dyad +
marginOfError)
//                   .boxed().collect(Collectors.toSet());
//
//             for (Integer i : allPossible){
//                if(withOutSites.get(withSite.getKey()).contains(i)){
//                   saved.get(withSite.getKey()).add(dyad);
//                   toRemove.get(withSite.getKey()).add(dyad);
//                   break;
//                }
//             }
//          }
//       }
//       return new SavedReads(saved, toRemove);
//    }

   public static void main(String[] args) {

      options = new Options();
      options.addOption("hr", true, "path to hinc reads file");
```

```
        options.addOption("rr", true, "path to rsa reads file");
        options.addOption("e",  true, "length of extra bases on end of Reads");
//        options.addOption("mm", true, "number of mismatches allowed");
        options.addOption("hc", true, "cut sites for hincII");
        options.addOption("rc", true, "cut sites for rsaI");
        options.addOption("ma", true, "margin of error for with and without cut sites
comparison");


        CommandLineParser parser = new DefaultParser();
        try {
            CommandLine cmd = parser.parse(options, args);

            if(cmd.hasOption("h")){
                printHelp();
            }

            hincBowtieProcessedReads = new File(cmd.getOptionValue("hr"));
            rsaBowtieProcessedReads = new File(cmd.getOptionValue("rr"));
            endOfReadsBuffer = Integer.parseInt(cmd.getOptionValue("e"));
            hincCutSitesFh = new File(cmd.getOptionValue("hc"));
            rsaCutSitesFh = new File(cmd.getOptionValue("rc"));
            marginOfError = Integer.parseInt(cmd.getOptionValue("ma"));

            String hincLine;
            hincBowtieFh = new ArrayList<>();
            BufferedReader hincBr = new BufferedReader(new
FileReader(hincBowtieProcessedReads));
            while((hincLine = hincBr.readLine()) != null){
                hincBowtieFh.add(hincLine);
            }
            LOG.info("hinc :" + hincBowtieFh.size());
            String rsaLine;
            rsaBowtieFh = new ArrayList<>();
            BufferedReader rsaBr = new BufferedReader(new
FileReader(rsaBowtieProcessedReads));
            while((rsaLine = rsaBr.readLine()) != null){
                rsaBowtieFh.add(rsaLine);
            }
            LOG.info("rsa :" + rsaBowtieFh.size());

            Application.rescue();

            Map<String, Map<String, List<Integer>>> dataToWrite = new HashMap<>();
```

```
dataToWrite.put("readsWithHincCutSites", hinc.getWith());
dataToWrite.put("readsWithRsaCutSites", rsa.getWith());
dataToWrite.put("readsWithoutHincCutSites", hinc.getWithOut());
dataToWrite.put("readsWithoutRsaCutSites", rsa.getWithOut());
dataToWrite.put("savedHincReads", hincSaved);
dataToWrite.put("savedRsaReads", rsaSaved);
dataToWrite.put("innerCutSiteRangeHinc", hinc.getInnerCutSiteRange());
dataToWrite.put("innerCutSiteRangeRsa", rsa.getInnerCutSiteRange());
Application.writeFiles(dataToWrite);

Integer total = 0;
System.out.println();
LOG.info("readsWithHincCutSites ");
for(Map.Entry<String, List<Integer>> entry : hinc.getWith().entrySet()){
   LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
   total += entry.getValue().size();
}
LOG.info("\ttotal " + total);

System.out.println();
LOG.info("readsWithRsaCutSites ");
total = 0;
for(Map.Entry<String, List<Integer>> entry : rsa.getWith().entrySet()){
   LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
   total += entry.getValue().size();
}
LOG.info("\ttotal " + total);

System.out.println();
LOG.info("readsWithoutHincCutSites ");
total = 0;
for(Map.Entry<String, List<Integer>> entry : hinc.getWithOut().entrySet()){
   LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
   total += entry.getValue().size();
}
LOG.info("\ttotal " + total);

System.out.println();
LOG.info("readsWithoutRsaCutSites ");
total = 0;
for(Map.Entry<String, List<Integer>> entry : rsa.getWithOut().entrySet()){
   LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
   total += entry.getValue().size();
}
```

```
    LOG.info("\ttotal " + total);

    System.out.println();
    LOG.info("savedHincReads ");
    total = 0;
    for(Map.Entry<String, List<Integer>> entry : hincSaved.entrySet()){
      LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
      total += entry.getValue().size();
    }
    LOG.info("\ttotal " + total);

    System.out.println();
    LOG.info("savedRsaReads ");
    total = 0;
    for(Map.Entry<String, List<Integer>> entry : rsaSaved.entrySet()){
      LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
      total += entry.getValue().size();
    }
    LOG.info("\ttotal " + total);

    System.out.println();
    LOG.info("innerCutSiteRangeHinc ");
    total = 0;
    for(Map.Entry<String, List<Integer>> entry : hinc.getInnerCutSiteRange().entrySet()){
      LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
      total += entry.getValue().size();
    }
    LOG.info("\ttotal " + total);

    System.out.println();
    LOG.info("innerCutSiteRangeRsa ");
    total = 0;
    for(Map.Entry<String, List<Integer>> entry : rsa.getInnerCutSiteRange().entrySet()){
      LOG.info("\t" + entry.getKey() + " " + entry.getValue().size());
      total += entry.getValue().size();
    }
    LOG.info("\ttotal " + total);


  } catch (ParseException e) {
    LOG.error("Failed to parse command line arguments: " + e.getMessage());
    printHelp();
  } catch (Exception e) {
    e.printStackTrace();
```

```
        }
    }

    private static void writeFiles(Map<String, Map<String, List<Integer>>> dataToWrite) {
        for (Map.Entry<String, Map<String, List<Integer>>> dataSet : dataToWrite.entrySet()){
//            File dir = new File("..\\data\\output\\" + endOfReadsBuffer);
//            if(!dir.exists()){
//                Boolean mkdir = dir.mkdir();
//            }
            File file = new File(endOfReadsBuffer + dataSet.getKey() + ".txt");
            try {
                BufferedWriter bw = new BufferedWriter(new FileWriter(file));
                for (Map.Entry<String, List<Integer>> data : dataSet.getValue().entrySet()){
                    for(Integer i : data.getValue()) {
                        bw.write(data.getKey() + "\t" + i + "\n");
                    }
                }
                bw.close();
            } catch (IOException e) {
                LOG.error("Could not create output file for " + dataSet.getKey());
                e.printStackTrace();
            }
        }
    }

    private static void printHelp() {
        HelpFormatter formatter = new HelpFormatter();
        formatter.printHelp("Main", options);
        System.exit(0);
    }

    private static Integer validate(Map<String, List<Integer>> data){
        Integer size = 0;
        for (Map.Entry<String, List<Integer>> entry : data.entrySet()){
            size += entry.getValue().size();
        }
        return size;
    }

    private static void removeSavedReads(SeparatedCutSites removeFrom, Map<String,
List<Integer>> toRemove){
        for (Map.Entry<String, List<Integer>> r : toRemove.entrySet()){
//            LOG.info(r.getKey() + "-before: " + removeFrom.getWith().get(r.getKey()).size());
            List<Integer> remove = new ArrayList<>();
```

```
            for (Integer i : r.getValue()) {
                remove.add(i);
            }
            removeFrom.getWith().get(r.getKey()).removeAll(remove);
//            LOG.info(r.getKey() + "-after: " + removeFrom.getWith().get(r.getKey()).size());
        }
    }
}
```