IDENTIFICATION OF FUNCTIONAL DOMAINS IN NON-CODING RNA

Daniel Sprague

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Pharmacology in the School of Medicine.

Chapel Hill
2020

Approved by:

Joseph Mauro Calabrese

Lee Graves

Daniel Dominguez

Jeremy Purvis

Daniel Schrider

ABSTRACT

Daniel Sprague: Identification of functional domains in non-coding RNA
(Under the direction of Joseph Mauro Calabrese)

Long non-coding RNAs (lncRNAs) are known to be important regulators of gene expression and other cellular functions. However, only a very small proportion of lncRNAs have been extensively studied. The remainder exist largely as annotations in a database with no known function, if any. A primary challenge to understanding how lncRNAs function is the poorly understood sequence-to-function relationship relative to protein coding genes. Within lncRNA transcripts, boundaries of functional sequence are not explicitly defined by exon-intron boundaries, and the code by which lncRNAs derive function is not nearly as explicit as in a protein coding reading frame. To address these challenges, we have developed a probabilistic framework, *hmmSEEKR*, for identifying where within a non-coding RNA functional regions may be located based off of enrichment of short motifs, or k-mers.

We used *hmmSEEKR* to identify functional sequence domains in several lncRNAs that silence gene expression through recruitment of Polycomb, using *XIST* as a model transcript. These predicted sequence domains share no detectable linear sequence alignment with *XIST*; however, they share high k-mer based similarity with known functional domains in *XIST* and precisely coincided with the location of RNA binding protein (RBP) interactions known to be important for Polycomb mediated silencing. Furthermore, we were able to extend our analysis to the entire transcriptome and identify many *XIST*-like sequence domains throughout the transcriptome that interact with Polycomb-associated RBPs. We have packaged these algorithms into python-based software and have included an in-depth walk-through of the code and tutorial of how to analyze sequences using *hmmSEEKR.*

ACKNOWLEDGEMENTS

This work wouldn't have been possible without all the fantastic people who helped me through the end. I would like to thank my advisor, Mauro Calabrese, for always pushing me to improve as well as for the opportunity to work on some very exciting science. I would also like to thank all the friends and family who made this work possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

$\in$ ..... in

$\mapsto$ ..... maps to

$\sum_X$ ..... sum the values over all items in the set X

"+" ..... query state in *hmmSEEKR*

"-" ..... null state in *hmmSEEKR*

bp ..... Basepair

BLAST ..... Basic local alignment search tool

eCLIP ..... Enhanced cross-linking immunoprecipitation

HMM ..... Hidden Markov Model

HMMER ..... Profile HMM tool for sequence alignment

hmmSEEKR .. Hidden Markov Model based Sequence Evaluation through k-mer representation

KL-Divergence Kullback-Leibler Divergence

KR(1-4) ..... Koala *Rsx* Repeats 1,2,3,4

lncRNA ..... Long non-coding RNA

$P(X, Y)$ ..... Joint probability distribution

$P(X|Y)$ ..... Conditional probability distribution

MX(A-E) ..... Mouse *Xist* Repeats A,B,C,E

HX(A-E) ..... Human *XIST* Repeats A,B,D,E

mRNA ..... Messenger RNA

OR(1-4) ..... Opossum *Rsx* Repeats 1,2,3,4

PRC ..... Polycomb Repressive Complex

PWM ..... Position weight matrix

RBP ..... RNA Binding Protein

RNA-IP ..... RNA immunoprecipitation

RNA-seq ..... RNA sequencing

SAF ..... Simple annotation format

SEEKR ..... Sequence evaluation through k-mer representation

XCI ..... X chromosome inactivation

Xi ..... Inactive X

XIST ..... Human X inactivation specific transcript

Xist ..... Mouse X inactivation specific transcript

CHAPTER 1: INTRODUCTION

1 Long non-coding RNAs

Protein coding genes have dominated the study of genetics for decades. The rules by which they function are well understood and we can often predict the function of an unknown gene based off of its sequence alone [1]. This is due to the well understood rules of protein coding reading frames and the principles of conservation in evolutionay biology [2–4]. The paradigm of the central dogma, DNA →RNA→Protein, was upended, however, with the discovery of the human non-coding *XIST* RNA and its essential function in the process of X chromosome inactivation (XCI) in all eutherians [5, 6]. This discovery led to RNA being thought of as not just a messenger, but also as a functional end product in the cell [5–11].

The advent of RNA sequencing has led to an explosion of annotated RNA transcripts that have no corresponding protein products in multiple mammalian species [12–14]. Even with numerous advances in genetics and genomics, these RNAs have proven particularly challenging to understand. The functions of non-coding RNAs range from gene expression regulation through RNA interference, to chromatin remodeling, and extensive association with human disease [7–11].

Long non-coding RNAs (lncRNAs) are defined as non-coding RNA transcripts longer than 200 base pairs (bp) [12]. lncRNAs have emerged as a major mechanism of gene regulation in eutherians [14–16], however they have been annotated in eukaryotic organisms as far back as *S. cerevisiae* [17]. Despite their ubiquitous presence, lncRNAs have evaded generalized understanding due to their variety and poor conservation across species [18]. That is, while proteins can have many different functions within the cell, those functions are encoded through a specific and predictable code, whereas few discernible patterns and poor conservation in lncRNAs occlude any obvious sequence-to-function relationship.

Of the lncRNAs that have been studied and have known function within the cell, there often is considerable work left to fully understand the mechanism of action of these transcripts, despite many years worth of work. A particular challenge with lncRNAs is that they are often

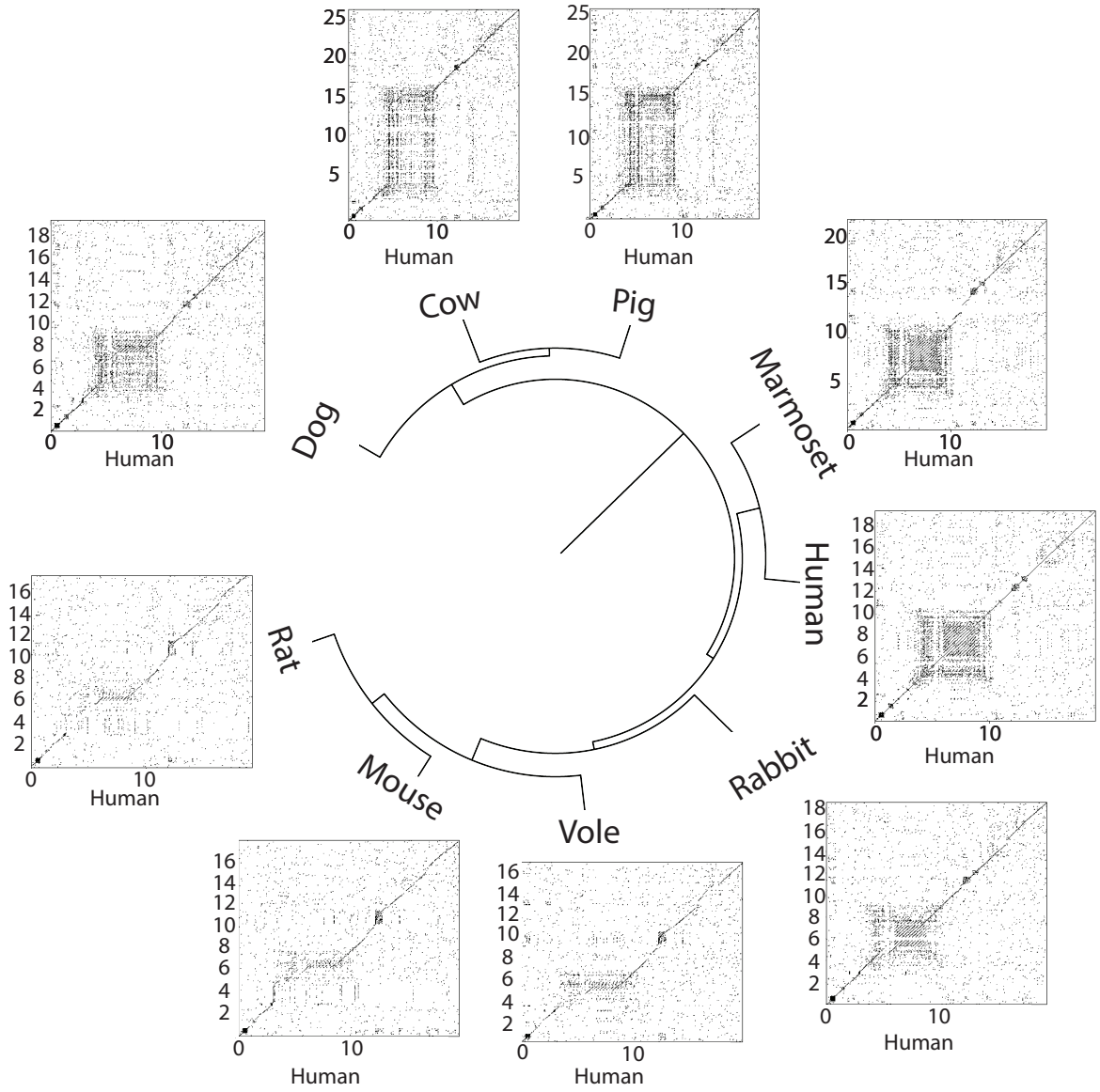

Figure 1.1: Human *Xist* aligned against *Xist* from other mammals using dotmatcher with window size = 20 and threshold = 50. Human *Xist* is along the x-axis and the indicated species is along the y-axis. Repeat D-like regions tend to be the largest domains of similarity between human and non-murid *Xists*.

poorly conserved related to protein coding genes (Figure 1.1), if they are conserved at all [18–20]. Furthermore, conventional alignment tools such as BLAST or nhmmer fail to detect meaningful relationships between two transcripts with similar function [21, 22].

lncRNAs have proven difficult to study experimentally, as well. lncRNAs rarely function in isolation, and are not known to be catalytic, but rather function through the concerted action

of the lncRNA transcript, RNA binding proteins, and other biological molecules such as DNA or other proteins [9, 23–26]. Rather than being a single functional unit in a larger sequence of molecular operations within the cell, such as kinases in a signalling cascade, lncRNAs serve as hubs wherein complex molecular processes take place in tandem with each other [23, 24, 27, 28].

1.1 Functional annotation

Long non-coding RNAs biochemically are very similar to mRNAs, in fact there is very little other than the presence of a reading frame that differentiates a messenger RNA from a lncRNA [29]. lncRNAs are thus genes for which the final functional product is an RNA transcript [30]. Given that RNA molecules are known to chemically bind to numerous biological molecules, including protein, DNA, and RNA itself [23–25], it perhaps comes as little surprise that lncRNAs are capable of widely varied function within the cell.

lncRNAs have generally been grouped into several sub-categories based on broadly defined features of their loci. The primary groups that have been annotated are intergenic lncRNAs (lincRNAs), which are found between annotated protein coding genes, anti-sense lncRNAs, whose loci are located on the anti-sense strand relative to a known protein coding gene, and intronic lncRNAs, which are encapsulated within the intron of a protein coding gene [7]. The vague and non-specific nature of these definitions underlines how little is known about lncRNAs in general. In comparison, sequence information alone it is possible to annotate protein coding genes down to their exact roles within the cell, not-with-standing larger classifications into cliques and communities within larger functional networks [31].

What is relatively well known is that lncRNAs are often involved in genetic regulation [7, 21, 24, 25]. Since the discovery of the lncRNA *XIST* and its role in XCI, and the further identification of additional lncRNAs in years following the introduction of next generation sequencing, it has become clear that a common functional role of lncRNAs is the regulation of gene expression in human and other mammalian genomes. While RNA-seq has allowed for the rapid discovery of genes that produce RNA transcripts as their final product, the elucidation of the function for these newly annotated transcripts has proven far more challenging [7, 21].

Guilt by association

Assigning functional roles to lncRNAs, if there is any function to be assigned, has proven to be a daunting task. One of the first techniques used is the so-called *guilt by association* method [7, 32]. Rather than trying to identify the specific function of a lncRNA, trends from gene expression data are used to attempt to identify features such as specificity of cell type expression [33–35]. A more advanced approach uses informatics techniques to identify protein coding genes and biological pathways that are co-regulated with specific lncRNAs, with the underlying assumption that if a lncRNA is co-regulated with a specific pathway, it is likely to have a functional role within that pathway [36].

This methodology is useful for identifying transcripts that may be useful for further targeted experimental studies, but *guilt by association* is an inherently correlative method that does provide real insights into the lncRNAs of interest. Despite this shortcoming, several functional lncRNAs have been identified and characterized this way [37, 38].

1.2 Polycomb-lncRNA mediated gene regulation

The connection between RNA and chromatin has long been known. Early chromatin purification experiments revealed twice as much RNA as DNA [39]. Further studies have shown that several lncRNAs, such as *Xist*, *Airn*, *Kcnq1ot1*, and others associate with chromatin and modulate the deposition or removal of regulatory markers on chromatin [19, 24, 40].

One feature shared by all these transcripts is the formation of RNA-protein complexes [24], often involving numerous RNA binding proteins at several different regions within the lncRNA transcript [22, 28, 41, 42]. *Xist*, *Kcnq1ot1*, and *Airn* all silence large genomic regions of chromatin through recruitment of Polycomb repressive complexes (PRC), causing deposition of silencing chromatin modification over many megabases of DNA [24]. These lncRNAs act in *cis* – meaning they only act on the their chromosomes of origin, however a few *trans* acting lncRNAs have since been discovered [25, 43]. The specificity of the lncRNA-PRC interaction required for *cis* silencing is intriguing, because PRCs are generally non-specific binders of RNAs [44, 45], however PRCs are recruited to very specific regions of *Xist*'s transcript following *Xist* expression. [42, 46, 47].

The two Polycomb complexes, PRC1 and PRC2, have been shown to compact chromatin and

repress active transcription [24, 48, 49]. PRC1 and PRC2 function in tandem with one another [24, 49], and are recruited to *Xist* in a specific manner through interactions with RBPs [42, 50]. The PRCs then spread across large regions of chromatin through alterations initiated by the expression of these lncRNAs, leading to megabase-scale silencing of chromatin [24].

2 Xist as a model lncRNA

The mouse *Xist* lncRNA is one of the most well characterized lncRNAs due to its essential role in XCI and because it is one of the few transcripts that is completely conserved in all eutherians [5, 6, 21, 22]. XCI is the process by which mammalian females transcriptionally silence a single X chromosome as a means of gene dosage compensation. Proper expression of *Xist* is required for initiation of *XCI*, and *Xist* is required for silencing virtually all genes on the inactive-X [5, 6, 51]. While X-inactivation is complex, involving many additional factors beyond *Xist*, and mechanistic details are still an area of active research, *Xist* provides a well-studied example of lncRNA activity.

Throughout this dissertation, we used human *XIST* as a guide to understand less studied lncRNAs, specifically the known functional analogs *Rsx*, *Airn*, and *KCNQ1OT1*. Despite most studies of *Xist* occurring in a mouse model, the availability of eCLIP experiments for 100+ RBPs makes human *XIST* a more viable case study. The ideas presented in this dissertation can be generalized to essentially any lncRNA; however, given how nascent the lncRNA field is and how little is known – *XIST* represents one of the few well studied lncRNAs in the eutherian genome.

3 k-mer based sequence comparison

Traditional pairwise alignment has been a mainstay of bioinformatics for decades [52]. Originally for detection of homologous sequences, pairwise alignment is now crucial for next generation sequencing experiments [53, 54]. A fundamental issue with pairwise alignment is its complexity. To obtain the best alignment between N sequences of length L, L^N calculations have to be made. To compare just 5 sequences of 100bp each, 10^{10} calculations would have to be made! For this reason, expedients have long been sought. Alignment-free methods of sequence comparison are one such simplification and have existed almost as long as sequence alignment has [55, 56].

A particularly common form of alignment-free analysis involves k-mer frequency comparisons [55]. A k-mer is defined as a sub-sequence of length k within a larger sequence. k-mer based

methods have been used for a variety of purposes, but most commonly for examining speciation in metagenomic samples and the evolutionary distance between two species [57–60]. For DNA, as the value of k is increased, the number of possible permutations of the four nucleotides increases exponentially (4^k). Therefore, the amount of shared overlap of k-mers is a function of their evolutionary distance from each other. Closely related species may show significant overlap for $k \in (14, 26)$, whereas distantly related species may have the same amount of overlap for signficantly smaller values of $k \in (8, 12)$ [57, 59, 60].

These methods are based on the frequencies of k-mers within a larger sequence, not on direct alignment between two sequences. Therefore statistics can quickly be calculated that quantify the degree of similarity between two sets of k-mers. Other methods, such as rapid detection of longest common sub-string [61], have also been developed. However, as described above these methods have been traditionally applied to evolutionary relations, and not as a functional model of sequence content.

Our lab has recently developed a k-mer based method for assessing functionality in long non-coding RNAs [21]. In the following section the motivation and algorithm for using k-mers to predict functionality in long non-coding RNAs is described.

3.1 SEEKR Algorithm

While the sequence relationship between functionally analogous lncRNAs is undetectable with traditional alignment techniques, their sequences are not random [5, 21, 22, 28, 42, 46, 47]. We hypothesized that lncRNAs with shared functions must contain some sequence similarity that confers shared function, even if conventional alignment algorithms do not detect the similarity. Many of the known and well studied lncRNAs function through the recruitment of RNA binding proteins, or RBPs. These proteins are highly conserved and recognize short, 4-6bp motifs on an RNA strand. lncRNAs are further differentiated from protein coding genes in that they do have reading frames – such that we expect that motifs for RNA binding proteins may be located anywhere in a sequence and still recruit that protein. These observations of lncRNA functionality led us to develop a k-mer based bag-of-words model for the sequence-to-function relationship in lncRNAs.

Our lab developed an algorithm for comparing k-mer content between lncRNAs, named

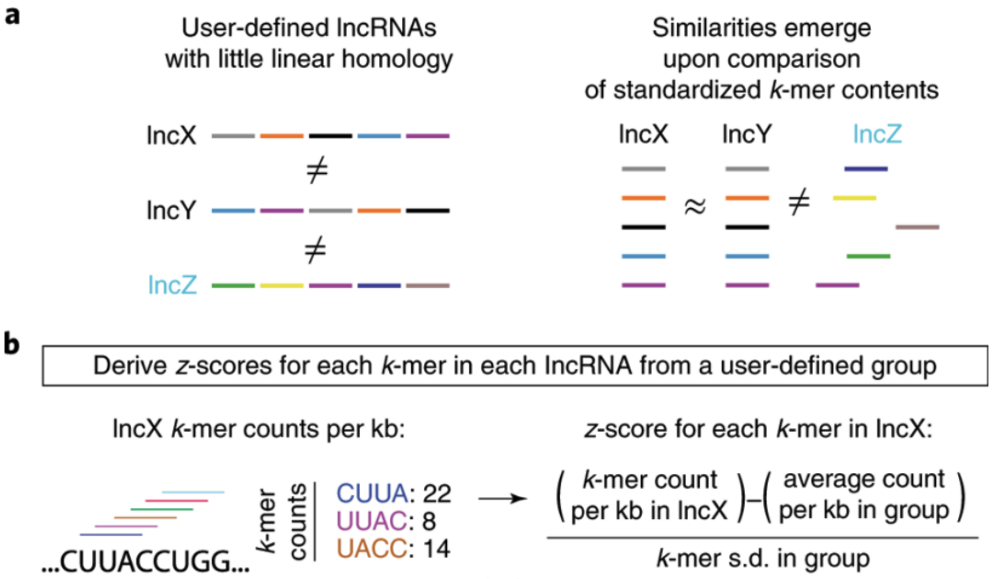

Figure 1.2: (A) LncRNAs of related function (names in black) may harbor similar sequence similarity in the form of motif content (colored bars) even if they lack linear homology. (B) In SEEKR, the abundance of all kmers of length k are counted by tiling across each lncRNA in a user-defined group in one nucleotide increments. [21].

SEEKR (SEquence Evaluation from k-mer Representation). Within the algorithm, each k-mer in a sequence is counted through a 1bp sliding window. k-mer counts for each lncRNA are then normalized by lncRNA length and standardized across a reference set (*e.g.* the transcriptome) to derive a matrix of standardized k-mer frequencies. The degree of similarity between two lncRNA transcripts can then be calculated using the pearson correlation coefficient [21].

4 Hidden Markov Models

A hidden markov model (HMM) is a generative probabilistic model for stochastic sequential processes [2, 62]. For some observable process, or sequence, $X = \{x_1, x_2, \ldots, x_n\}$, we assume there is an underlying, or latent, random process $Y = \{y_1, y_2, \ldots, y_n\}$ that is controlling the behavior of X. A generative probabilistic model such as an HMM calculates the joint probability distribution $P(X, Y)$, rather than the more commonly seen classifier, or discriminative model, that models the conditional probability distribution $P(Y|X)$. Modeling the joint distribution $P(X, Y)$ allows an HMM to not only calculate the probability of any given sequence, but allows for generation of stochastic realizations of the model, *i.e.* an HMM can generate stochastic sequences that behave similarly to the data the model was trained on. HMMs therefore reflect a simplified, hypothesized process by which the data may be reasonably generated. Compara-

tively, most discriminative classifiers simply attempt to map a data point x_i to some class label y_j, $x_i \mapsto y_j$ [63].

To allow for tractable calculation of probabilities, an HMM relies on the Markov assumption, which states that the current observation is conditionally independent of all other observations, except for the observation immediately prior. This assumption is formally defined in Equation 1.1 [62].

$$P(x_t|x_{t-1}, x_{t-2}, \ldots, x_1) = P(x_t|x_{t-1}) \tag{1.1}$$

This assumption appears simplistic, but it is often sufficient for many modeling tasks that would be impossible without simplifying assumptions. HMMs have been used in many fields, from weather prediction to text prediction, such as a common search engine query *e.g.*:

$$P(\texttt{park}|\texttt{dog,nearest,the,where's}) > P(\texttt{office}|\texttt{dog,nearest,the,where's})$$

Could be accurately modeled with the markov assumption:

$$P(\texttt{park}|\texttt{dog}) > P(\texttt{office}|\texttt{dog})$$

A markov based model would be able to reliably suggest the word *park* instead of *office* because the word *office* would almost never follow the word *dog*, regardless of the remainder of the sentence. This assumption has proven to be wildly useful in the field of bioinformatics, especially with probabilistic modeling of DNA sequences. Mammalian chromosomes can be hundrends of millions of basepairs long, but in reality the relationships between nucleotides exist on much shorter scales. Often a nucleotide can be accurately predicted from preceeding sequence using only the previous 5-6 nucleotides [2].

4.1 Gene finding and other applications

HMMs have been used for many purposes, from sequence alignment [4] to modeling the timing of switches in fluorescence from FRET experiments [64]. An early application of HMMs to biological data was gene finding in the 1990s [2]. During this period the Human Genome Project was well underway, and only a fraction of protein coding genes were known. Huge regions of the human genome were being sequenced but massively parallel RNA sequencing technology did not

exist at the time, so it was unknown how many undiscovered genes were in the human genome. Therefore, computational models were developed to predict regions within the genome that contained reading frames[2, 65, 66].

The majority of the methods developed utilized HMMs at their core [2, 65, 66]. The reason for this was relatively straightforward. Protein coding genes are composed of distinct functional domains, and each of these functional domains contains unique nucleotide distributions [2, 65, 66]. Therefore, given a general model of protein coding gene's structure:

$$\text{Promoter} \rightarrow \text{5' UTR} \rightarrow \text{Exon} \rightarrow \text{Intron} \rightarrow \cdots \rightarrow \text{3' UTR}$$

Using the nomenclature defined in 1.4.1, this sequence above would be the sequence of hidden states Y. The actual DNA sequence that we observe, X, is controlled by the presence of these functional domains, and an HMM calculates the joint probability of the parse of the sequence Y and the sequence X, or $P(X, Y)$. A parse is here is defined as taking a string of symbols, *e.g.* nucleotides in a sequence, and identifying which portions of that string correspond to a set of predefined functional states. The idea is that some parses of X will be more likely than others, and that the model can be used to identify the most likely parse, Y^*, of the data.

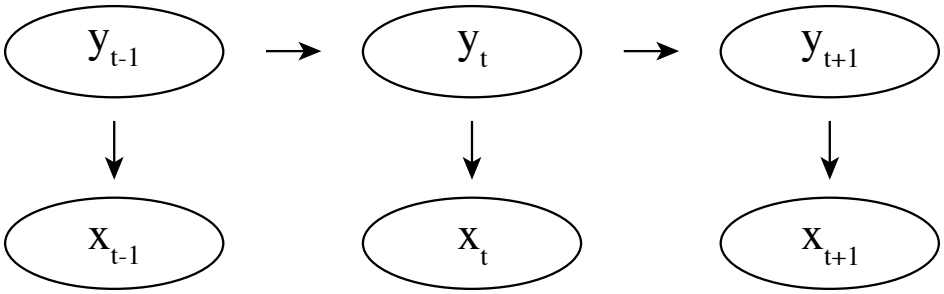

Figure 1.3: Conditional dependencies in a hidden markov model.

4.2 Model structure

The HMM is constructed with the Markov assumption placed on the sequence of hidden states Y, so that for a sequence of length L the probability of the current hidden state y_t is dependent only on the previous hidden state y_{t-1}. Further, y_t does not depend on the observed sequence X.

$$P(y_t, X_{1:t}, Y_{1:t}) = P(y_t|y_{t-1}) \tag{1.2}$$

And the probability of each observation $x_t \in X$ is conditionally independent of everything except the emitting state at time t, y_t, and is not dependent on any other observation.

$$P(x_t, X_{1:t-1}, Y_{1:t}) = P(x_t|y_t) \tag{1.3}$$

These conditional independence assumptions can be represented in the graphical form illustrated in Figure 1.3.

From here the joint probability for an HMM can be derived [62]. This explicitly allows the calculation of the probability for all possible sequences of observations and all possible sequences of hidden states.

$$P(X, Y) = P(y_1)p(x_1|y_1)\prod_{t=2}^{L} P(y_t|y_{t-1})P(x_t|y_t) \tag{1.4}$$

This joint probability distribution allows us to work with the three most common tasks that HMMs are used for: evaluation, decoding, and learning.

4.3 Inference and Algorithms

There are three primary questions that an HMM is used to answer.

1. Evaluation: What is the probability of being in a given state, at a particular point in time? A related quantity is the total probability of the observed sequence, *i.e.* $P(X)$. This value is crucial to convert the joint distribution to conditional distributions, especially with respect to Baye's Theorem and the calculation of maximum likelihood estimates for parameters.

2. Decoding: What is the most likely parse of the sequence?

3. Learning: What are the most likely set of parameters, given the observed data?

Combinatorial Explosion

To answer the above questions through complete enumeration over all possible combinations of observed sequences and the corresponding possible sequences of hidden states, K^L combinations would have to explored [62], where L is the length of the sequence and K is the number of hidden states. For an average gene of $L = 1000$bp and 5 hidden states, $5^{1000} \approx 10^{698}$, total

calculations would have to be made. This is clearly intractable, even for far shorter sequences. Fortunately algorithms have been developed that make these calculations linear with respect to the length of the sequence [62].

Observation Likelihood

A key quantity when working with HMMs is the total likelihood of the observed sequence $P(X)$. Specifically, we want to know the probability of the observed sequence over *all possible parses* $\phi_i \in \Phi$. Here Φ refers to the set of all possible parses, and ϕ_i refers to an individual parse selected from Φ.

$$P(X) = \sum_{\phi_i \in \Phi} P(X, \phi_i) \tag{1.5}$$

To make it more clear what this quantity is, an example is provided. Let $X = ATC$, an arbitrary DNA sequence, and let the set of hidden states be two labels, $\{0, 1\}$. The set of *all possible parses* Φ is then a set of 2^3 elements, and represents all possibles ways that each observation in X can be labeled:

$$\Phi = \{\{000\}, \{001\}, \{010\}, \{011\}, \{100\}, \{101\}, \{110\}, \{111\}\}$$

A randomly chosen parse i from this set, $\phi_i \in \Phi$ could be

$$\phi_i = \{010\}$$

Using equation 1.4, there is a joint probability of observing $X = ATC$ and $Y = \phi_i = 010$.

$$P(X = ATC, Y = 010)$$

Given that X is fixed, there are still 7 other joint probabilities for each of the remaining possible parses

$$P(X = ATC, Y = 000)$$

$$\vdots$$

$$P(X = ATC, Y = 111)$$

The law of total probability (Eq. 1.6) states that, for two random variables A and B with a joint distribution $P(A, B)$ the marginal distribution $P(A)$ can be calculated by summing over all possible values of B [67].

$$P(A) = \sum_{B} P(A, B) \tag{1.6}$$

Within the context of our HMM, the probability of our observation $X = ATC$ can be calculated over all possible parses. To do this, we apply equation 1.6 and sum each of the joint probabilities above to get the probability that $X = ATC$, yielding equation 1.5.

This quantity is important for several reasons, particularly in learning problems such as parameter estimation, but as described above is impossible to calculate through a brute force approach, as the size of the set Φ combinatorally explodes.

Fortunately, there exists an algorithm which can efficiently calculate $P(X)$ with a computational complexity of LK^2, compared to the K^L complexity described above [62].

4.4 Forward and Backward Algorithms

Forward Algorithm

The forward algorithm recursively calculates the total probability of the sequence X starting from the first observation. This calculation is made feasible by using the conditional independence relationships demonstrated in Figure 1.3 and the joint probability distribution in equation 1.4.

Initialization

$$\alpha_i(t) = p(x_1|y_i)p(y_i) \tag{1.7}$$

Induction

$2 \leq t \leq L$

$$\alpha_i(t) = p(x_t|y_i) \sum_{j} p(y_i|y_j)\alpha_j(t-1) \tag{1.8}$$

Termination

$$P(X) = \sum_i \alpha_i(t = L) \tag{1.9}$$

Here i and j represent any single hidden state (in a programming environment, it represents the current hidden state under consideration in a `for` loop through the list of all hidden states). $\alpha_i(t)$ reads as the probability of the sequence X ending at state i at time t. The final term, summed over all hidden states, represents the total probability of the observed sequence X [62].

Backward Algorithm

The backwards probability β is the probability of seeing the observations from time t + 1 to the end, given that we are in state i at time t ($\beta_i(t)$) [62].

Initialization

$$\beta_i(t = L) = 1 \tag{1.10}$$

Induction

$T - 1 \leq t \leq 1$

$$\beta_i(t - 1) = p(x_{t-1}|y_i) \sum_j p(y_i|y_j)\beta_j(t) \tag{1.11}$$

Termination

$$P(X) = \sum_i \beta_i(t = 1) = \sum_i \alpha_i(t = L) \tag{1.12}$$

The forward and backward algorithms can be combined to calculate the probability of being in any hidden state i at any position within the sequence, *i.e.* $P(Y_t = i|x_{1:L})$. The reason both equations are used is that it allows for information prior to the observation as well as information posterior to observation to be considered.

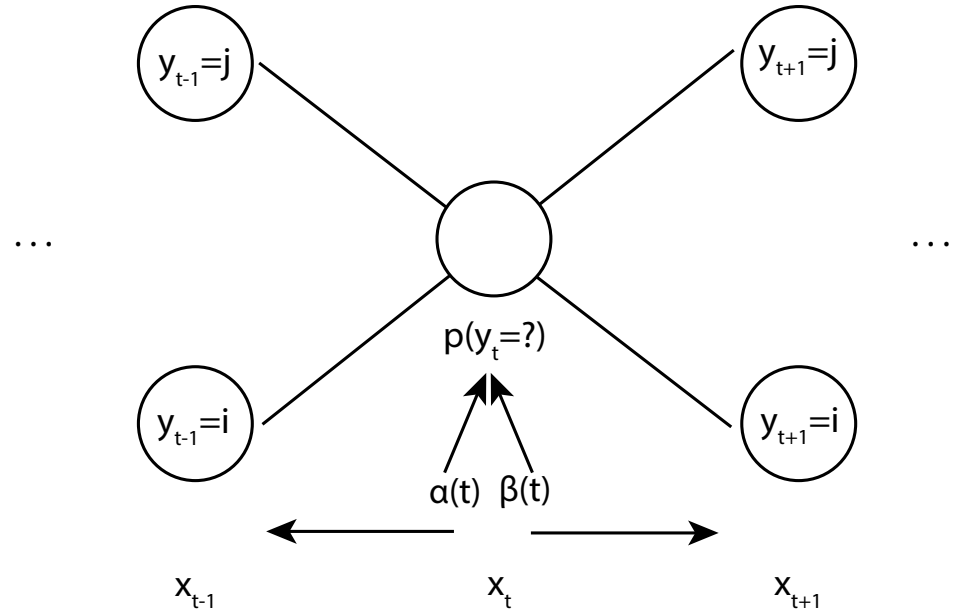

Figure 1.4: Information from the entire sequence $x_{1:T}$ is incorporated through the forward and backward probabilities. The formal definition of this probability is defined in equation 1.13

$$p(y_t = i|x_{1:L}) = \frac{\alpha_i(t)\beta_i(t)}{\sum_j \alpha_j(t)\beta_j(t)} \tag{1.13}$$

Figure 1.4 illustrates how information from the entire sequence is incorporated through the forward probability, α, and the backward probability, β, to get a *smoothed probability* for $y_t = i$.

4.5 Viterbi Algorithm

Perhaps one of the most useful tasks for an HMM is that of decoding, or identifying the most likely parse of the sequence, ϕ_{max}. That is, ϕ_{max} yields the highest joint probability $P(X, Y = \phi_{max})$ [62].

Mathematically, the Viterbi algorithm is very similar to the forward algorithm. Rather than taking the sum at each step in the recursion, instead the state transition that maximizes the probability at time t is stored in a list, and then a back-trace is calculated yielding the path of maximizing hidden states.

Initialization

Calculate the probability for each state at time $t = 1$, $\mu_i(t = 1)$, and initialize a list with each state, $V_i(t = 1)$

$$\mu_i(t = 1) = \max_i p(x_1|y_i)p(y_i) \tag{1.14}$$

$$V_i(t = 1) = i$$

Induction

Calculate the probability of being in each state i at time t, after transitioning from state j at time $t - 1$, then save the largest probability $\mu_i(t)$ for each state. $\mu_i(t)$ reads as the largest probability for the sequence X to end in state i at time t following a transition from state j at time $t - 1$ [62].

$$\mu_i(t) = \max_j \left[p(x_t|y_i)p(y_i|y_j)V_j(t - 1)\right] \tag{1.15}$$

Then, for each state i, save the state transition $j \to i$ that maximized $\mu_i(t)$.

$$V_i(t) = \operatorname*{argmax}_j \left[p(x_t|y_i)p(y_i|y_j)V_j(t - 1)\right]$$

Termination

The probability of the most likely end state is then

$$P^*(t = L) = \max_i \mu_i(t = L) \tag{1.16}$$

And the back-trace for the Viterbi most likely path is initialized at the last observation $t = L$:

$$V^*(t = L) = \operatorname*{argmax}_i P^*(t = L)$$

At time $t = L - 1$, we then retrieve $V^*(t = L - 1)$, which is the state that transitioned to the known maximizing state at the end of the sequence $V^*(t = L)$. This recursion is then repeated to the beginning of the sequence.

$$V^*(t - 1) = \operatorname*{argmax}_i \mu_i(t) \tag{1.17}$$

Once the list created from equation 1.17 has been inverted, as it is a back-trace, the path $V^* = \phi_{max}$, which is the parse $\phi \in \Phi$ that maximizes the joint probability of the HMM [62].

4.6 Baum-Welch Algorithm

A final common task for HMMs is to learn the set of parameters that maximize the likelihood of the model, *i.e.* what set of parameters θ, maximizes the probability of the observed sequence $P(X|\theta)$. The Baum-Welch algorithm does this through an expectation-maximization (EM) algorithm that estimates a new set of parameters, θ^*, by iterating through the forward and backward algorithms, and then calculation of the new likelihood $P(X|\theta^*)$ using the updated parameter values. This algorithm is guaranteed to converge to a *local* maximum – not necessarily a global maximum, meaning that better estimates for the parameters may exist. Furthermore, it is possible for the Baum-Welch algorithm to overfit the data, *i.e.* $P(X|\theta^{\texttt{MLE}}) > P(X|\theta^{\texttt{true}})$.

Initiation

Any value, including randomized values, can be initially assigned to the parameter values in $\theta = \{A, E, \pi\}$, however if *a priori* knowledge exists then that may speed convergence to the MLE. The Baum-Welch algorithm utilizes the forward and backward algorithm, and being described above, will be compressed here.

Forward and Backward Probabilities

1. Calculate the forward probabilities, $\alpha_i(t)$, for each state for each observation, representing the probability of seeing the observations $x_{1:t}$ and ending in state i at time t.

2. Calculate the backward probabilities, $\beta_i(t)$, for each state for each observation, representing the probability of the observations $x_{t+1:T}$ given starting state i at time t.

3. $\sum_i \alpha_i(T) = \sum_i \beta_i(1) = P(X|\theta)$

Update

1. Calculate the probability of being state i at time t, through Bayes theorem:

$$\gamma_i(t) = \frac{P(y_t = i, X|\theta)}{P(X|\theta)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_j \alpha_j(t)\beta_j(t)}$$

2. Calculate the probability of transitioning from state i at time t to state j at time $t+1$

$$\epsilon_{ij}(t) = P(y_t = i, y_{t+1} = j|X, \theta) = \frac{P(y_t = i, y_{t+1} = j, X|\theta)}{P(X|\theta)} = \frac{\alpha_i(t)A_{ij}\beta_j(t+1)E_j(x_{t+1})}{\sum_j \alpha_j(t)\beta_j(t)}$$

3. Calculate the expected number of transitions from state i to state j relative to the expected occurrence of i over the length of X, and update the transition matrix A accordingly.

$$A^*_{ij} = \frac{\sum_{t=1}^{T-1} \epsilon_{ij}(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

4. Calculate the expected number of occurrences of each observation category in the emission distribution (in the use-case in this dissertation, all k-mers), "v" (an observation in X) in state i, relative to the total occurrence of state i.

$$E^*_i(v) = \frac{\sum_{t=1}^{T} 1_{x_t=v}\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

Where $1_{x_t=v}$ is an indicator that equals 1 if $x_t = v$, and 0 otherwise.

REFERENCES

[1] J. C. Whisstock and A. M. Lesk, *Prediction of protein function from protein sequence and structure*, 2003.

[2] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *Journal of Molecular Biology*, 1997, ISSN: 00222836.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, 1990, ISSN: 00222836.

[4] T. J. Wheeler and S. R. Eddy, "Nhmmer: DNA homology search with profile HMMs," *Bioinformatics*, 2013, ISSN: 14602059.

[5] C. J. Brown, "The human XIST gene: analysis of a 17 kb inactive X-specific RNA that contains conserved repeats and is highly localized within the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 527–542, 1992 10, ISSN: 0092-8674.

[6] N. Brockdorff, "The product of the mouse Xist gene is a 15 kb inactive X-specific transcript containing no conserved ORF and located in the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 515–526, 1992 10, ISSN: 0092-8674.

[7] J. L. Rinn and H. Y. Chang, "Genome Regulation by Long Noncoding RNAs," *Annual Review of Biochemistry*, 2012, ISSN: 0066-4154.

[8] Y. Lee, C. Ahn, J. Han, H. Choi, J. Kim, J. Yim, J. Lee, P. Provost, O. Rådmark, S. Kim, and V. N. Kim, "The nuclear RNase III Drosha initiates microRNA processing," *Nature*, 2003, ISSN: 00280836.

[9] F. Yang, F. Yi, X. Han, Q. Du, and Z. Liang, "MALAT-1 interacts with hnRNP C in cell cycle regulation," *FEBS Letters*, 2013, ISSN: 00145793.

[10] V. Tripathi, J. D. Ellis, Z. Shen, D. Y. Song, Q. Pan, A. T. Watt, S. M. Freier, C. F. Bennett, A. Sharma, P. A. Bubulya, B. J. Blencowe, S. G. Prasanth, and K. V. Prasanth, "The nuclear-retained noncoding RNA MALAT1 regulates alternative splicing by modulating SR splicing factor phosphorylation," *Molecular Cell*, 2010, ISSN: 10972765.

[11] B. J. Raphael, R. H. Hruban, A. J. Aguirre, *et al.*, "Integrated Genomic Characterization of Pancreatic Ductal Adenocarcinoma," *Cancer Cell*, 2017, ISSN: 18783686.

[12] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles, J. Lagarde, L. Veeravalli, X. Ruan, Y. Ruan, T. Lassmann, P. Carninci, J. B. Brown, L. Lipovich, J. M. Gonzalez, M. Thomas, C. A. Davis, R. Shiekhattar, T. R. Gingeras, T. J. Hubbard, C. Notredame, J. Harrow, and R. Guigó, "The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression," *Genome Research*, 2012, ISSN: 10889051.

[13] C. C. Hon, J. A. Ramilowski, J. Harshbarger, N. Bertin, O. J. Rackham, J. Gough, E. Denisenko, S. Schmeier, T. M. Poulsen, J. Severin, M. Lizio, H. Kawaji, T. Kasukawa, M. Itoh, A. M. Burroughs, S. Noma, S. Djebali, T. Alam, Y. A. Medvedeva, A. C. Testa, L. Lipovich, C. W. Yip, I. Abugessaisa, M. Mendez, A. Hasegawa, D. Tang, T. Lassmann, P. Heutink, M. Babina, C. A. Wells, S. Kojima, Y. Nakamura, H. Suzuki, C. O. Daub, M. J. De Hoon, E. Arner, Y.

Hayashizaki, P. Carninci, and A. R. Forrest, "An atlas of human long non-coding RNAs with accurate 5 ends," *Nature*, 2017, ISSN: 14764687.

[14] G. K. Bogu, P. Vizán, L. W. Stanton, M. Beato, L. Di Croce, and M. A. Marti-Renom, "Chromatin and RNA Maps Reveal Regulatory Long Noncoding RNAs in Mouse," *Molecular and Cellular Biology*, 2016, ISSN: 0270-7306.

[15] I. Sarropoulos, R. Marin, M. Cardoso-Moreira, and H. Kaessmann, "Developmental dynamics of lncRNAs across mammalian organs and species," *Nature*, 2019, ISSN: 14764687.

[16] M. Sauvageau, L. A. Goff, S. Lodato, B. Bonev, A. F. Groff, C. Gerhardinger, D. B. Sanchez-Gomez, E. Hacisuleyman, E. Li, M. Spence, S. C. Liapis, W. Mallard, M. Morse, M. R. Swerdel, M. F. D'Ecclessis, J. C. Moore, V. Lai, G. Gong, G. D. Yancopoulos, D. Frendewey, M. Kellis, R. P. Hart, D. M. Valenzuela, P. Arlotta, and J. L. Rinn, "Multiple knockout mouse models reveal lincRNAs are required for life and brain development," *eLife*, 2013, ISSN: 2050084X.

[17] R. O. Niederer, E. P. Hass, and D. C. Zappulla, "Long noncoding RNAs in the yeast S. Cerevisiae," in *Advances in Experimental Medicine and Biology*, 2017.

[18] P. Johnsson, L. Lipovich, D. Grandér, and K. V. Morris, *Evolutionary conservation of long non-coding RNAs; Sequence, structure, function*, 2014.

[19] F. Sleutels, R. Zwart, and D. P. Barlow, "The non-coding Air RNA is required for silencing autosomal imprinted genes," *Nature*, 2002, ISSN: 00280836.

[20] K. C. Pang, M. C. Frith, and J. S. Mattick, *Rapid evolution of noncoding RNAs: Lack of conservation does not mean lack of function*, 2006.

[21] J. M. Kirk, S. O. Kim, K. Inoue, M. J. Smola, D. M. Lee, M. D. Schertzer, J. S. Wooten, A. R. Baker, D. Sprague, D. W. Collins, C. R. Horning, S. Wang, Q. Chen, K. M. Weeks, P. J. Mucha, and J. M. Calabrese, "Functional classification of long non-coding RNAs by k-mer content," *Nature Genetics*, 2018, ISSN: 15461718.

[22] D. Sprague, S. A. Waters, J. M. Kirk, J. R. Wang, P. B. Samollow, P. D. Waters, and J. M. Calabrese, "Nonlinear sequence similarity between the Xist and Rsx long noncoding RNAs suggests shared functions of tandem repeat domains," *RNA*, 2019, ISSN: 14699001.

[23] C. Chu, Q. C. Zhang, S. T. Da Rocha, R. A. Flynn, M. Bharadwaj, J. M. Calabrese, T. Magnuson, E. Heard, and H. Y. Chang, "Systematic discovery of Xist RNA binding proteins," *Cell*, 2015, ISSN: 10974172.

[24] M. D. Schertzer, K. C. Braceros, J. Starmer, R. E. Cherney, D. M. Lee, G. Salazar, M. Justice, S. R. Bischoff, D. O. Cowley, P. Ariel, M. J. Zylka, J. M. Dowen, T. Magnuson, and J. M. Calabrese, "lncRNA-Induced Spread of Polycomb Controlled by Genome Architecture, RNA Abundance, and CpG Island DNA," *Molecular Cell*, 2019, ISSN: 10974164.

[25] E. Hacisuleyman, C. J. Shukla, C. L. Weiner, and J. L. Rinn, "Function and evolution of local repeats in the Firre locus," *Nature Communications*, 2016, ISSN: 20411723.

[26] K. C. Wang, Y. W. Yang, B. Liu, A. Sanyal, R. Corces-Zimmerman, Y. Chen, B. R. Lajoie, A. Protacio, R. A. Flynn, R. A. Gupta, J. Wysocka, M. Lei, J. Dekker, J. A. Helms, and

H. Y. Chang, "A long noncoding RNA maintains active chromatin to coordinate homeotic gene expression," *Nature*, 2011, ISSN: 00280836.

[27] B. Moindrot, A. Cerase, H. Coker, O. Masui, A. Grijzenhout, G. Pintacuda, L. Schermelleh, T. B. Nesterova, and N. Brockdorff, "A Pooled shRNA Screen Identifies Rbm15, Spen, and Wtap as Factors Required for Xist RNA-Mediated Silencing," *Cell Reports*, 2015, ISSN: 22111247.

[28] N. Brockdorff, "Local tandem repeat expansion in Xist RNA as a model for the functionalisation of ncRNA," *Non-coding RNA*, 2018, ISSN: 2311553X.

[29] P. P. Amaral, M. E. Dinger, T. R. Mercer, and J. S. Mattick, *The eukaryotic genome as an RNA machine*, 2008.

[30] C. P. Ponting, P. L. Oliver, and W. Reik, *Evolution and Functions of Long Noncoding RNAs*, 2009.

[31] Y. Loewenstein, D. Raimondo, O. C. Redfern, J. Watson, D. Frishman, M. Linial, C. Orengo, J. Thornton, and A. Tramontano, *Protein function annotation by homology-based inference.* 2009.

[32] M. Guttman, I. Amit, M. Garber, C. French, M. F. Lin, D. Feldser, M. Huarte, O. Zuk, B. W. Carey, J. P. Cassady, M. N. Cabili, R. Jaenisch, T. S. Mikkelsen, T. Jacks, N. Hacohen, B. E. Bernstein, M. Kellis, A. Regev, J. L. Rinn, and E. S. Lander, "Chromatin signature reveals over a thousand highly conserved large non-coding RNAs in mammals," *Nature*, 2009, ISSN: 00280836.

[33] T. R. Mercer, M. E. Dinger, S. M. Sunkin, M. F. Mehler, and J. S. Mattick, "Specific expression of long noncoding RNAs in the mouse brain," *Proceedings of the National Academy of Sciences of the United States of America*, 2008, ISSN: 00278424.

[34] U. Perron, P. Provero, and I. Molineris, "In silico prediction of lncRNA function using tissue specific and evolutionary conserved expression," *BMC Bioinformatics*, 2017, ISSN: 14712105.

[35] S. Lefever, J. Anckaert, P. J. Volders, M. Luypaert, J. Vandesompele, and P. Mestdagh, "decodeRNA- predicting non-coding RNA functions using guilt-by-association," *Database : the journal of biological databases and curation*, 2017, ISSN: 17580463.

[36] D. Thiel, N. D. Conrad, E. Ntini, R. X. Peschutter, H. Siebert, and A. Marsico, "Identifying lncRNA-mediated regulatory modules via ChIA-PET network analysis," *BMC Bioinformatics*, 2019, ISSN: 14712105.

[37] R. A. Gupta, N. Shah, K. C. Wang, J. Kim, H. M. Horlings, D. J. Wong, M. C. Tsai, T. Hung, P. Argani, J. L. Rinn, Y. Wang, P. Brzoska, B. Kong, R. Li, R. B. West, M. J. Van De Vijver, S. Sukumar, and H. Y. Chang, "Long non-coding RNA HOTAIR reprograms chromatin state to promote cancer metastasis," *Nature*, 2010, ISSN: 00280836.

[38] K. M. Broadbent, D. Park, A. R. Wolf, D. Van Tyne, J. S. Sims, U. Ribacke, S. Volkman, M. Duraisingh, D. Wirth, P. C. Sabeti, and J. L. Rinn, "A global transcriptional analysis of Plasmodium falciparum malaria reveals a novel family of telomere-associated lncRNAs," *Genome Biology*, 2011, ISSN: 14747596.

[39] I. J. Paul and J. D. Duerksen, "Chromatin-associated RNA content of heterochromatin and euchromatin," *Molecular and Cellular Biochemistry*, 1975, ISSN: 03008177.

[40] R. R. Pandey, T. Mondal, F. Mohammad, S. Enroth, L. Redrup, J. Komorowski, T. Nagano, D. Mancini-DiNardo, and C. Kanduri, "Kcnq1ot1 Antisense Noncoding RNA Mediates Lineage-Specific Transcriptional Silencing through Chromatin-Level Regulation," *Molecular Cell*, 2008, ISSN: 10972765.

[41] T. B. Nesterova, S. Y. Slobodyanyuk, E. A. Elisaphenko, A. I. Shevchenko, C. Johnston, M. E. Pavlova, I. B. Rogozin, N. N. Kolesnikov, N. Brockdorff, and S. M. Zakian, *Characterization of the genomic Xist locus in rodents reveals conservation of overall gene structure and tandem repeats but rapid evolution of unique sequence*, 2001.

[42] G. Pintacuda, G. Wei, C. Roustan, B. A. Kirmizitas, N. Solcan, A. Cerase, A. Castello, S. Mohammed, B. Moindrot, T. B. Nesterova, and N. Brockdorff, "hnRNPK Recruits PCGF3/5-PRC1 to the Xist RNA B-Repeat to Establish Polycomb-Mediated Chromosomal Silencing," *Molecular Cell*, 2017, ISSN: 10974164.

[43] S. Somarowthu, M. Legiewicz, I. Chillón, M. Marcia, F. Liu, and A. M. Pyle, "HOTAIR Forms an Intricate and Modular Secondary Structure," *Molecular Cell*, 2015, ISSN: 10974164.

[44] C. Davidovich, X. Wang, C. Cifuentes-Rojas, K. J. Goodrich, A. R. Gooding, J. T. Lee, and T. R. Cech, "Toward a consensus on the binding specificity and promiscuity of PRC2 for RNA," *Molecular Cell*, 2015, ISSN: 10974164.

[45] C. Cifuentes-Rojas, A. J. Hernandez, K. Sarma, and J. T. Lee, "Regulatory Interactions between RNA and Polycomb Repressive Complex 2," *Molecular Cell*, 2014, ISSN: 10974164.

[46] X. Wang, K. J. Goodrich, A. R. Gooding, H. Naeem, S. Archer, R. D. Paucek, D. T. Youmans, T. R. Cech, and C. Davidovich, "Targeting of Polycomb Repressive Complex 2 to RNA by Short Repeats of Consecutive Guanines," *Molecular Cell*, 2017, ISSN: 10974164.

[47] J. Zhao, B. K. Sun, J. A. Erwin, J. J. Song, and J. T. Lee, "Polycomb proteins targeted by a short repeat RNA to the mouse X chromosome," *Science*, 2008, ISSN: 00368075.

[48] J. A. Simon and R. E. Kingston, *Mechanisms of Polycomb gene silencing: Knowns and unknowns*, 2009.

[49] M. Leeb, D. Pasini, M. Novatchkova, M. Jaritz, K. Helin, and A. Wutz, "Polycomb complexes act redundantly to repress genomic repeats and genes," *Genes and Development*, 2010, ISSN: 08909369.

[50] M. Almeida, G. Pintacuda, O. Masui, Y. Koseki, M. Gdula, A. Cerase, D. Brown, A. Mould, C. Innocent, M. Nakayama, L. Schermelleh, T. B. Nesterova, H. Koseki, and N. Brockdorff, "PCGF3/5-PRC1 initiates Polycomb recruitment in X chromosome inactivation," *Science*, 2017, ISSN: 10959203.

[51] Y. Hoki, N. Kimura, M. Kanbayashi, Y. Amakawa, T. Ohhata, H. Sasaki, and T. Sado, "A proximal conserved repeat in the Xist gene is essential as a genomic element for X-inactivation in mouse," *Development*, 2009, ISSN: 09501991.

[52] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, 1981, ISSN: 00222836.

[53] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras, "STAR: Ultrafast universal RNA-seq aligner," *Bioinformatics*, 2013, ISSN: 13674803.

[54] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, 2009, ISSN: 14747596.

[55] B. Haubold, "Alignment-free phylogenetics and population genetics," *Briefings in Bioinformatics*, 2014, ISSN: 14774054.

[56] S. Vinga and J. Almeida, "Alignment-free sequence comparison - A review," *Bioinformatics*, 2003, ISSN: 13674803.

[57] G. E. Sims and S. H. Kim, "Whole-genome phylogeny of Escherichia coli/Shigella group by feature frequency profiles (FFPs)," *Proceedings of the National Academy of Sciences of the United States of America*, 2011, ISSN: 00278424.

[58] K. Yang and L. Zhang, "Performance comparison between k -tuple distance and four model-based distances in phylogenetic tree reconstruction," *Nucleic Acids Research*, 2008, ISSN: 03051048.

[59] H. Yi and L. Jin, "Co-phylog: An assembly-free phylogenomic approach for closely related organisms," *Nucleic Acids Research*, 2013, ISSN: 03051048.

[60] J. Qi, B. Wang, and B. I. Hao, "Whole Proteome Prokaryote Phylogeny Without Sequence Alignment: A K-String Composition Approach," *Journal of Molecular Evolution*, 2004, ISSN: 00222844.

[61] I. Ulitsky, D. Burstein, T. Tuller, and B. Chor, "The average common substring approach to phylogenomic reconstruction," in *Journal of Computational Biology*, 2006.

[62] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 1989, ISSN: 15582256.

[63] A. Y. Ng and M. I. Jordan, "On discriminative vs. Generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in Neural Information Processing Systems*, 2002, ISBN: 0262042088.

[64] I. Sgouralis and S. Pressé, *An Introduction to Infinite HMMs for Single-Molecule Data Analysis*, 2017.

[65] L. Pachter, M. Alexandersson, and S. Cawley, "Applications of generalized pair Hidden Markov models to alignment and gene finding problems," *Journal of Computational Biology*, 2002, ISSN: 10665277.

[66] J. Henderson, "Finding genes in DNA with a Hidden Markov Model," *Journal of Computational Biology*, 1997, ISSN: 10665277.

[67] B. C. Brookes and A. N. Kolmogorov, "Foundations of the Theory of Probability," *The Mathematical Gazette*, 1951, ISSN: 00255572.

CHAPTER 2: XIST AND RSX

1 Introduction

The sex chromosomes of therian (eutherian and metatherian) mammals evolved from a pair of identical autosomes after the split of therian and monotreme mammals from their most recent common ancestor. Since that divergence, the Y chromosome has lost the large majority of its protein coding genes, creating a gene dosage imbalance between XY males and XX females. Part of the system that compensates this imbalance is a process known as X-chromosome Inactivation (XCI). Initiated early during female development, XCI results in the transcriptional silencing of one X chromosome in each somatic cell in female mammals. In eutherians, XCI is mediated by a long non-coding RNA (lncRNA) called Xist [1–4].

The silencing function of Xist is thought to be mediated by the concerted action of several domains of tandemly repeated sequence that are interspersed throughout its length. These repeat domains harbor binding sites for distinct subsets of proteins that, through incompletely understood mechanisms, help Xist achieve different aspects of its function. "Repeat A" binds proteins called SPEN and RBM15, and is required for the stabilization of spliced Xist RNA, and for *Xist* to silence actively transcribed regions of the X chromosome [5–12]. "Repeat B", and at least a portion of "Repeat C", bind HNRNPK to recruit the Polycomb Repressive Complex 1 (PRC1) to the inactive X chromosome [13, 14]. "Repeat E" binds many proteins, including CIZ1, and is required for the stable association of *Xist* with X-linked chromatin and for the sustained recruitment of Polycomb Repressive Complex 2 (PRC2) to the inactive X [15–17].

Intriguingly, metatherians (marsupials) may have convergently evolved their own lncRNA, *Rsx*, to mediate XCI in XX females [18]. *Rsx* shares no linear sequence similarity with Xist and is located in a different syntenic block on the marsupial X. Nevertheless, *Rsx* shares a number of surprising similarities with *Xist*. Both *Xist* and *Rsx* are expressed exclusively from the inactive X in females and are retained in the nucleus, forming what has been described as a "cloud-like" structure around their chromosome of synthesis. Moreover, both lncRNAs are spliced yet unusu-

ally long in their final processed form, and their expression correlates with the accumulation of histone modifications deposited by the Polycomb Repressive Complexes (PRCs) on the inactive X [18].

Studies performed over the last three decades indicate that Xist is required for normal XCI in eutherians [1–4]. Given the similarities between *Rsx* and *Xist*, it has been proposed that the marsupial *Rsx* is the functional analogue of *Xist* [18]. While this hypothesis has yet to be directly tested, expression of an *Rsx* transgene on a mouse autosome does, to a certain extent, induce local gene silencing, supporting the notion that *Rsx* harbors *Xist*-like function [18].

Despite their lack of linear sequence similarity, *Xist* and *Rsx* both harbor long, internal domains of tandemly repeated sequence [18, 19]. We recently discovered that evolutionarily unrelated lncRNAs that encode similar functions often harbor non-linear sequence similarity in the form of k-mer content, where a k-mer is defined as all possible combinations of a nucleotide substring of a given length k [20]. Below, we describe our use of k-mer based methods to investigate the possibility that the repeat domains in *Xist* and *Rsx* harbor non-linear sequence similarity that might be suggestive of shared function.

2 Results

2.1 *Xist* and *Rsx* share no local alignments

Xist and *Rsx* are both notable for their domains of highly repetitive sequence, which can be identified by aligning each lncRNA to itself and visualizing the alignment data as a dot plot ([21]). In mouse *Xist*, the four major repetitive regions are referred to as Repeats A, B, C, and E (Figure 1A; [22]). Repeats A, B, and E are conserved in eutherian mammals, whereas Repeat C appears to be specific to murid rodents (Figure 1C;[23, 24]). In human *Xist*, the four major repetitive regions are referred to as Repeats A, B, D, and E (Figure 1B; [25]). Relative to mouse, human Repeat B is comprised of two shorter Repeat B-like regions that appear to have been disrupted by insertion (Figure 1B; [23, 24]). Human Repeat D is comprised of eight core repeats flanked by several additional repeats that exhibit partial similarity to its core (Figure 1B; [23–25]). While Repeat D is absent in murid rodents (Figure 1C), Repeat D-like sequence appears in many other mammals ([23, 24]).

In contrast to *Xist*, which is mostly comprised of non-repetitive sequence, nearly all of the

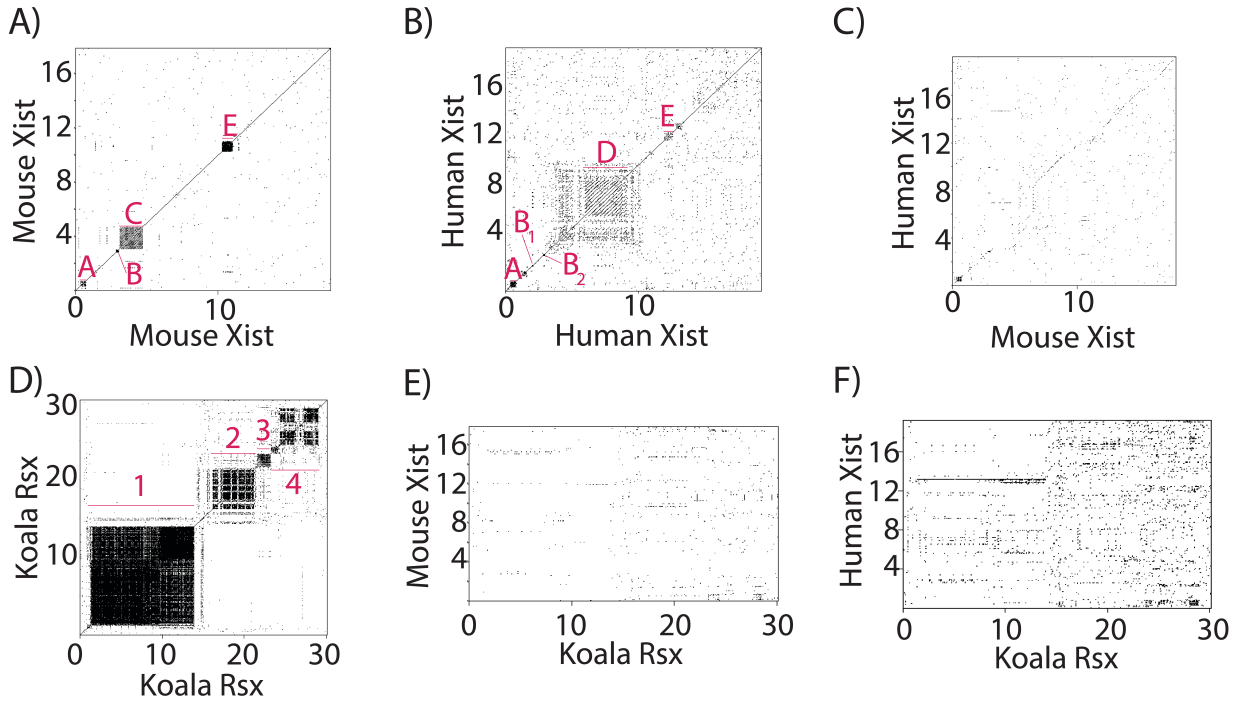

Figure 2.1: Dot plots comparing mouse and human *Xist* and *Rsx* to themselves and to each other. The location of repeat domains in all three lncRNA-to-self plots are marked with red bars and names/numbers.

sequence in *Rsx* can be assigned to one of four repetitive domains (Figure 1D; [19]). Here, we refer to the repetitive domains in *Rsx* as Repeats 1 through 4. It has been suggested that *Rsx* Repeat 1 is functionally analogous to *Xist* Repeat A, because both repeats are the first to occur in each lncRNA, and because both repeats contain GC-rich elements [18, 19]. Beyond this observation, little is known about the repetitive regions in *Rsx* and how they might relate to those in *Xist*. Hypothesizing that the repeat domains in *Xist* and *Rsx* recruit similar subsets of proteins, we expected that dot plots comparing the sequence of *Xist* to the sequence of *Rsx* would reveal regions of sequence similarity. However, this was not the case (Figure 1E, F), nor was significant similarity between *Xist* and *Rsx* detected using BLASTN or the hidden-markov based nhmmer [26, 27].

2.2 Tandem repeats domains in *Xist* and *Rsx* share k-mer content

We hypothesized that sequence similarity between *Xist* and *Rsx* might become apparent using an algorithm we recently developed to detect sequence similarity between evolutionarily unrelated lncRNAs [20]. In our algorithm, called SEEKR (Sequence evaluation through k-mer representation), groups of lncRNAs are compared to each other by counting the number of occurrences of each k-mer of a given length k in each lncRNA, then normalizing k-mer counts by the length of the lncRNA in question, and finally calculating a z-score for each k-mer in each lncRNA. The list of z-scores for each k-mer in a lncRNA is referred to as its "k-mer profile" and represents the abundance of each k-mer in the lncRNA relative to the abundance of each k-mer in the other lncRNAs that were analyzed as part of the group. In SEEKR, k-mer profiles from lncRNAs of interest are compared to each other using Pearson's correlation. We previously demonstrated that SEEKR can be used to quantify the similarity between any number of lncRNAs regardless of their evolutionary relationships or differences in their lengths, and that similarities in k-mer profiles correlated with lncRNA protein binding potential, subcellular localization, and *Xist*-like repressive activity. A major strength of SEEKR is that it ignores positional information in similarity calculations, allowing it to quantify non-linear sequence relationships [20].

In order to compare *Xist* and *Rsx* via SEEKR, we calculated the k-mer profile at $k = 4$ of individual repeat domains in mouse *Xist* and koala *Rsx*, using all mouse lncRNAs from GENCODE as a background set to derive the mean and standard deviation of the counts for each k-mer [28]. The mechanisms through which *Xist* functions have been most extensively studied in mouse [4]. For this reason, we primarily used the repetitive regions from mouse *Xist* as search features in this work. However, because of the conservation of Repeat D-like domains in non-murid eutherian mammals ([23, 24]), we also included the sequence of human *Xist* Repeat D in our analyses. We used the sequence from koala *Rsx* as our exemplar, owing to the high-quality of the koala genome build relative to builds from other marsupials [19].

In our previous work, we found that SEEKR performed best when the length of the lncRNA or lncRNA fragment being studied was similar to 4^k, i.e. the total number of possible k-mers at k-mer length k. In tests of *Xist*-like repressive activity, we found that comparisons of lncRNAs

using k-mer lengths of 7+ underperformed relative to comparisons using smaller k-mer lengths, owing to the fact that most annotated lncRNAs are much less than 4^7 (16384) nucleotides long, and k-mer profiles of individual lncRNAs at $k = 7$ (≈16384 possible k-mers) are dominated by "0" values [20]. Based on this observation, and because Repeats A and B, two essential repetitive regions within *Xist* [7, 11–14], are each about 4^4 (256) nucleotides in length, we reasoned that k-mer profiles at $k = 4$ (4^4=256 possible k-mers) would provide a reasonable estimate of sequence complexity for the repeats without being dominated by "0" values.

We also noted that relative to most lncRNAs, k-mer content in the repetitive regions of *Xist* and *Rsx* was skewed (Figure S2A). We therefore elected to $\log_2$-transform z-scores in k-mer profiles prior to comparison via Pearson's correlation, recognizing that this transformation would reduce skew and allow us to evaluate similarity in the context of a log-linear scale (Figure S2B).

The individual repeat domains in *Xist* and *Rsx* vary substantially in terms of their length and sequence complexity. *Xist* repeats tend to be shorter and lower in overall complexity than repeats in *Rsx* (Figure 2A, B). Despite these differences, using SEEKR, we identified substantial levels of similarity between the repeat domains of *Xist* and *Rsx*. The Repeat A region of *Xist* was most similar to *Rsx* Repeat 4, exhibited a weak positive correlation with Repeat 2, and had negative correlations with *Rsx* Repeats 1 and 3 (Pearson's r of 0.21, for Repeat A versus Repeat 4, respectively, and r of -0.02, 0.09, and -0.08 for Repeats 1, 2, and 3, respectively; Figure 2C). In contrast, *Xist* Repeat B was most similar to *Rsx* Repeat 1 and had negative correlations with *Rsx* Repeats 2 through 4 (Pearson's r of 0.33 for Repeat B versus Repeat 1; Figure 2C). Repeat C, which is specific to murid rodents (i.e. it is not found in other eutherians), had no appreciable correlation with any *Rsx* repeat, whereas human Repeat D had positive correlations with *Rsx* Repeat 1 and 4 (r of 0.20, 0.12, respectively; Figure 2C). The k-mer profile of *Xist* Repeat E had positive correlations that increased progressively in *Rsx* Repeats 2, 3, and 4. (Pearson's r of 0.15, 0.25, 0.40, respectively; Figure 2C).

We sought to quantify the strength of the similarity between repeat domains in *Xist* and *Rsx* relative to other mouse lncRNAs. To do this, we used Pearson's correlation to compare the k-mer profile of each *Xist* repeat domain to the k-mer profiles of the set of spliced GENCODE M18 mouse lncRNAs [28]. We compared this distribution of Pearson's r values to the r value obtained when comparing each *Xist* repeat to each *Rsx* repeat.

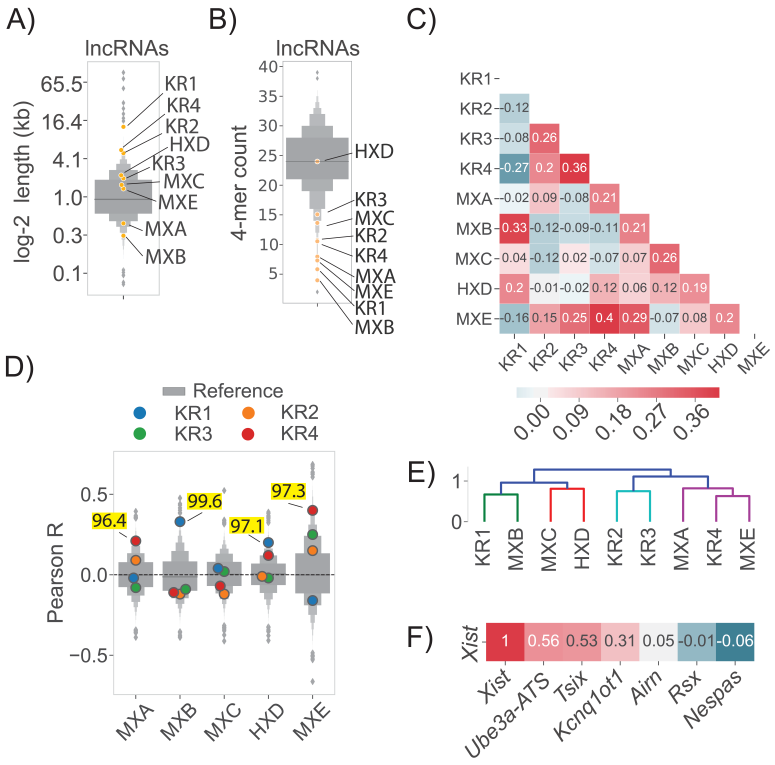

Figure 2.2: Length of *Xist* and *Rsx* repeat domains and (B) sequence complexity estimated by the number of unique 4-mers that constitute 25% of the total 4-mer counts in a transcript, each relative to all other GENCODE M18 lncRNA transcripts. For panels A – E: "M", "H", and "K" signify mouse, human, and koala, respectively, "X" and "R" signify *Xist* and *Rsx*, respectively, and the final letter or number in each abbreviation signifies the repeat domain in question. (C) Correlation matrix displaying the Pearson's r value derived from comparing k-mer profiles at $k = 4$ of each of the four repeat domains in koala *Rsx* (Repeats 1 – 4), the major repeats in mouse *Xist* (Repeats A, B, C, E), and human *Xist* Repeat D. The set of mouse lncRNAs from GENCODE was used to derive mean and standard deviation values for length normalized abundance of each k-mer. (D) Similarity of repeat domains in *Xist* and *Rsx* relative to all lncRNA transcripts in the mouse GENCODE M18 database [28]. Each subplot shows the distribution of Pearson's r values describing the similarity between the *Xist* repeat in question and the set of GENCODE lncRNA transcripts. Similarities between *Xist* and *Rsx* that are above the 95^{th} percentile of similarity for all mouse lncRNAs are highlighted in yellow. (E) The correlation matrix in (A) subject to hierarchical clustering. Colors represent clusters for all descendent links beneath the first node in the dendrogram with distance less than 70% of the largest distance between all clusters. (F) SEEKR- derived similarity (in the form of Pearson's r; [20]) between full-length *Xist*, other cis- repressive lncRNAs in mouse, and koala *Rsx* [19]

.

This analysis revealed striking similarities between the repeat domains of *Xist* and *Rsx*. *Xist* Repeat B was more similar to *Rsx* Repeat 1 than it was similar to 99.6% of all lncRNAs (similarity ranked 65th out of 17523 comparisons), despite the fact that the two repeats differ in length

by 50-fold (Figures 2A, D; Tables S1, S2). *Xist* Repeat A was more similar to *Rsx* Repeat 4 than it was similar to 96.4% of all other lncRNAs (its similarity ranked 626nd out of 17523 comparisons), *Xist* Repeat D was more similar to *Rsx* Repeat 1 than it was similar to 97.1% of all other lncRNAs (its similarity ranked 515th out of 17523 comparisons), and *Xist* Repeat E was more similar to *Rsx* Repeat 4 than it was similar to 97.3% of all other lncRNAs (its similarity ranked 467th out of 17523 comparisons; Figure 2D; Table S1, S2). No other repeat domains in *Xist* and *Rsx* fell above the 95^{th} percentile in terms of their similarity to each other. Similar trends were observed when we used k-mer lengths $k = 4$, 5 and 6 for this analysis (Figure S3A).

Current models suggest that the tandem repeats in *Xist* have distinct functions [1–4]. Thus, we were surprised to find that the repeat domains within *Xist* also exhibited high levels of similarity to each other (Table S1, S2). Repeat A was more similar to Repeat E than it was similar to 99.6% of all lncRNAs. Likewise, Repeats B and C were more similar to each other than they were similar to 97.8% and 99.6% of all other lncRNAs, respectively. Finally, Repeats C and D were more similar to each other than they were similar to 97.0% and 96.2% of all other lncRNAs, respectively (Table S1, S2).

The similarities between specific domains of *Xist* and *Rsx* were also evident in an unsupervised hierarchical cluster of the matrix from Figure 2C. *Xist* Repeat B and *Rsx* Repeat 1 formed a basal cluster which joined with a second basal cluster comprising *Xist* Repeat C and *Xist* Repeat D. *Rsx* Repeat 4 and *Xist* Repeat E formed a basal cluster that that joined with *Xist* Repeat A. This multilevel cluster (*Rsx* Repeat 4, *Xist* Repeat E, and *Xist* Repeat A) joined with another basal cluster comprising *Rsx* Repeats 2 and 3 (Figure 2E).

At k-mer length $k = 4$, Pearson's correlation with and without log-transformation of k-mer z-scores, as well as Spearman's correlation of non-transformed z-scores, detected similar relationships between *Xist* and *Rsx* repeat domains (Figure S4). While the similarities between individual domains were still evident at higher k-mer lengths, particularly when using Pearson's correlation of log-transformed k-mer counts, the clustering patterns that we observed at k-mer length $k = 4$ began to dissolve (Figure S4). At high k-mer lengths, Spearman's correlation was the least informative method of comparison, owing to the large number of "zero" values that populate k-mer profiles at these lengths (Figure S4). Thus, to a certain extent, the similarities in the repeat domains of *Xist* and *Rsx* are detectable regardless of prior assumptions about log-linear,

linear, and monotonic relationships between k-mer profiles. However, the most robust similarities are detected using Pearson's correlation of log-transformed k-mer counts (Figure S4). We observed that the similarities between *Xist* and *Rsx* were obscured when the k-mer profiles of the full-length lncRNAs were compared to each other (Pearson's r of -0.01 for the comparison of full-length *Xist* to full-length *Rsx*; Figure 2F). This loss of similarity highlights the utility of domain-based similarity searches, particularly for lncRNAs whose functional domains may comprise a fraction of their overall length. The dissimilarity between k-mer profiles of full-length *Xist* and full-length *Rsx* likely stems from the fact that virtually all of *Rsx* is comprised of repetitive sequence domains that harbor limited k-mer diversity relative to the non-repetitive sequence of *Xist* (Figure 2B and compare Figures 1A-C to 1D).

2.3 Tandem repeat sequence properties

Qualitative similarities between *Xist* and *Rsx* repeat domains were also revealed using MEME to visualize motifs that were enriched within individual domains [29]. In *Xist* Repeat A, MEME identified one motif comprised of short runs of G and C nucleotides and one motif most notable for runs of T nucleotides (Figure 3A). Similar patterns were seen in the motifs enriched in *Rsx* Repeat 4 (Figure 3B). The single motif from Repeat B was almost exclusively comprised of two tandemly arranged 'GCCCC' motifs, and motifs containing runs of 'G' and 'C' nucleotides could be seen in *Rsx* Repeat 1 (Figure 3A, B). The pyrimidine-rich runs that were characteristic of *Xist* Repeat E were also observed *Rsx* Repeat 4 (Figure 3A, B). *Rsx* Repeat 2 was unique in its enrichment of AAAG and GAAA motifs (Figure 3B). Several of the repeat domains in *Xist* and *Rsx* could be distinguished by the presence of k-mers comprised of runs of individual nucleotides that extended for two or more consecutive positions (such as AA, CC, GG, or TT; File S1). Similar to enriched motifs, k-mers containing mononucleotide runs may function to recruit different subsets of RNA binding proteins [30, 31]. We therefore sought to quantify the enrichment of k-mers containing mononucleotide runs in the repeat domains of *Xist* and *Rsx*, reasoning that this analysis might provide insight into function. Similar to what we observed in our motif analysis (Figure 3), *Rsx* Repeat 2 had the highest length-normalized abundance of polyA k-mers, followed closely by *Rsx* Repeat 3 (Figure 4A). Repeat B, which is only 250 nucleotides long and is almost entirely comprised of polyC sequence, had the highest length-normalized abundance of polyC

Figure 2.3: The top three de novo motifs identified by MEME in *Xist* repeats (panel A; all repeats from mouse *Xist* except for Repeat D, which is the human sequence), and in the four repeats in koala (B) and opossum (C) *Rsx*. The length in nucleotides of each repeat is shown in parentheses below the repeat name. The number of matches to each motif, as well as the expectation-value for that number, is shown below each motif logo. Some repeats had less than three motifs detected by MEME.

k-mers, followed by *Rsx* Repeat 1, and *Xist* Repeats C, D, and A (Figure 4B). Mouse Repeat A had the highest length-normalized abundance of polyG k-mers, followed by *Rsx* Repeats 1, 2, and 4 (Figure 4C). *Xist* Repeats A and E, as well as *Rsx* Repeats 3 and 4 had the highest length-normalized abundance of polyT k-mers, reflecting the high degree of SEEKR-detected similarity between these regions (Figure 4D). Similar trends were detected when we used k-mer lengths k = 4, 5, and 6 for this analysis (Figure S3B). Thus, certain *Xist* and *Rsx* repeat domains share similarity in their overall k-mer profiles (Figure 2), in their enriched motifs (Figure 3), and in their enrichment in subsets of low-complexity k-mers that are comprised of mononucleotide runs (Figure 4). The repeat domains also harbor differences in sequence composition that are consistent with their lack of alignment via methods designed to detect linear sequence similarity (Figure 1).

2.4 Enrichment of HNRNPK tandem repeat motifs in *Xist* and *Rsx*

Xist Repeat B is known to bind a protein called HNRNPK, and this binding activity is essential for *Xist* to recruit PRC1 to the inactive X chromosome [14]. Given the quantitative and qualitative sequence similarities between *Xist* Repeat B and *Rsx* Repeat 1 (Figures 2 – 4), we sought to compare HNRNPK-binding potential between the two repeats using two conceptually distinct approaches. First, we weighted z-scores of individual k-mers in all *Xist* and *Rsx* repeat domains by the probability that the k-mer would occur in the Position-Weight-Matrix (PWM) describing the HNRNPK-binding motif from ([31]; see PWM in Figure 4E). We then summed HNRNPK-scaled z-scores over each repeat, and plotted the results in a manner similar to Figure 4A-D. In this analysis, a positive sum indicates that k-mers matching the HNRNPK PWM occur more frequently in the domain in question than they occur in other lncRNAs in the GENCODE database.

On a length-normalized basis, *Xist* Repeats B, C, A, and D, in descending order, had positive sums of HNRNPK-scaled z-scores. Repeat 1 was the only repeat in *Rsx* to have a positive sum, perhaps consistent with a role in recruiting HNRNPK to *Rsx* (Figure 4E). The sum of HNRNPK-scaled z-scores in *Rsx* Repeat 1 was lower than the sums in *Xist* Repeats B, C, and A (Figure 4E), which might be taken as evidence that on a length-normalized basis, *Xist* Repeats B, C, and A have a higher density of k-mers that are likely to bind HNRNPK than *Rsx* Repeat 1 or any other *Rsx* repeat. However, at 13 kb in length, *Rsx* Repeat 1 is ≈50 times longer than *Xist* Re-

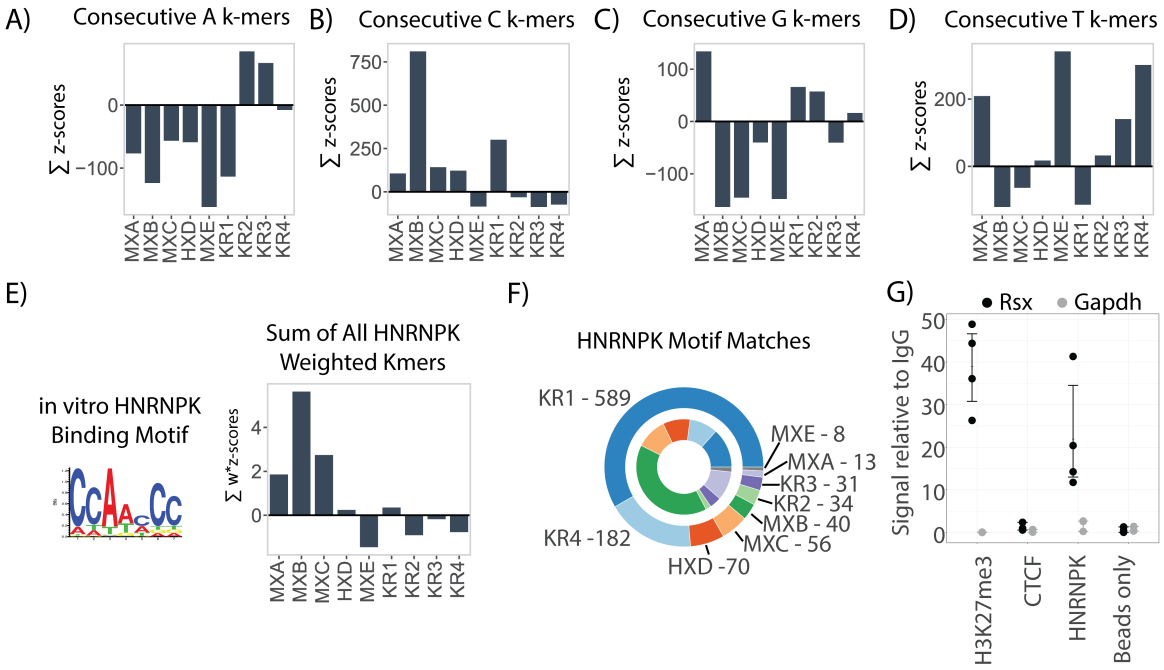

Figure 2.4: The sum of z-scores in each repeat for k-mers containing consecutive (A) A, (B) C, (C) G, and (D) T nucleotides. For this analysis we defined "consecutive" as at least two consecutive nucleotides of the specified identity, and used k-mer length k = 5 (Methods). Repeat abbreviations as in Figure 2. (E) The sum of z-scores for all k-mers in each *Xist* and *Rsx* repeat domain after weighting the k-mers by the likelihood with which they fit the consensus HNRNPK-binding motif. Motif logo that describes the consensus HNRNPK-binding motif obtained from (Ray et al., 2013) is also shown. (F) Component arcs of outer circle indicate the proportion and number of HNRNPK-binding motif matches detected by FIMO ($p < 0.01$) in each *Xist* and *Rsx* repeat domain. Component arcs of inner circle indicate the proportion of motif matches in each repeat domain normalized for domain length. Repeat abbreviations as in Figure 2. (G) *Rsx* enrichment relative to IgG control after RNA IP-qPCR in cultured fibroblasts from female M. domestica. For each antibody, left (black) is enrichment of *Rsx*, right (grey) is enrichment of Gapdh. The histone modification H3K27me3 is enriched on the inactive X in marsupials, so an association with *Rsx* was expected. CTCF has nanomolar affinity for RNA and along with "bead only"/no-antibody IP serves as a negative control demonstrating IP specificity [32]. Dots represent values from replicate RNA IP experiments; error bars represent bootstrap 95 CI.

peat B, and is over half of the length of full-length *Xist* itself [19, 22, 25]. Thus, we also counted the absolute number of matches to HNRNPK-binding motifs in *Xist* and *Rsx* repeats. *Rsx* Repeat 1 had 15 times more matches to HNRNPK-binding motifs than did *Xist* Repeat B (589

matches in Repeat 1 compared to 40 matches in Repeat B; Figure 4F; [29]). *Rsx* Repeat 4 also had a large number of matches to HNRNPK-binding motifs (182 matches), and human Repeat D and mouse Repeat C each had more HNRNPK-binding sites than Repeat B (70 and 56 matches, respectively, compared to 40 in Repeat B; Figure 4F). CLIP performed in mouse and human cells supports a direct association between HNRNPK and Repeat C and Repeat D, respectively ([33, 34]). Collectively, these data support the ideas that mouse Repeat C and human Repeat D co-operate with Repeat B in recruiting HNRNPK to *Xist*, and suggest that *Rsx* Repeat 1, and to a lesser extent, *Rsx* Repeat 4, could also recruit HNRNPK to *Rsx*.

We next used RNA immunoprecipitation (RNA IP) followed by RT-qPCR to determine whether we could detect evidence of HNRNPK association with *Rsx*. In fibroblast cells derived from a female gray short-tailed opossum, Monodelphis domestica, we found that HNRNPK IP enriched for *Rsx* 20-fold over IgG control IPs (Figure 4G). This enrichment was similar to that seen for an IP using an antibody that detects histone H3-lysine27-trimethylation (H3K27me3), a modification known to be enriched on the opossum inactive X (Figure 4G; (Wang et al., 2014)). Gapdh mRNA was not enriched by IP of HNRNPK or H3K27me3 (Figure 4G). IP of CTCF, a protein that binds RNA with nanomolar affinity in a sequence non-specific manner, showed neither *Rsx* nor Gapdh enrichment (Figure 4G; [32]). Leaving out HNRNPK antibody prior to performing IP and qPCR also led to a loss of *Rsx* signal ("beads only" in Figure 4G). DNase-treated input RNA (no reverse transcription control) did not yield signal in qPCR assays, indicating DNase digestion prior to cDNA synthesis and qPCR proceeded to completion (not shown). These data support our computational analyses and suggest that HNRNPK associates with *Rsx* in marsupial cells.

2.5 Conservation of *Xist* and *Rsx* Repeats

Considering that not all of the repeat domains in *Xist* exhibit conservation across eutherian mammals, we sought to determine whether or not the repeat domains in koala *Rsx* were conserved in another marsupial. *Rsx* was originally identified in opossum [18], but the most current assembly of the opossum genome (mondom5) harbors significant gaps within the sequence of *Rsx*.

To assemble a complete sequence of opossum *Rsx* for comparison to koala, we used Oxford Nanopore technology to sequence two Bacterial Artificial Chromosomes (BACs) that encom-

passed the opossum *Rsx* locus (VMRC18-839J22 and VMRC18-303M7). De novo assembly and polishing of sequence reads identified a single 235,139 base contig aligning to chrX that had on average a 0.5% error rate with the mondom5 assembly (File S3). Our assembly filled in 16620 bases of unannotated sequence in the *Rsx* locus, 361 bases of which were a part of the spliced *Rsx* lncRNA annotation from [18].

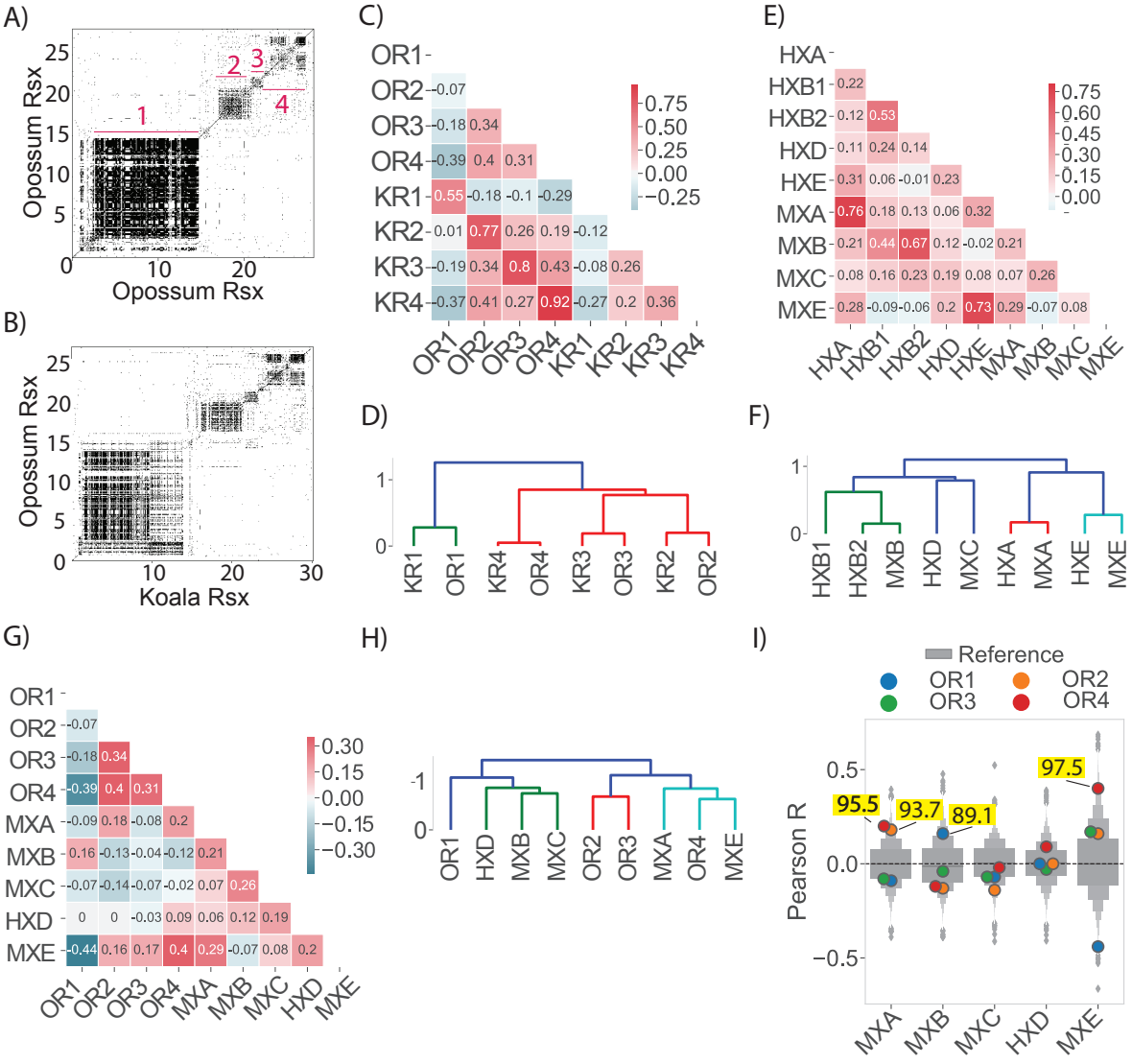

Figure 2.5: (A,B) Dot plots of opossum *Rsx* aligned to (A) itself or (B) koala *Rsx*. (C) Similarity between repeat domains in koala and opossum *Rsx* as calculated in Figure 2C. (D) Hierarchical cluster of similarity values from (C). (E) Similarity between repeat domains in mouse and human *Xist* as calculated in Figure 2C. (F) Hierarchical clustering of similarity values from (E). G) Similarity between repeat domains in opossum *Rsx* and *Xist* repeat domains as calculated in Figure 2C. (H) Hierarchical cluster of similarity values from (G). (I) Percentiles for Pearson's R for opossum *Rsx* repeat domains compared to each *Xist* repeat domain as in Figure 2D. Numbers mentioned in the body of the manuscript are highlighted in yellow.

Alignment of this ungapped assembly of spliced opossum *Rsx* to koala *Rsx* revealed high levels of similarity between their repeat domains in a dot plot analysis (Figure 5A, B). This similarity could also be seen at the level of *k*-mers (Figure 5C, D), and by extraction of enriched motifs using MEME (Figure 3B, C). Opossum and koala, which are members of distantly related American and Australian marsupial families, respectively, diverged approximately 82 million years ago [35]. By comparison, mouse and human are separated by approximately 90 million years of evolution [35]. Repeat domains 1 through 4 in opossum and koala *Rsx* exhibited levels of sequence similarity that approximated or exceeded the similarity found between the repeat domains in mouse and human *Xist* (with the exception of Repeat C/Repeat D; Figure 5C-F). Thus, the repeat domains in *Rsx* appear to be at least as conserved between distantly related marsupials as the repeat domains in *Xist* are conserved among eutherians.

Next, we compared the *k*-mer contents of repeat domains in *Xist* to the *k*-mer contents of repeat domains in opossum *Rsx*. We identified a level of similarity (Figure 5G, H, I) that mirrored the similarity we found between repeat domains in *Xist* and koala *Rsx* (Figure 2B, D, E). *Xist* Repeat A was most similar to opossum Repeats 2 and 4 (93.7^{th} and 95.5^{th} percentile relative to all other mouse lncRNAs, respectively); *Xist* Repeat B was most similar to opossum Repeat 1 (89.1^{st} percentile relative to all other lncRNAs); and *Xist* Repeat E was most similar to opossum Repeat 4 (97.5^{th} percentile relative to all other lncRNAs; Figure 5I). Thus, the major repeat domains in *Rsx* are conserved between opossum and koala, and the repeat domains in *Rsx* from both marsupials harbor *k*-mer contents similar those in repeat domains from mouse and human *Xist*.

2.6 Multiple protein-binding motifs are enriched to extreme levels in *Xist* and *Rsx* repeat domains

We examined the extent to which *Xist* and *Rsx* repeat domains were enriched for sequence motifs known to recruit RNA binding proteins, hypothesizing that the patterns of enrichment might provide additional insight into similarities between the two lncRNAs. For this analysis, we downloaded PWMs for all mammalian RNA binding proteins available in the CISBP-RNA database [31], and for each PWM in each repeat, we quantified enrichment by weighting *k*-mer *z*-scores by the probability that the *k*-mer matched the PWM, then calculating the sum of those weights, as we did for the HNRNPK PWM in Figure 4E. To gauge the extent of enrichment rel-

ative to other mouse lncRNAs, we determined the percentile rank of the sum for each PWM in each repeat relative to the sums generated from the same PWM-weighting procedure performed on all mouse lncRNAs. We then hierarchically clustered repeat domains from *Xist* and *Rsx* based on the percentile ranks of motif enrichment for each domain. The results of these analyses are shown in Figure 6A.

Using ranked enrichment of protein-binding motifs as a metric for hierarchical clustering, we identified the same relationships between *Xist* and *Rsx* repeat domains as we did when we hierarchically clustered domains by their k-mer content alone (dendrogram in Figure 6A compared to dendrograms in Figures 2E and 5H). Via protein-binding motif enrichment, *Xist* Repeats B, C, and D formed a second order cluster that next joined with *Rsx* Repeat 1. This clustering order is the same as that detected using k-mer content alone (Figure 6A vs. 2E and 5H). Likewise, protein-binding motif enrichment grouped Repeats A and E together with *Rsx* Repeat 4, while *Rsx* Repeats 2 and 3 formed a separate cluster that joined with the Repeat-A-E-4 cluster. Again, similar clustering patterns were obtained based purely on k-mer content (Figure 6A vs. 2E and 5H). We note that the motifs used to create these clusters are limited in complexity and are capable of recruiting different proteins depending on cellular and sequence contexts [30, 31]. Thus, the enrichment of a particular protein-binding motif in an individual *Xist* or *Rsx* repeat domain does not provide direct evidence that the protein binds to the domain. Nevertheless, these results are consistent with the notions that lncRNA k-mer content encodes information about protein-binding potential [20], and that the various repeats in *Xist* and *Rsx* encode function through the concerted recruitment of multiple RNA-binding proteins.

A closer inspection of the protein-binding motifs that were enriched in each repeat domain yielded several insights. First, our motif analysis uncovered relationships between repeat domains that were not obvious from direct k-mer comparisons. For example, both human and mouse *Xist* Repeat B were enriched in motifs that recruit polyC-binding proteins and little else (Figure 6A; Table S4). *Rsx* Repeat 1, which most closely resembles *Xist* Repeat B at the level of k-mers, was also enriched in polyC-binding motifs, in both koala and opossum (Figure 6A; Table S4). However, Repeat 1 from koala and opossum *Rsx* were also enriched in many motifs that were absent in Repeat B, such as motifs that bind the proteins SRSF1, SRSF9, and RBM5 (Figure 6A; Table S4). In addition, while both pure k-mer analysis and motif analysis identified similarities

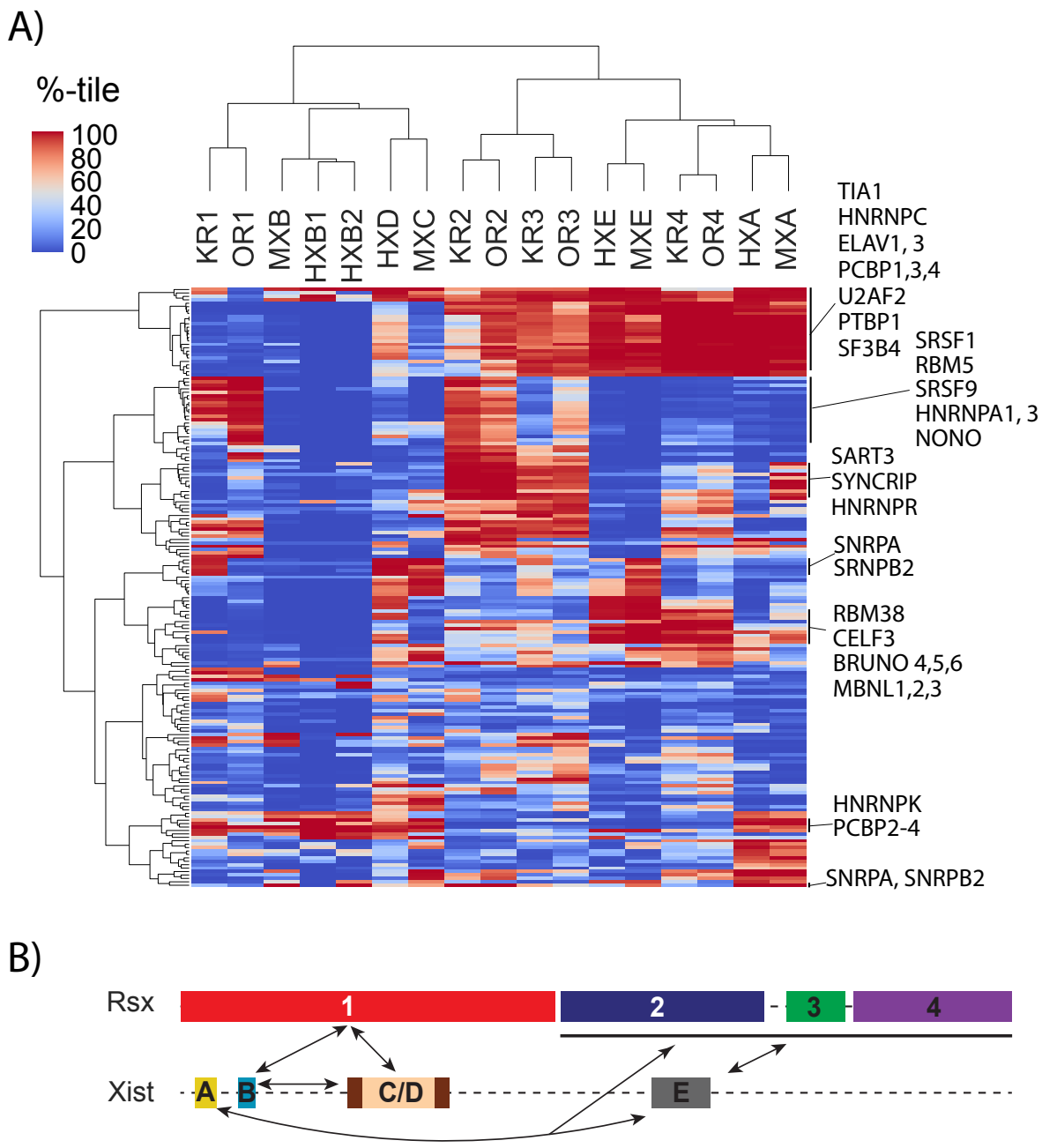

Figure 2.6: (A) Hierarchically clustered heatmap of PWM weighted z-scores for each repeat in *Xist* and *Rsx*, expressed as a percentile relative to the set of all GENCODE M18 lncRNA annotations. (B) Regions predicted to have similar protein-binding functions in *Xist* and *Rsx*. Arrows connect domains in each lncRNA that have similar k-mer and motif contents.

between *Xist* Repeats A and E and *Rsx* Repeats 2, 3, and 4, our motif analysis also identified similarities exclusive to pairs of domains within this group, such as similarities between *Rsx* Repeats 2 and 3 and similarities between *Xist* Repeat E and *Rsx* Repeat 4 (Figure 6A; Table S4).

Second, within individual repeat domains, many protein-binding motifs were enriched to extreme levels. Well over half of the motifs analyzed (101 out of 175) were in the 99^{th} percentile in terms of their enrichment relative to other mouse lncRNAs in at least one *Xist* or *Rsx* repeat domain, and all repeat domains in *Xist* and *Rsx* harbored multiple protein-binding motifs that were enriched at the 99^{th} percentile or greater (Figure 6A; Table S4). This extremity was notable considering that most *Xist* and *Rsx* repeats are greater in length than the average mouse lncRNA (Figure 2A). For example, at 13kb in length, *Rsx* Repeat 1 is longer than 99.97% of spliced mouse lncRNAs (Figure 2A). Nevertheless, on a length-normalized basis, multiple protein-binding motifs were enriched in Repeat 1 at the 99^{th} percentile, in both koala and opossum *Rsx*. Inasmuch as motif density is known to be an important driver of associations between proteins and RNA [20, 30, 34], our data suggest that at the level of sequence composition, the repeat domains in *Xist* and *Rsx* each have the potential to serve as high-affinity binding platforms for multiple proteins.

Lastly, many of the most strongly enriched motifs in both *Xist* and *Rsx* are known to recruit near-ubiquitous RNA-binding proteins that play core roles in the process of splicing [36]. These included PTBP1, RMB5, SF3B4, SNRPA, SNRPB2, U2AF2, multiple SR proteins, and multiple HNRNP proteins, including HNRNPA1, HNRNPC, and HNRNPK (Figure 6A; Table S4). We recognize that the motifs available for this analysis are biased towards RNA-binding proteins whose functions are best understood; overwhelmingly, these proteins are splicing factors [30, 31]. Nevertheless, it is possible that an extreme enrichment for a motif that recruits a ubiquitously expressed splicing factor may confer a function that a single binding motif would not. For example, we presume that the function of Repeat B could not be recapitulated by a single motif that binds HNRNPK [14].

3 Discussion

Xist has served as a paradigmatic regulatory lncRNA for more than 25 years [1–4]. Nevertheless, it has been challenging to apply the information gained from the study of *Xist* to other lncRNAs. This is because *Xist* has little linear sequence similarity to other RNAs, even to lncRNAs like *Rsx*, which seem likely to encode analogous functions [18, 37]. In the present study, we used a non-linear method of sequence comparison called SEEKR [20] to compare the repetitive

regions of *Xist* and *Rsx*. Our data provide sequence-based evidence to support the hypothesis that *Xist* and *Rsx* are functional analogues that arose through convergent evolution, and provide insights into mechanisms through which their repeat domains may encode function.

Unexpectedly, at the level of *k*-mers, the repeat domains of *Xist* and *Rsx* partitioned into two major clusters. *Xist* Repeats B, C, and D were highly similar to each other and to *Rsx* Repeat 1, whereas *Xist* Repeats A and E were most similar to each other and to *Rsx* Repeats 2, 3, and 4. From prior analyses of sequence content, there is little that would have suggested that the repeats in these two lncRNAs would cluster together in such a manner. However, prior molecular analyses of *Xist* are consistent with such a clustering [1–4].

Specifically, *Xist* Repeats B and C are known to play important roles in recruiting PRC1 to the inactive X through their ability to bind HNRNPK and possibly other proteins [14]. The similarity between Repeats B and C and *Xist* Repeat D and *Rsx* Repeat 1 suggests that the latter two repeats may also play roles in recruiting PRC1. Consistent with this possibility, we found that an antibody specific to HNRNPK robustly retrieved *Rsx* RNA in an IP. Moreover, eCLIP data show that *Xist* Repeat D is enriched for HNRNPK binding in human cells ([34]). Thus, even within *Xist*, murid and non-murid mammals may have convergently evolved separate repeats to recruit PRC1, in the form of Repeats C and D, respectively.

Relatedly, *Xist* Repeats A and E have been implicated in recruitment of PRC2 to the inactive X, both via direct and cooperative means [13, 15, 17, 38–42]. The similarity between *Xist* Repeats A and E and *Rsx* Repeats 2, 3, and 4 suggests that the *Rsx* repeats could also play roles in recruiting PRC2.

Based on these data, we propose that the two major clusters of repeats in *Xist* and *Rsx* function in part to cooperatively recruit PRC1 and PRC2 to chromatin. Within *Xist*, Repeat B plays a dominant role in recruiting PRC1 via its ability to bind HNRNPK; in turn, PRC1-induced chromatin modifications likely stimulate loading of PRC2 onto chromatin of the inactive X [13, 14]. Nevertheless, a PRC1-dominant model does not preclude other repeats in *Xist* or *Rsx* from functioning in PRC2 recruitment. Indeed, while there does not appear to be a single domain in *Xist* that is absolutely required to recruit PRC2 during the early stages of XCI [12, 40], it is possible that multiple domains in *Xist* recruit PRC2 duplicatively, such that deletion of any single domain alone does not cause complete loss in PRC2 recruitment. This hypothesis is supported by

prior studies that link both Repeat A and E to recruitment of PRC2 [13, 15, 17, 38–42], and by our own data that show *Xist* Repeats A and E and *Rsx* Repeats 2, 3, and 4 have similar k-mer profiles and motif contents. PRC1, PRC2, and related complexes function cooperatively in flies, mammals, and plants [43–45]. Considering this cooperativity, it is conceivable that the repeat domains in *Xist* and *Rsx* also cooperate to distribute PRC1 and PRC2 on chromatin.

Beyond recruiting PRCs, *Xist* evades nuclear export, it associates with transcribed regions of chromatin, and it induces Polycomb-independent gene silencing [1, 2, 4]. It is possible that *Rsx* carries out many, if not all of these actions, and that *Rsx* relies on sets of proteins similar to those employed by *Xist* to achieve them [18, 37]. We found that all *Xist* and *Rsx* repeat domains harbored extreme levels of enrichment for multiple motifs known to recruit different subsets of RNA-binding proteins. Most of these proteins have been best-characterized in the context of splicing, rather than epigenetic silencing.

In light of these data, we suggest that the repeat domains in *Xist* and *Rsx* may encode some of their functions not by recruiting a set of dedicated RNA silencing factors, but by engaging with ubiquitously-expressed RNA-binding proteins in ways that are distinct from most other RNAs. Such a model was recently proposed [2], and agrees well with what is known about the specificity of RNA-protein interactions. Most RNA-binding proteins have limited sequence specificity, and are capable of binding many thousands of regions in hundreds to thousands of expressed RNAs [30, 31, 34]. SPEN and HNRNPK are two RNA-binding proteins that are critical for *Xist*-induced silencing, yet they clearly associate with RNAs other than *Xist* [33, 34]. Relatedly, many other proteins important for *Xist*-induced silencing play central roles in RNA splicing and nuclear export and, through these latter roles, likely associate with a large portion of the transcriptome [9]. Thus, *Xist* and *Rsx* may distinguish themselves from other chromatin-associated transcripts not necessarily by the proteins to which they bind, but by the manner in which they bind these proteins.

That the related repeat domains were present in a different order in *Xist* and *Rsx* supports the notion that within a lncRNA, the order of functional domains is likely to be less important than the presence of the functional domains (Figure 6B). This notion is consistent with a body of work that suggests lncRNAs encode regulatory function in a modular fashion, via discrete domains that recruit distinct subsets of effector proteins [10, 12, 14, 16, 20, 46–54].

From a methodological standpoint, our manuscript outlines approaches that should prove useful in the study of functional domains in other sets of RNAs. Intuitively, k-mer based comparisons like SEEKR seem most likely to succeed in identifying similarity when the domains of interest are repetitive. By nature, repetitive domains that share enrichments of similar subsets of k-mers will be more similar to each other than they will be similar to the average non-repetitive region in the transcriptome.

Nevertheless, similarity between two repetitive domains, when observed, should be carefully considered, especially when the similarity occurs in lncRNAs such as *Xist* and *Rsx*, which are expressed at similar levels in equivalent subcellular compartments [18, 37]. Motif density is known to be a dominant factor driving protein/RNA interactions [20, 30, 37]. All other variables being equal, two lncRNAs that harbor domain-specific k-mer similarity should possess similar protein-binding profiles that could specify similar or analogous function.

However, SEEKR is not limited to analysis of repetitive domains. It also has the ability to detect similarity between repetitive and non-repetitive domains and between strictly non-repetitive domains as well. In any given sequence, a set of k-mers can be arranged in repetitive or non-repetitive ways, and SEEKR has no inherent preference for one over the other. As a contrived example, the sequence of *Xist* Repeat D can be shuffled in a way that eliminates its repeated monomers, yet entirely preserves its k-mer content (File S1). By BLAST, this shuffled sequence has little internal similarity to itself or to Repeat D (Figure S6). Yet, by k-mer content, the shuffled sequence and Repeat D are literally identical (Figure S6). In a real-world example, the top five lncRNAs that SEEKR found to be the most similar to Repeat D are not nearly as repetitive as Repeat D itself (Figure S6). Of all *Xist* and *Rsx* repeats, Repeat D is the most complex (Figure 2B). Nevertheless, these results demonstrate that k-mer based similarity searches performed with repetitive domains can identify non-repetitive top hits.

With regard to non-repetitive domains, our 2018 study showed that SEEKR rivaled BLAST-like alignment in its ability to detect lncRNA homologues in human and mouse [20]. The majority of homologues detected by SEEKR either lacked obvious repetitive elements, or were predominantly comprised of non-repetitive sequence; the lncRNAs H19, Hottip, Malat1, Miat, and RMST being specific examples. We also found that SEEKR could identify *Xist*-like repressive activity in several synthetic and natural lncRNAs that lacked repetitive elements [20]. Thus, even

in non-repetitive regions of RNA, SEEKR should be capable of detecting meaningful similarities. However, functional domains comprised of high-complexity sequence elements will likely remain challenging to identify, regardless of the method in use.

Key variables to decide upon when using SEEKR are the k-mer length and the appropriate set of RNAs that define the background k-mer frequency; i.e. the set of RNAs used to define the means and standard deviations from which k-mer z-scores are calculated. At present, data regarding functional domains in lncRNAs are too limited to arrive at conclusive recommendations for either variable. We favor using a k-mer length at which 4k most closely resembles the length of the shortest domain being analyzed. This approach minimizes the number of k-mers that yield counts of zero in the domain. Data from the present study as well as our prior work suggest that this minimization increases discriminatory power ([20]).

In terms of the set of RNAs that should be used to define the background k-mer frequency, it is worth noting that SEEKR measures relative, not absolute, similarity. Pearson's r values returned by SEEKR reflect the similarity between two sequences relative to the k-mer frequency present in the background set of RNAs. We have found that using a background set of all lncRNAs in a genome provides a convenient way to identify trends. For example, in the present study, we used all known spliced lncRNAs in the mouse as a background set. Accordingly, we were able to identify properties in the repeat domains of *Xist* and *Rsx* that were distinct from the average spliced lncRNA annotated by GENCODE [28].

In our initial description of SEEKR, we used k-mer contents of full-length lncRNAs as search features; we did not examine k-mer contents at the level of individual domains [20]. The domain-centric approaches outlined in the present study may be better suited for lncRNAs such as *Xist* and *Rsx*, which have multiple functions that are likely to be distributed amongst multiple domains. Indeed, at the level of k-mers, full-length *Xist* and *Rsx* were negatively correlated to each other. Similarities between the two lncRNAs emerged only when we took a domain-centric approach. Other eutherian lncRNAs known to harbor *Xist*-like silencing function, such as Kcnq1ot1 and Airn, are exceptionally long – each on the order of 90kb. Extrapolating from our findings above, we would expect these lncRNAs to harbor the greatest levels of similarity to each other not at the level of their full-length transcripts, but at the level of specific domains.

4 Methods

4.1 Long read sequencing of *Rsx*

High molecular weight DNA from VMRC18-839J22 and VMRC18-303M7 BACs was prepared using the NucleoBond BAC 100 kit (Machery Nagel). DNA from the two BAC preparations was pooled, sheared to an average length of 20kb using a g-TUBE (Covaris), and then sequenced on the Oxford Nanopore Technologies (ONT) MinION using an R9.4 flow cell (FLO-MIN106) following the 1D ligation protocol (SQ*k*-LSK109).

Reads were base-called with Albacore 2.3.1 (ONT) then assembled using Flye 2.3.5b [55]. The six resulting scaffolds were aligned to E. coli K12 (NC000913.3), opossum chromosome X (MonDom5, NC008809.1) and the pCC1BAC cloning vector (EU140750.1). Scaffolds consisting entirely of E. coli or cloning vector DNA were removed. Three scaffolds aligned to adjacent regions of the MonDom5 X chromosome. These were merged together into a single candidate assembly sequence that was then polished iteratively with Racon 1.3.2 four times [56], followed by Nanopolish 0.10.1 [57], to produce a final complete assembly of 235,139 nucleotides.

This polished assembly sequence was aligned again to MonDom5 using BLASTN to establish start and end coordinates to use as a reference when replacing the gaps in MonDom5 with the completed sequence in our assembly. The final sequence of opossum *Rsx* used in this work was generated using splice annotations from [18], and replacing the N's in mondom5 with the corresponding sequence from our polished assembly (nucleotide substitutions are listed in Table S3). Raw sequencing reads were deposited in NCBI's SRA, under accession number PRJNA522427.

4.2 Nanopore sequencing and Rsx splice structure determination

High molecular weight DNA from VMRC18-839J22 and VMRC18-303M7 BACs was prepared using the NucleoBond BAC 100 kit (Machery Nagel). DNA from the two BAC preparations was pooled, sheared to an average length of 20kb using a g-TUBE (Covaris), and then sequenced on the Oxford Nanopore Technologies (ONT) MinION using an R9.4 flow cell (FLO-MIN106) following the 1D ligation protocol (SQ*k*-LSK109). Reads were base-called with Albacore 2.3.1 (ONT) then assembled using Flye 2.3.5b [55]. The six resulting scaffolds were aligned to E. coli K12 (NC000913.3), opossum chromosome X (MonDom5, NC008809.1) and the pCC1BAC cloning vector (EU140750.1). Scaffolds consisting entirely of E. coli or cloning vector DNA were removed.

Three scaffolds aligned to adjacent regions of the MonDom5 X chromosome. These were merged together into a single candidate assembly sequence that was then polished iteratively with Racon 1.3.2 four times [56], followed by Nanopolish 0.10.1 [57], to produce a final complete assembly of 235,139 nucleotides.

This polished assembly sequence was aligned again to MonDom5 using BLASTN to establish start and end coordinates to use as a reference when replacing the gaps in MonDom5 with the completed sequence in our assembly. The final sequence of opossum *Rsx* used in this work was generated using splice annotations from [18], and replacing the N's in mondom5 with the corresponding sequence from our polished assembly. Raw sequencing reads were deposited in NCBI's SRA, under accession number PRJNA522427.

4.3 Defining tandem repeat domains of Xist/Rsx

The sequence of all *Xist* and *Rsx* repeat domains used in this work can be found in File S1. The sequences of all full-length *Xist* and *Rsx* lncRNAs used in this work can be found in File S2. The spliced mouse *Xist* sequence was sourced from the mm10 build of the mouse genome and annotations for the tandem repeats were sourced from [22]. The spliced human *Xist* sequence was sourced from the hg38 build of the human genome and the annotations for the tandem repeats were sourced from [24, 25].

The sequences of spliced *Xist* used to generate the dot plots in Figure S1 were obtained directly from annotations in the UCSC genome browser, or, for genomes in which full annotations were unavailable, were reconstructed from partial annotations by UCSC and RNA-seq data from [47]. In the case of the vole Microtus rossiaemeridionalis, *Xist* sequence was obtained directly from [23].

Spliced koala *Rsx* was obtained from [19]. To identify repeat domains, *Rsx* was aligned to itself using EMBOSS dotmatcher with a 10bp window and a 40 threshold [21]. Starts and stop positions of each repeat were defined by visual inspection of the dot plot. We considered separating the fourth major repeat in *Rsx* into two repeat domains, one 500bp and the other 5000bp in length (see Figure 1D); however, analysis of the shorter sub-repeat within Repeat 4 revealed its k-mer content to be highly similar to the larger sub-repeat (not shown). Thus, to simplify our analyses and to clarify our presentation, we elected to merge the sub-repeats. Repeat domains

in opossum *Rsx* (after filling in the gaps in assembly; see below) were defined in the identical manner.

4.4 *k*-mer correlations

SEEKR was performed essentially as described in [20], with minor modifications. As a reference for normalization, we first calculated the mean and standard deviation for all *k*-mers at $k = 4$ in the GENCODE M18 lncRNA annotation file. We then generated length normalized counts of all *k*-mers at $k = 4$ for each repeat domain in *Xist* and *Rsx* and calculated *z*-scores for each *k*-mer by subtracting the mean and dividing by the standard deviation for each *k*-mer from our reference set of GENCODE lncRNAs. Prior to performing Pearson's correlation, *z*-scores were $\log_2$ transformed.

To generate the distributions of Pearson's values in Figure 2B and Figure 5I, we calculated the *k*-mer profile for each repeat domain and each GENCODE M18 lncRNA using the mean and standard deviation values from the full-length GENCODE M18 lncRNA annotation file, as described above. We then $\log_2$-transformed the *z*-scores and used Pearson's correlation to compare all lncRNAs to the *Xist* repeat in question.

4.5 Motif enrichment algorithm

To weight the sums of *z*-scores by the HNRNPK PWM in Figure 4E we performed the following calculation. For all *k*-mers at k = 5 we calculated the probability of a given *k*-mer's sequence occurring in the PWM for HNRNPK. The probability was defined as the independent probability of each letter in the *k*-mer occurring at the corresponding location within the PWM for each possible frame within the PWM. The HNRNPK motif is 8nt long, therefore there were 3 possible frames for a 5-mer to fall within. The *z*-score for the *k*-mer in question was then weighted by taking the sum of the product between the *z*-score and each probability. The height of the bars in Figure 4E represent the sum of weighted *z*-scores for each *Xist* and *Rsx* repeat domain. The set of mouse lncRNAs from GENCODE M18 was used to derive *z*-scores that described the length normalized abundance of each *k*-mer in each repeat domain.

4.6 *De novo* motif analysis

Motifs in each *Xist* and *Rsx* repeat domain were detected with MEME (version 5.0.2; [29]), run using the following options: -mod anr -dna -bfile bkg.meme -nmotifs 100 -minw 4 -maxw 12

-maxsites 1000, where the "bkg.meme" file specified a background frequency of 0.25 for all four nucleotides.

4.7 Consecutive k-mer analysis

To calculate the sums of z-scores for k-mers containing matches to mononucleotide runs in Figures 4A-D, we used the following approach. A mononucleotide run was defined as at least two consecutive occurrences of the nucleotide in question. For each nucleotide [A—C—G—T], we multiplied the z-score for each k-mer that contained a run by (the nucleotide length of the run minus 1). The sum of these products for each repeat domain at k-mer length k = 5 is plotted in Figures 4A-D. Identical trends were seen using k-mer lengths $k = 4$, 5, and 6 (Figure S3). k-mer length k = 5 was chosen for plotting in Figure 4 to emphasize trends that were present but less pronounced when using k-mer length $k = 4$. The set of mouse lncRNAs from GENCODE M18 was used to derive z-scores that described the length normalized abundance of each k-mer in each repeat domain.

4.8 Detecting HNRNPK-binding motif matches

Motifs occurrences in each *Xist* and *Rsx* repeat domain were detected with FIMO (version 5.0.2; [29]), run using the following non-default option: –thresh 0.01.

4.9 RNA Immunoprecipitation

Cultured female M. domestica fibroblast cells were harvested at 70% confluency by scraping, then aliquoted into 1 107 cells, pelleted by centrifugation at 200g, then snap-frozen and stored at -80C until used. RIPs from non-crosslinked cells were performed essentially as described in [42], using the following antibodies from Abcam: H3K27me3 (ab6002), CTCF (ab70303), HNRNPK (ab39975), and mouse IgG (ab18413). Briefly, cell pellets were gently resuspended in 1 mL of ice-cold RIPA buffer supplemented with 1 EDTA-free Proteinase Inhibitor Cocktail (Thermo Scientific) and lysed for 15 min at 4C. Samples were sonicated at 4C (Qsonica Q700 with cup horn accessory) at 12% amplitude for fifteen 30 second intervals, with 30 second resting steps between intervals. Cell debris was removed by centrifugation (at 6000 g for 5 minutes), and samples were subsequently diluted to 1mg of protein per ml with ice-cold RIPA buffer. Lysates with 1mg of total protein (i.e. 500ul) were incubated with the appropriate antibody coupled to Protein G beads (Life Technologies), overnight at 4 C with end-over-end rotation. Beads with no antibodies (mock

IP) were used as background control. Beads were removed from lysate using a magnetic stand and were re-suspended in 1ml of ice cold NP-40 buffer (50 mM Tris at pH 7.5, 50 mM NaCl, 10 mM EDTA, 1% Nonidet P-40, 0.5% sodium deoxycholate, 0.1% SDS) and washed for 15min at 4 C with end-over-end rotation, repeated twice, followed by three washes with RIPA buffer. Following the last wash, beads were collected and re-suspended in 1ml of Trizol (Life Technologies) for RNA extraction. 10% of the input lysate (i.e. 50ul) was processed in parallel. RNA was cleaned using RNeasy spin columns (Qiagen), following the manufacturer's "RNA Cleanup" protocol, with on-column RNase-free DNase Set (Qiagen) treatment. cDNA was synthesized using input and immunoprecipitated RNA with SuperScript III reverse transcriptase (Life Technologies) and random hexamer priming. *Rsx* was detected by RT-qPCR (in technical triplicate) with primer pair L2 from (Grant et al., 2012). Cycle threshold (C_t) values were normalized to input and relative to the IgG. Fold enrichment was determined by relative quantification, which was calculated using the $2^e(\Delta\Delta C_t)$ method. The level of Gapdh mRNA enrichment was used as an internal non-target index in the qPCR analysis.

REFERENCES

[1] B. P. Balaton, T. Dixon-McDougall, S. B. Peeters, and C. J. Brown, *The eXceptional nature of the X chromosome*, 2018.

[2] N. Brockdorff, "Local tandem repeat expansion in Xist RNA as a model for the functionalisation of ncRNA," *Non-coding RNA*, 2018, ISSN: 2311553X.

[3] S. T. Da Rocha and E. Heard, *Novel players in X inactivation: Insights into Xist-mediated gene silencing and chromosome conformation*, 2017.

[4] A. Sahakyan, Y. Yang, and K. Plath, *The Role of Xist in X-Chromosome Dosage Compensation*, 2018.

[5] C. Chu, Q. C. Zhang, S. T. Da Rocha, R. A. Flynn, M. Bharadwaj, J. M. Calabrese, T. Magnuson, E. Heard, and H. Y. Chang, "Systematic discovery of Xist RNA binding proteins," *Cell*, 2015, ISSN: 10974172.

[6] J. M. Engreitz, A. Pandya-Jones, P. McDonel, A. Shishkin, K. Sirokman, C. Surka, S. Kadri, J. Xing, A. Goren, E. S. Lander, K. Plath, and M. Guttman, "The Xist lncRNA exploits three-dimensional genome architecture to spread across the X chromosome," *Science*, 2013, ISSN: 10959203.

[7] Y. Hoki, N. Kimura, M. Kanbayashi, Y. Amakawa, T. Ohhata, H. Sasaki, and T. Sado, "A proximal conserved repeat in the Xist gene is essential as a genomic element for X-inactivation in mouse," *Development*, 2009, ISSN: 09501991.

[8] C. A. McHugh, C. K. Chen, A. Chow, C. F. Surka, C. Tran, P. McDonel, A. Pandya-Jones, M. Blanco, C. Burghard, A. Moradian, M. J. Sweredoski, A. A. Shishkin, J. Su, E. S. Lander, S. Hess, K. Plath, and M. Guttman, "The Xist lncRNA interacts directly with SHARP to silence transcription through HDAC3," *Nature*, 2015, ISSN: 14764687.

[9] B. Moindrot, A. Cerase, H. Coker, O. Masui, A. Grijzenhout, G. Pintacuda, L. Schermelleh, T. B. Nesterova, and N. Brockdorff, "A Pooled shRNA Screen Identifies Rbm15, Spen, and Wtap as Factors Required for Xist RNA-Mediated Silencing," *Cell Reports*, 2015, ISSN: 22111247.

[10] D. P. Patil, C. K. Chen, B. F. Pickering, A. Chow, C. Jackson, M. Guttman, and S. R. Jaffrey, "M6 A RNA methylation promotes XIST-mediated transcriptional repression," *Nature*, 2016, ISSN: 14764687.

[11] M. E. Royce-Tolland, A. A. Andersen, H. R. Koyfman, D. J. Talbot, A. Wutz, I. D. Tonks, G. F. Kay, and B. Panning, "The A-repeat links ASF/SF2-dependent Xist RNA processing with random choice during X inactivation," *Nature Structural and Molecular Biology*, 2010, ISSN: 15459993.

[12] A. Wutz, T. P. Rasmussen, and R. Jaenisch, "Chromosomal silencing and localization are mediated by different domains of Xist RNA," *Nature Genetics*, 2002, ISSN: 10614036.

[13] M. Almeida, G. Pintacuda, O. Masui, Y. Koseki, M. Gdula, A. Cerase, D. Brown, A. Mould, C. Innocent, M. Nakayama, L. Schermelleh, T. B. Nesterova, H. Koseki, and N. Brockdorff,

"PCGF3/5-PRC1 initiates Polycomb recruitment in X chromosome inactivation," *Science*, 2017, ISSN: 10959203.

[14] G. Pintacuda, G. Wei, C. Roustan, B. A. Kirmizitas, N. Solcan, A. Cerase, A. Castello, S. Mohammed, B. Moindrot, T. B. Nesterova, and N. Brockdorff, "hnRNPK Recruits PCGF3/5-PRC1 to the Xist RNA B-Repeat to Establish Polycomb-Mediated Chromosomal Silencing," *Molecular Cell*, 2017, ISSN: 10974164.

[15] R. Ridings-Figueroa, E. R. Stewart, T. B. Nesterova, H. Coker, G. Pintacuda, J. Godwin, R. Wilson, A. Haslam, F. Lilley, R. Ruigrok, S. A. Bageghni, G. Albadrani, W. Mansfield, J. A. Roulson, N. Brockdorff, J. F. Ainscough, and D. Coverley, "The nuclear matrix protein CIZ1 facilitates localization of Xist RNA to the inactive X-chromosome territory," *Genes and Development*, 2017, ISSN: 15495477.

[16] M. J. Smola, T. W. Christy, K. Inoue, C. O. Nicholson, M. Friedersdorf, J. D. Keene, D. M. Lee, J. M. Calabrese, and K. M. Weeks, "SHAPE reveals transcript-wide interactions, complex structural domains, and protein interactions across the Xist lncRNA in living cells," *Proceedings of the National Academy of Sciences of the United States of America*, 2016, ISSN: 10916490.

[17] H. Sunwoo, D. Colognori, J. E. Froberg, Y. Jeon, and J. T. Lee, "Repeat E anchors Xist RNA to the inactive X chromosomal compartment through CDKN1A-interacting protein (CIZ1)," *Proceedings of the National Academy of Sciences of the United States of America*, 2017, ISSN: 10916490.

[18] J. Grant, S. K. Mahadevaiah, P. Khil, M. N. Sangrithi, H. Royo, J. Duckworth, J. R. McCarrey, J. L. Vandeberg, M. B. Renfree, W. Taylor, G. Elgar, R. D. Camerini-Otero, M. J. Gilchrist, and J. M. Turner, "Rsx is a metatherian RNA with Xist-like properties in X-chromosome inactivation," *Nature*, 2012, ISSN: 00280836.

[19] R. N. Johnson, D. O'Meally, Z. Chen, G. J. Etherington, S. Y. Ho, W. J. Nash, C. E. Grueber, Y. Cheng, C. M. Whittington, S. Dennison, E. Peel, W. Haerty, R. J. O'Neill, D. Colgan, T. L. Russell, D. E. Alquezar-Planas, V. Attenbrow, J. G. Bragg, P. A. Brandies, A. Y. Y. Chong, J. E. Deakin, F. Di Palma, Z. Duda, M. D. Eldridge, K. M. Ewart, C. J. Hogg, G. J. Frankham, A. Georges, A. K. Gillett, M. Govendir, A. D. Greenwood, T. Hayakawa, K. M. Helgen, M. Hobbs, C. E. Holleley, T. N. Heider, E. A. Jones, A. King, D. Madden, J. A. Graves, K. M. Morris, L. E. Neaves, H. R. Patel, A. Polkinghorne, M. B. Renfree, C. Robin, R. Salinas, K. Tsangaras, P. D. Waters, S. A. Waters, B. Wright, M. R. Wilkins, P. Timms, and K. Belov, "Adaptation and conservation insights from the koala genome," *Nature Genetics*, 2018, ISSN: 15461718.

[20] J. M. Kirk, S. O. Kim, K. Inoue, M. J. Smola, D. M. Lee, M. D. Schertzer, J. S. Wooten, A. R. Baker, D. Sprague, D. W. Collins, C. R. Horning, S. Wang, Q. Chen, K. M. Weeks, P. J. Mucha, and J. M. Calabrese, "Functional classification of long non-coding RNAs by k-mer content," *Nature Genetics*, 2018, ISSN: 15461718.

[21] P. Rice, L. Longden, and A. Bleasby, *EMBOSS: The European Molecular Biology Open Software Suite*, 2000.

[22] N. Brockdorff, "The product of the mouse Xist gene is a 15 kb inactive X-specific transcript containing no conserved ORF and located in the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 515–526, 1992 10, ISSN: 0092-8674.

[23] T. B. Nesterova, S. Y. Slobodyanyuk, E. A. Elisaphenko, A. I. Shevchenko, C. Johnston, M. E. Pavlova, I. B. Rogozin, N. N. Kolesnikov, N. Brockdorff, and S. M. Zakian, *Characterization of the genomic Xist locus in rodents reveals conservation of overall gene structure and tandem repeats but rapid evolution of unique sequence*, 2001.

[24] Z. C. Yen, I. M. Meyer, S. Karalic, and C. J. Brown, "A cross-species comparison of X-chromosome inactivation in Eutheria," *Genomics*, 2007, ISSN: 08887543.

[25] C. J. Brown, "The human XIST gene: analysis of a 17 kb inactive X-specific RNA that contains conserved repeats and is highly localized within the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 527–542, 1992 10, ISSN: 0092-8674.

[26] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, 1990, ISSN: 00222836.

[27] T. J. Wheeler and S. R. Eddy, "Nhmmer: DNA homology search with profile HMMs," *Bioinformatics*, 2013, ISSN: 14602059.

[28] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles, J. Lagarde, L. Veeravalli, X. Ruan, Y. Ruan, T. Lassmann, P. Carninci, J. B. Brown, L. Lipovich, J. M. Gonzalez, M. Thomas, C. A. Davis, R. Shiekhattar, T. R. Gingeras, T. J. Hubbard, C. Notredame, J. Harrow, and R. Guigó, "The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression," *Genome Research*, 2012, ISSN: 10889051.

[29] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble, "MEME Suite: Tools for motif discovery and searching," *Nucleic Acids Research*, 2009, ISSN: 03051048.

[30] D. Dominguez, P. Freese, M. S. Alexis, A. Su, M. Hochman, T. Palden, C. Bazile, N. J. Lambert, E. L. Van Nostrand, G. A. Pratt, G. W. Yeo, B. R. Graveley, and C. B. Burge, "Sequence, Structure, and Context Preferences of Human RNA Binding Proteins," *Molecular Cell*, 2018, ISSN: 10974164.

[31] D. Ray, H. Kazan, K. B. Cook, M. T. Weirauch, H. S. Najafabadi, X. Li, S. Gueroussov, M. Albu, H. Zheng, A. Yang, H. Na, M. Irimia, L. H. Matzat, R. K. Dale, S. A. Smith, C. A. Yarosh, S. M. Kelly, B. Nabet, D. Mecenas, W. Li, R. S. Laishram, M. Qiao, H. D. Lipshitz, F. Piano, A. H. Corbett, R. P. Carstens, B. J. Frey, R. A. Anderson, K. W. Lynch, L. O. Penalva, E. P. Lei, A. G. Fraser, B. J. Blencowe, Q. D. Morris, and T. R. Hughes, "A compendium of RNA-binding motifs for decoding gene regulation," *Nature*, 2013, ISSN: 00280836.

[32] J. T. Kung, B. Kesner, J. Y. An, J. Y. Ahn, C. Cifuentes-Rojas, D. Colognori, Y. Jeon, A. Szanto, B. C. delRosario, S. F. Pinter, J. A. Erwin, and J. T. Lee, "Locus-specific targeting to the X chromosome revealed by the RNA interactome of CTCF," *Molecular Cell*, 2015, ISSN: 10974164.

[33] D. Cirillo, M. Blanco, A. Armaos, A. Buness, P. Avner, M. Guttman, A. Cerase, and G. G. Tartaglia, *Quantitative predictions of protein interactions with long noncoding RNAs: To the Editor*, 2016.

[34] E. L. Van Nostrand, G. A. Pratt, A. A. Shishkin, C. Gelboin-Burkhart, M. Y. Fang, B. Sundararaman, S. M. Blue, T. B. Nguyen, C. Surka, K. Elkins, R. Stanton, F. Rigo, M. Guttman, and G. W. Yeo, "Robust transcriptome-wide discovery of RNA-binding protein binding sites with enhanced CLIP (eCLIP)," *Nature Methods*, 2016, ISSN: 15487105.

[35] S. Kumar, G. Stecher, M. Suleski, and S. B. Hedges, "TimeTree: A Resource for Timelines, Timetrees, and Divergence Times," *Molecular biology and evolution*, 2017, ISSN: 15371719.

[36] M. C. Wahl and R. Lührmann, *Snapshot: Spliceosome dynamics I*, 2015.

[37] X. Wang, K. C. Douglas, J. L. VandeBerg, A. G. Clark, and P. B. Samollow, "Chromosome-wide profiling of X-chromosome inactivation and epigenetic states in fetal brain and placenta of the opossum, Monodelphis domestica," *Genome Research*, 2014, ISSN: 10889051.

[38] C. Cifuentes-Rojas, A. J. Hernandez, K. Sarma, and J. T. Lee, "Regulatory Interactions between RNA and Polycomb Repressive Complex 2," *Molecular Cell*, 2014, ISSN: 10974164.

[39] C. Davidovich, X. Wang, C. Cifuentes-Rojas, K. J. Goodrich, A. R. Gooding, J. T. Lee, and T. R. Cech, "Toward a consensus on the binding specificity and promiscuity of PRC2 for RNA," *Molecular Cell*, 2015, ISSN: 10974164.

[40] A. Kohlmaier, F. Savarese, M. Lachner, J. Martens, T. Jenuwein, and A. Wutz, "A chromosomal memory triggered by Xist regulates histone methylation in X inactivation," *PLoS Biology*, 2004, ISSN: 15449173.

[41] X. Wang, K. J. Goodrich, A. R. Gooding, H. Naeem, S. Archer, R. D. Paucek, D. T. Youmans, T. R. Cech, and C. Davidovich, "Targeting of Polycomb Repressive Complex 2 to RNA by Short Repeats of Consecutive Guanines," *Molecular Cell*, 2017, ISSN: 10974164.

[42] J. Zhao, B. K. Sun, J. A. Erwin, J. J. Song, and J. T. Lee, "Polycomb proteins targeted by a short repeat RNA to the mouse X chromosome," *Science*, 2008, ISSN: 00368075.

[43] N. P. Blackledge, N. R. Rose, and R. J. Klose, "Targeting Polycomb systems to regulate gene expression: Modifications to a complex story," *Nature Reviews Molecular Cell Biology*, 2015, ISSN: 14710080.

[44] Z. Li, X. Fu, Y. Wang, R. Liu, and Y. He, "Polycomb-mediated gene silencing by the BAH–EMF1 complex in plants," *Nature Genetics*, 2018, ISSN: 15461718.

[45] B. Schuettengruber, N. Oded Elkayam, T. Sexton, M. Entrevan, S. Stern, A. Thomas, E. Yaffe, H. Parrinello, A. Tanay, and G. Cavalli, "Cooperativity, specificity, and evolutionary stability of polycomb targeting in Drosophila," *Cell Reports*, 2014, ISSN: 22111247.

[46] E. Hacisuleyman, C. J. Shukla, C. L. Weiner, and J. L. Rinn, "Function and evolution of local repeats in the Firre locus," *Nature Communications*, 2016, ISSN: 20411723.

[47] H. Hezroni, D. Koppstein, M. G. Schwartz, A. Avrutin, D. P. Bartel, and I. Ulitsky, "Principles of Long Noncoding RNA Evolution Derived from Direct Comparison of Transcriptomes in 17 Species," *Cell Reports*, 2015, ISSN: 22111247.

[48] R. Johnson and R. Guigó, "The RIDL hypothesis: Transposable elements as functional domains of long noncoding RNAs," *RNA*, 2014, ISSN: 14699001.

[49] D. R. Kelley, D. G. Hendrickson, D. Tenen, and J. L. Rinn, "Transposable elements modulate human RNA abundance and splicing via specific RNA-protein interactions," *Genome Biology*, 2014, ISSN: 1474760X.

[50] F. Liu, S. Somarowthu, and A. M. Pyle, "Visualizing the secondary and tertiary architectural domains of lncRNA RepA," *Nature Chemical Biology*, 2017, ISSN: 15524469.

[51] Z. Lu, Q. C. Zhang, B. Lee, R. A. Flynn, M. A. Smith, J. T. Robinson, C. Davidovich, A. R. Gooding, K. J. Goodrich, J. S. Mattick, J. P. Mesirov, T. R. Cech, and H. Y. Chang, "RNA Duplex Map in Living Cells Reveals Higher-Order Transcriptome Structure," *Cell*, 2016, ISSN: 10974172.

[52] Y. Lubelsky and I. Ulitsky, "Sequences enriched in Alu repeats drive nuclear localization of long RNAs in human cells," *Nature*, 2018, ISSN: 14764687.

[53] S. Somarowthu, M. Legiewicz, I. Chillón, M. Marcia, F. Liu, and A. M. Pyle, "HOTAIR Forms an Intricate and Modular Secondary Structure," *Molecular Cell*, 2015, ISSN: 10974164.

[54] M. C. Tsai, O. Manor, Y. Wan, N. Mosammaparast, J. K. Wang, F. Lan, Y. Shi, E. Segal, and H. Y. Chang, "Long noncoding RNA as modular scaffold of histone modification complexes," *Science*, 2010, ISSN: 00368075.

[55] M. Kolmogorov, J. Yuan, Y. Lin, and P. A. Pevzner, "Assembly of long, error-prone reads using repeat graphs," *Nature Biotechnology*, 2019, ISSN: 15461696.

[56] R. Vaser, I. Sović, N. Nagarajan, and M. Šikić, "Fast and accurate de novo genome assembly from long uncorrected reads," *Genome Research*, 2017, ISSN: 15495469.

[57] N. J. Loman, J. Quick, and J. T. Simpson, "A complete bacterial genome assembled de novo using only nanopore sequencing data," *Nature Methods*, 2015, ISSN: 15487105.

CHAPTER 3: HMMSEEKR

1 Introduction

XIST and *Rsx* provide compelling examples of how the sequence-to-function relationship in lncRNAs may be modular in nature [1–5]. These two transcripts demonstrate clear sequence regions, defined by the boundaries of their tandem repeat domains [2, 6–8], that are conserved and are clearly functional as demonstrated by the binding of RNA binding proteins to their transcripts [6, 7, 9–13]. Our model of lncRNA function, specifically the bag-of-words model that the order of short motifs in a sequence is less important than their overall density [14], implicitly means that a tandem repeated sequence may not be essential for recruitment of these RNA binding proteins [2].

It is possible to show that the k-mer content of a sequence can be perfectly preserved while destroying the tandomly repeated nature of a sequence (Fig 3.1 A-C). This can be done by building a graph whose nodes are $(k-1)$-mers and whose edges are connected such that a sequence is reconstructed with the exact same k-mer frequencies as the original sequence [15]. Furthermore, if we examine lncRNA sequences in mouse with very high SEEKR correlation to HXD it is clear that not all these sequences contain substantial tandemly repeated sequence as in *XIST* (Fig 3.1 D-H).

Therefore, a challenge in understanding lncRNA function is identifying where functional modules may be located within a sequence [16–18]. Clearly, the answer is not solely exonic sequence as in a protein coding reading frame. Within *XIST* these sequence regions are sub-sequences within the spliced transcript [1–3, 6, 7, 9, 11]. Compounding the issue is that as far as is known, no specific boundary motifs (*e.g.*, splice junction boundaries) are required. Potential splice junctions in genes are relatively easy to identify based off a combination of sequence content, and the presence of conserved 5' and 3' splice site motifs that demarcate the beginning and end of an intron, respectively [19].

To model this hypothesized sequence structure, we have developed an HMM that is designed

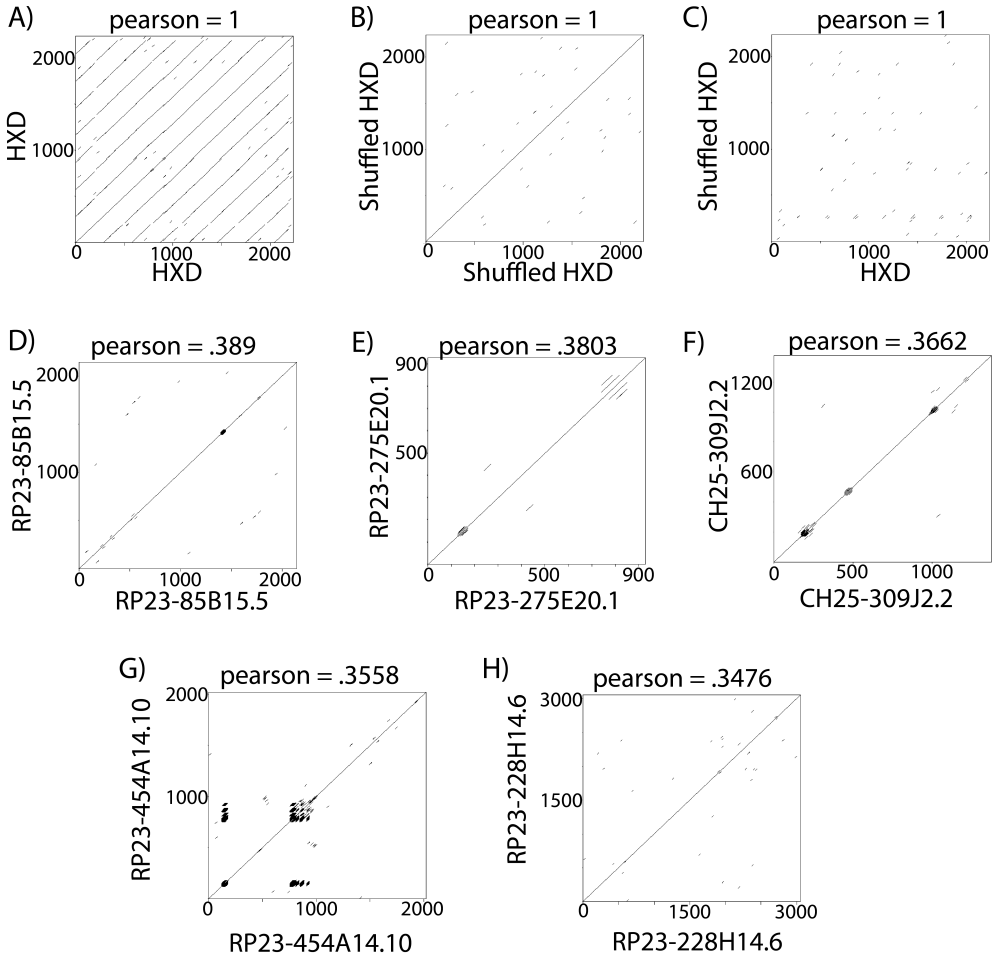

Figure 3.1: Dot plot alignments and SEEKR-defined similarity between human Xist Repeat D, shuffled Repeat D, and the top 5 mouse lncRNAs that are most similar to Repeat D. Repeat D was shuffled using μShuffle and preserving k-mer content at $k = 4$ (Jiang et al., 2008). Dotplots were generated using a window size 20 nucleotides and a threshold of 50% identity.

to identify sub-sequences within a larger transcript that contain regions of elevated k-mer content to a known functional domain. Given how little is known in the field about the sequence-to-function relationship in lncRNAs, we use the archetypal lncRNA *XIST* and its known functional sub-sequences (A-,B-,D-,E- repeats) as our model.

We demonstrate here the first computational tool capable of detecting sequence similarity between functionally related but non-homologous long non-coding RNAs in the human genome. Rather than deriving functional relationships through homology as in traditional sequence alignment [20, 21], we consider and model the problem of functional analogy between non-coding transcripts. lncRNAs are free to mutate at significantly faster rates than mRNAs however that

lack of conservation has been shown to not necessarily imply lack of function [2, 12, 22]. The model developed here is capable of identifying sequence features that are much too subtle and jumbled for even the most sensitive alignment algorithms to detect [20, 21, 23], yet these same features have been shown to strongly predictive of function [2, 14].

2 Results

2.1 Model Structure

Given a transcript sequence of k-mers, X, of length L, and a set of predefined functional features Y, the objective is to map each $x \in X$ to a functional state, $y \in Y$ through an HMM. Our underlying biological hypothesis is that non-coding RNAs contain sub-sequences of specific k-mer content that are enriched for motifs that bind a specific RBP or subset of RBPs and that these domains are free to start and stop within a larger sequence[1, 16, 17]. The primary purpose of the HMM is to identify the most likely positions within a sequence that these RBP-interacting functional domains begin and end. We therefore defined the first functional feature to be the *query*, which is a categorical distribution of k-mer frequencies from training sequences known to have some specific functional role (e.g. tandem repeats of *XIST*). We defined one additional functional feature which comprises a *null* feature. The null state is a categorical distribution of k-mer frequencies that represent the average of the transcriptome, or background sequence, which is designed to encapsulate all other sequences that don't match the *query*.

The two hidden states, *query* and *null* are represented by the symbols "+" and "-" respectively, throughout the rest of this chapter, as this is how they are coded within the python implementation of *hmmSEEKR*.

From a probabilistic point of view, the HMM models an RNA sequence by assuming the sequence we observe (X) is stochastically sampled from an unknown sequence of hidden states (functional domains) that control the behavior of X [24]. Prior HMM based models have primarily modeled the emission of a sequence of individual nucleotides and incorporated high-order interactions through conditional probabilities [19, 25, 26].

As an example, if the following sequence were observed, the probability of that sequence occurring could be calculated using varying amounts of prior information. Generally, the more relationships captured within the sequence, the better the sequence can be modeled (Table 3.1).

Observed Sequence $X = ATCGA$		
Markov Order	P(X)	Parameters
0	$P(A)P(T)P(C)P(G)P(A)$	4
1	$P(A)P(T\|A)P(C\|T)P(G\|C)P(A\|G)$	16
2	$P(A)P(T\|A)P(C\|AT)P(G\|TC)P(A\|CG)$	64
3	$P(A)P(T\|A)P(C\|AT)P(G\|ATC)P(A\|TCG)$	256

Table 3.1: Progressively more prior history can be incorporated to model the probability distribution of a sequence, however doing so exponentially increases the number of parameters to be estimated.

Given the underlying assumptions of SEEKR [14] we chose to model the emissions of k-mers directly rather than individual nucleotides conditioned on k previous nucleotides. When constructing a parse of a sequence, this choice better represents the bag-of-words model underyling SEEKR [14] while allowing each emission to be conditionally dependent only on the functional feature emitting it (Figure 3.2).

These two probabilities, *e.g.* $P(ATCGA)$ *vs.* $P(A|ATCG)$, are very similar to each other, and are related by a normalizing factor. *E.g.*, if $k = 5$, then the probability of observing the 5-mer $ATCGA$ is equivalent to the probability of observing A given the probability of the preceding nucleotides $ATCG$, multiplied by the probability of observing $ATCG$.

$$P(ATCGA) = P(A|ATCG)P(ATCG)$$

This example illustrates another reason that we chose to model the joint probability of a k-mer rather than the conditional probability of a nucleotide on the preceding k-mer – the joint probability distribution better models differentials between enriched and depleted k-mers in a bag-of-words model, whereas the conditional distribution better models longer-range interactions in the sequence. *E.g.*, the k-mer ATCG might be quite rare in the training data, but A preceded by ATCG might have a high probability so long as it is observed, and so within the context of the HMM this sparsity of ATCG would not get modeled as desired in SEEKR.

The last modeling choice we make is to construct the HMM such that each hidden state y has a non-zero probability of transitioning to any other state, including itself. This is formally known as an ergodic HMM [24]. Prior HMM models of DNA sequences are primarily left-right structured [19, 21, 25, 26], due to prior knowledge of the structure of a gene, or motif. If a model

for a gene were constructed, the transition Intron → 5' UTR can be assigned a probability of zero given prior knowledge. Within lncRNAs, we and others hypothesized and shown that protein binding functional domains are modular in nature and therefore the hidden states within our HMM are free to transition in all directions.

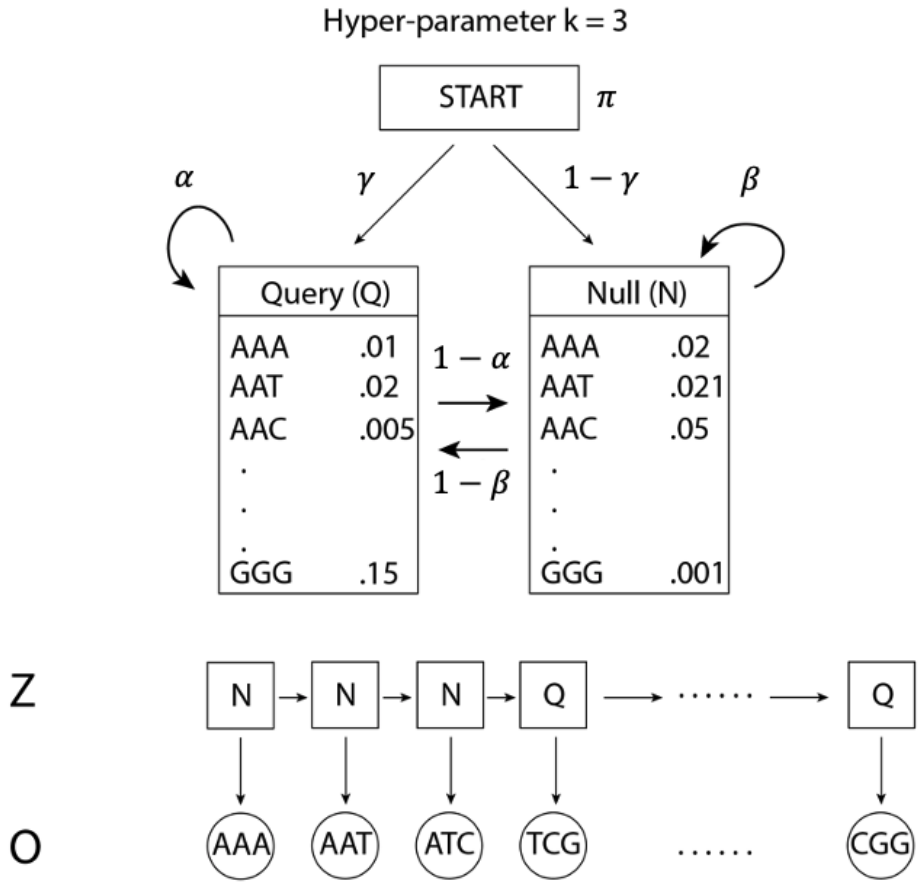

Figure 3.2: Graphical model of HMM SEEKR. The Query ("+") hidden state represents the functional domain to be identified in other lncRNA sequences and the Null ("-") hidden state represents the average k-mer frequencies of the transcriptome. The α and β parameters represent the self-transition probabilities for the hidden state.

2.2 Viterbi Parsing and Scoring

The primary use for the HMM in *hmmSEEKR* is identifying the most likely parse, ϕ_{max} for a given sequence or set of sequences, $X_i \in \{X_1, X_2, \ldots, X_n\}$, where each X_i is a sequence of k-mers. The regions that we define as "hits" are all sub-sequences of contiguously query-labeled sequence, *e.g.* for a sequence/parse pair such as:

$$(X_i = ATCGCCCG, \phi_{max} = -, -, +, +, +, +, +, +)$$

The "hit" in this example as defined in *hmmSEEKR* is the sub-sequence CGCCCG. There may be any number of hits within a sequence, but a hit is always composed of contiguous "+" (query) labeled nucleotides. The Viterbi path through each sequence was calculated as in algorithms 5, 6 (Methods), and was implemented in corefunctions.py within the *hmmSEEKR* package. This code can be called as in section 3.4.3 within the mSEEKR.py program.

The score for the HMM, S, is calculated for each "hit" within the transcript. We assume that the set of state labels from the Viterbi algorithm are now known. The score is the log-likelihood of the hit belonging to the query state relative to the null state (Equation 3.1; Methods 4.2, HMM Score). In Equation 3.1, x_i is an individual k-mer within the hit, L is the length of the hit, and the probability distributions $p(x_i|+)$ and $p(x_i|-)$ correspond to the k-mer frequency distributions in the emission matrix.

$$S = \sum_{i=1}^{L} [\log_2 p(x_i|+) - \log_2 p(x_i|-)] \tag{3.1}$$

Table 3.2 illustrates the output of mSEEKR.py when scanning mouse *Xist* using a human *XIST* query and parameters $\theta = (k = 4, \alpha = .9999, \beta = .9999)$ (Methods 4.1 Transition Parameters and k). Here, an HMM was trained on human *XIST* repeat A as the query hidden state, and the null hidden state was trained on the set of all unspliced lncRNAs in mice (Methods 3.4.3,[27]), and mouse *Xist* was scanned for matches to the repeat A query. In mouse *Xist*, repeat A is defined as spanning basepairs 292-713 [7], and *hmmSEEKR* called basepairs 201-748 as a hit to the repeat A query within mouse Xist (Table 3.2), yielding 100% recovery of the original sequence. The score S is displayed in the kmerLLR column in the output. As the original definition for repeat A was formally defined by the presence of the tandem repeat [6, 7], and not based on functionality, such as RBP motif enrichment or experimentally determined presence of protein binding, it is significantly more challenging to accurately assess the false positive rate of the HMM (Figure 3.3).

As there are no pre-existing annotations for these k-mer based sequence features within lncR-

Rank	Start	End	Length	kmerLLR	seqName
0	201	748	547	285.225	>xist
1	10295	11052	757	42.734	>xist
2	1532	1576	44	30.685	>xist
3	11326	11450	124	29.306	>xist
4	17941	17946	5	-0.968	>xist

Table 3.2: Output of *mSEEKR.py*. Each hit from scanning *XIST* with an HMM trained on the A-repeat of mouse *Xist* is shown and sorted by the *hmmSEEKR* score, "kmerLLR". The parameter vector $\theta = (k, \alpha, \beta)$ for this HMM is $\theta_A = (4, .9999, .9999)$.

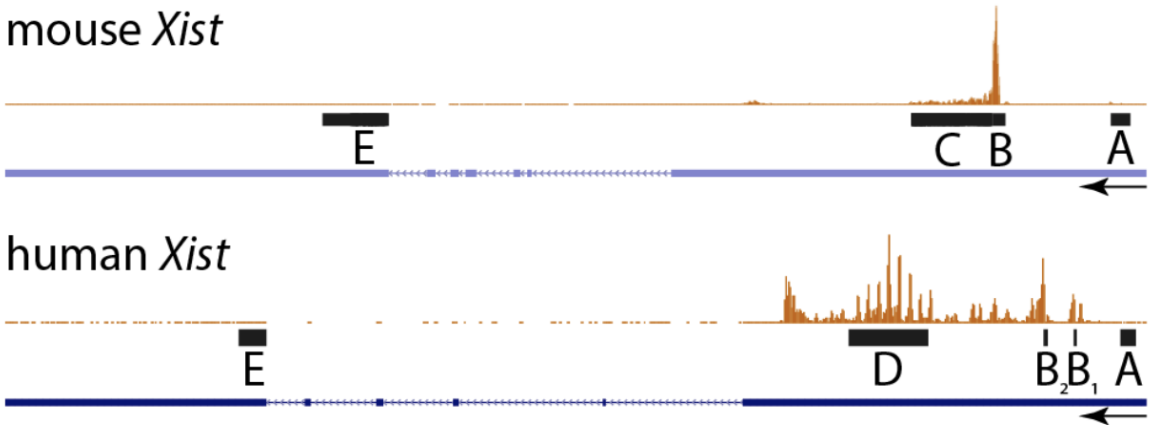

Figure 3.3: UCSC wiggle- density display of HNRNPK CLIP data (orange) aligned over the mouse (mm9) and human (hg38) Xist genomic loci. Mouse and human clip data are from [28, 29]. Arrows denote direction of Xist transcription. Black rectangles indicate genomic locations of mouse and human repeat sequences used in this work.

NAs, it is difficult to define what constitutes a true positive and a true negative. For example, HNRNPK binds large regions of sequence outside the formal definitions of the B- and D-repeats in human *XIST* (Figure 3.3). To help elucidate what sequence features within *XIST* and other lncRNAs are functional, beyond mere repetitiveness, we turned to protein binding data to identify proteins most associated with core tandem repeat domains within *XIST*.

2.3 Xist Associated RBPs

A crucial feature of the *XIST* transcript are the 4 core repeats that bind unique subsets of RNA binding proteins [3–5, 9, 11]. Several RBPs have been shown to be crucial for the function of *XIST*, including HNRNPK, RBM15, and several others [3–5, 9, 11, 30]. The tandem repeats have been defined based only on their repetitiveness, but Figure 3.1 demonstrates that a non-repetitive sequence can have the same motif content as a repetitive one. Indeed, examination of

ENCODE eCLIP data for HNRNPK reveals significant eCLIP binding adjacent to and extending beyond the formally defined B- and D-repeats of *XIST* (Figure 3.3). Therefore, we hypothesized that the functional elements of a lncRNA such as *XIST* are not defined by repetitiveness but rather by enrichment of protein binding motifs [2, 4, 14, 31, 32]. To further test *hmmSEEKR*, we sought to identify the proteins most associated with the A,B,D,E-repeats of *XIST* so that we could validate the predictive power of our model against existing protein binding data.

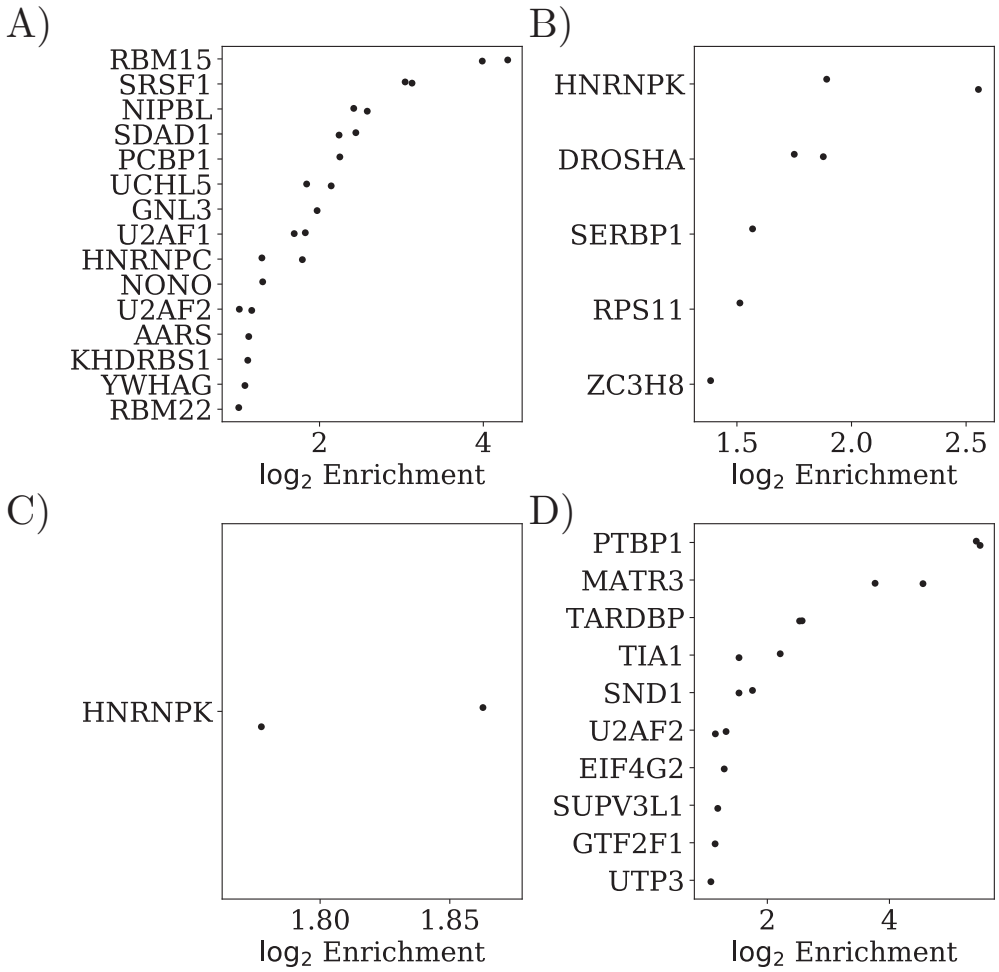

Figure 3.4: Proteins selectively enriched within tandem repeats of *XIST*. A) $\log_2$ ratio of read density for Repeat A compared to the remainder of *XIST*. B) $\log_2$ ratio of read density for Repeat B compared to the remainder of *XIST* is shown. C) $\log_2$ ratio of read density for Repeat D compared to the remainder of *XIST*. D) $\log_2$ ratio of read density for Repeat E compared to the remainder of *XIST*. A-D) Individual points within each protein represent biological replicates.

To do this, we compared eCLIP signal for all proteins with data in the ENCODE database inside each tandem repeat relative to the remainder of the *XIST* transcript (Methods 4.5, Xist En-

riched Proteins). We found that each repeat within *XIST* contained at least one protein that was at least 2x more enriched in within the repeat than the remainder of the *XIST* transcript (Figure 3.4A). RBM15 and SRSF1 were found to be the most specific proteins to repeat A. RBM15 is required for full *XIST* silencing functionality [10], and that RBM15 recruits the m6A complex to the 5' region of *XIST* [33]. SRSF1 has previously been shown to bind the A-repeat of *XIST* but its function within the sequence is currently unknown.

Both the B-repeat and D-repeat were found to be significantly enriched for HNRNPK over the rest of the transcript (Figure 3.4B-C). HNRNPK is required for recruitment of PRC1 to the *XIST* transcripts [3]. Deletion of the B-repeat region has previously been shown to be sufficient to abrogate *XIST* dependent PRC recruitment, as well as *XIST* mediated silencing. Our analysis showed that DROSHA is also enriched within the B-repeat of *XIST*, however no previous studies have identified DROSHA as a direct interactor with *XIST* and so the function of this interaction is unknown.

Xist Repeat	eCLIP Data
A	RBM15,SRSF1
B	HNRNPK
D	HNRNPK
E	TIA1,MATR3,PTBP1

Table 3.3: Most enriched RBPs for each tandem repeat domain in *XIST*.

The E-repeat of *XIST* was highly enriched for several proteins. The E-repeat itself is composed primarily of T-rich sequence, and therefore the proteins we observe often contain similar motifs ([32]). PTBP1, MATR3, TARDBP, and TIA1 were all found to be the most enriched within the E-repeat. PTBP1 has previously been shown to be required for proper *XIST* expression and splicing during development [34]. TARDBP depletion is also associated with increased expression of mis-spliced *XIST* transcripts [34]. A recent study found that MATR3 and PTBP1 form a condensate mediated by the *XIST* E-repeat and this condensate is required for *XIST* localization to the Xi [35].

2.4 Detection of Rsx Domains

Our *Rsx* study in Chapter 2 relied on the existence of tandem repeats within the *Rsx* transcript to perform the domain based SEEKR analysis. On a whole transcript level, *XIST* and

Rsx were slightly anti-correlated despite their shared function. The relationship between the two sequences didn't become apparent until we parsed out the tandem repeats within *Rsx* and performed pairwise comparisons between each *XIST* tandem repeat.

We hypothesized that we could use *hmmSEEKR* to identify regions of non-linear sequence similarity between *XIST* and *Rsx* without *a priori* identification of tandem repeats. To do this, we trained 4 separate HMMs on the A,B,D, and E-repeats of *XIST* at $k \in \{2, 3, 4, 5, 6\}$ and all pairwise combinations of $\alpha, \beta \in \{.5, .75, .9, .99, .999, .9999\}$. The results outlined in Figure 3.5 and Table 3.4 are using the set of parameters that yielded the best F1 score (Methods 3.4.X).

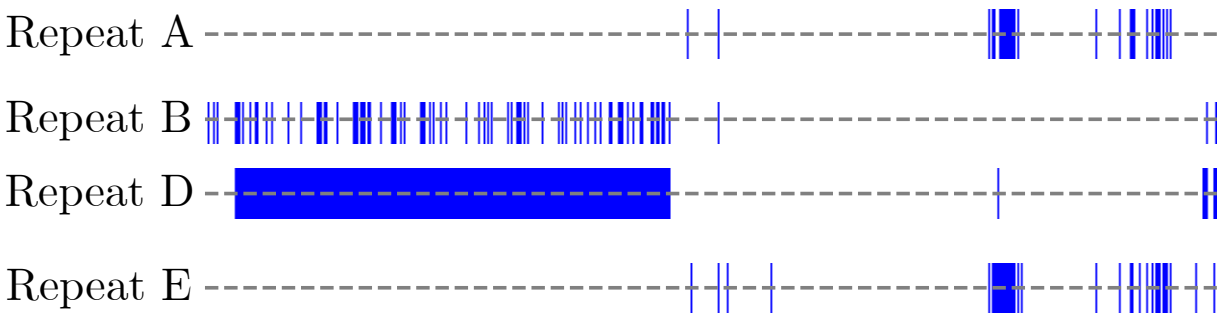

Figure 3.5: *hmmSEEKR* hits within koala *Rsx* for an HMM trained on each *XIST* repeat. Blue regions represent hits to the query, whereas dashed grey lines represent null sequence. The parameter vectors $\theta = (k, \alpha, \beta)$ for each repeat A,B,D,E used as a query were: $\theta_A = (4, .9999, .9999)$, $\theta_B = (4, .9999, .9999)$, $\theta_D = (2, .9999, .9999)$, $\theta_E = (4, .9999, .9999)$.

We found that the HMM trained on the *XIST* D-repeat ($\theta_D = (2, .9999, .9999)$) captured the entire tandem repeat from transcript coordinates 1,000-14,000 (Figure 3.5, Table 3.4). Likewise, the HMM trained on the *XIST* B-repeat ($\theta_B = (4, .9999, .9999)$) captured a significant portion of *Rsx* repeat 1, and these regions were significantly more C-rich than the remainder of *Rsx* repeat 1. Additional C-rich sequence were also identified at the 5' and 3' regions of *Rsx* (Figure 3.5). The HMMs trained on the A-repeat ($\theta_A = (4, .9999, .9999)$) and E-repeat ($\theta_E = (4, .9999, .9999)$) both significantly aligned with *Rsx* repeat 4, with 100% precision and 25.45% recall of the full-length repeat 4. In our original analysis, we mapped A,E-repeats to repeats 2,3, and 4 in *Rsx*, however repeats 2 and 3 had relatively marginal correlation to the A,E-repeats, whereas the A,E→4 relationship was the strongest of all comparisons in the original *Rsx* analysis [2].

Rsx Repeat Name	Start	End	Xist Association	Precision	Recall
1	1000	14000	B,D	98.42%	100%
2	17500	21000	A,E	100%	1.32%
3	21500	22500	A,E	-	0%
4	23000	27500	A,E	100%	25.45%

Table 3.4: Precision and recall for each HMM using the tandem repeat definitions from [2]. Precision is defined as the number of correct nucleotides relative to the number of incorrect nucleotides, whereas recall is the fraction of the total domain retrieved by the HMM. The parameter vectors $\theta = (k, \alpha, \beta)$ for each repeat A,B,D,E used as a query were: $\theta_A = (4, .9999, .9999)$, $\theta_B = (4, .9999, .9999)$, $\theta_D = (2, .9999, .9999)$, $\theta_E = (4, .9999, .9999)$.

2.5 Sequence based prediction of RBP binding in *KCNQ1OT1*

There is at least one conserved lncRNA in the mammalian transcriptome that is known to silence gene expression in *cis* through a PRC-mediated mechanism similar to *XIST*, *KCNQ1OT1*. The mouse *Kcnq1ot1* has been shown to silence megabase scale regions of chromosome 7 and human *KCNQ1OT1* silences a large region of chromosome 11. Unlike *XIST*, the sequence of *KCNQ1OT1* is not predominantly comprised of tandem repeat domains (Figure 3.6), barring a region in the 3' region of the sequence.

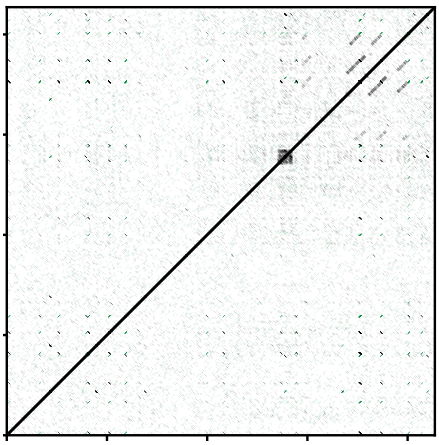

Figure 3.6: Dot plot alignment of human *KCNQ1OT1* against itself using flexidot software [36]. The window parameter is 20bp and the threshold is 40%. Black regions correspond to windows that succesfully aligned against each other.

Furthermore, eCLIP data for RBPs known to be essential for *XIST*'s function show that they predominantly bind in the 5' half of *KCNQ1OT1*'s sequence (Figure 3.7). Therefore, we sought to use the *hmmSEEKR* package to identify functional sub-sequences within *KCNQ1OT1* that are similar in k-mer content to *XIST* A-,B-,D-, and E-repeats.

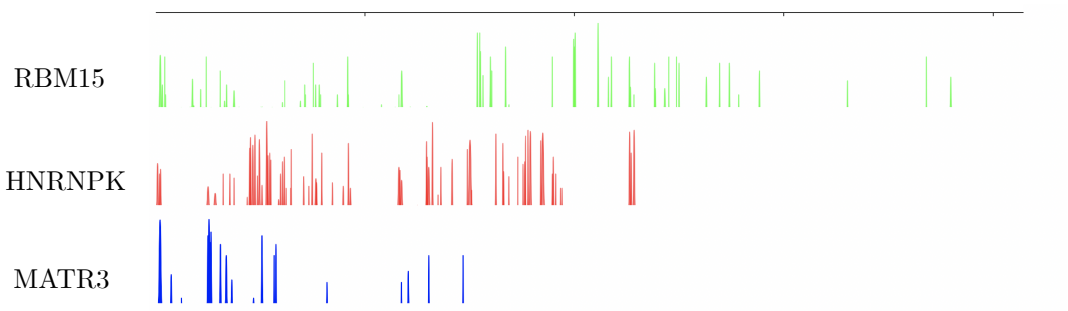

Figure 3.7: Browser tracks for RBM15 (top), HNRNPK (middle), and MATR3 (bottom) within the *KCNQ1OT1* locus. Tracks represent peak intensity from ENCODE narrowPeak files for each protein, regions with no signal were not called as statistically significant peaks.

KCNQ1OT1 contained regions of similarity to each of the *XIST* HMMs that we trained (Figure 3.8 A-D) with parameterizations $\theta_A = (4, .9999, .9999), \theta_B = (4, .9999, .9999), \theta_D = (2, .9999, .9999), \theta_E = (4, .9999, .9999)$. We found that the 5' region of *KCNQ1OT1*'s sequence in particular containins regions of similarity to A-,B-,D-, and E-repeats of *XIST* (Figure 3.8 A-D) – suggesting that this region of *KCNQ1OT1* could be a multipurpose binder of numerous RBPs. Indeed, eCLIP data for RBM15 and SRSF1 (Figure 3.8 A), HNRNPK (Figure 3.8 B-C), and MATR3 and PTBP1 (Figure 3.8 D) reveal elevated read density for all these proteins at the 5' region of *KCNQ1OT1*'s sequence. Additionally, a large portion of the inner region of *KCNQ1OT1*'s sequence was identified by *hmmSEEKR* to have similarity to the B- and D-repeats of *Xist*. (Figure 3.8 B-C). Similar to actual B- and D-repeats of *XIST*, these regions within *KCNQ1OT1* correspond to elevated HNRNPK read density (Figure 3.8 B,C).

We then sought to compare the read density for each protein in Table 3.3 found within our *hmmSEEKR* prediction for the associated *XIST* query against a randomized shuffling of the hits found within *KCNQ1OT1* (Method 3.4.6 randomization). We found that the A-repeat HMM significantly outperformed randomized shuffles for both RBM15 (Figure 3.8 E; $p < 10^{-8}$, chi-square test) and SRSF1 (Figure 3.8 F; $p < 10^{-5}$, chi-square test). HNRNPK was also significantly better predicted by the HMMs trained on the B- and D-repeats of *XIST* (Figure 3.8 G,H;$p < 10^{-33}$ B-repeat, $p < 10^{-51}$ D-repeat, chi-square test). Finally, the E-repeat trained HMM significantly outperformed a shuffled parse for both MATR3 (Figure 3.8 I, $p < 10^{-28}$, chi-square test) and PTBP1 (Figure 3.8 J; $p < 10^{-43}$, chi-square test). Thus, our predictions from *hmmSEEKR* would support our hypothesis that *KCNQ1OT1* does contain shared sequence features with *XIST*. Un-

der the traditional hypotheses of sequence alignment, *XIST* and *KCNQ1OT1* have no shared sequence features [20, 21, 23], however using a k-mer based similarity metric we have identified $> 10^4$ basepairs of sequence in *KCNQ1OT1* that map to functional domains in *XIST*.

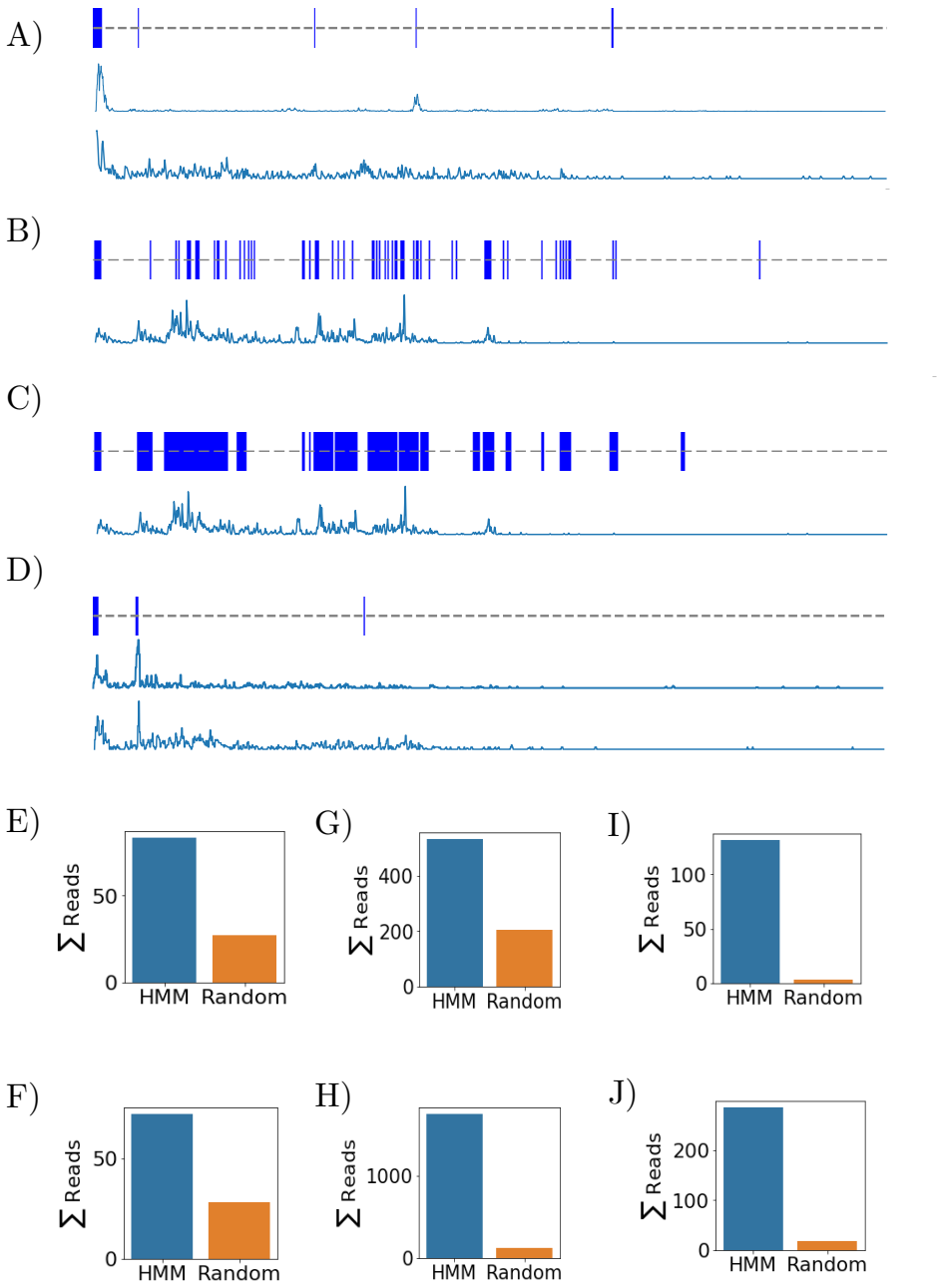

Figure 3.8: *hmmSEEKR* predicts *XIST* associated protein binding in *KCNQ1OT1*. A) HMM viterbi parse showing hits to the A-repeat in *KCNQ1OT1* (blue), RBM15 eCLIP read density (middle plot), and SRSF1 eCLIP read density (bottom plot). B) HMM viterbi parse showing hits to the B-repeat (top, blue regions), and HNRNPK eCLIP read density (bottom). C) HMM viterbi parse in *KCNQ1OT1* for the D-repeat (top, blue regions) and HNRNPK eCLIP read density (bottom). D) HMM viterbi parse for E-repeat in *KCNQ1OT1* (top, blue regions) and eCLIP read density for MATR3 (middle) and PTBP1 (bottom). E-J) Total eCLIP reads assigned to the HMM or randomized parses for RBM15, SRSF1, HNRNPK (B-repeat), HNRNPK (D-repeat), MATR3, and PTBP1 respectively. HMM parameterizations were $\theta_A = (4, .9999, .9999), \theta_B = (4, .9999, .9999), \theta_D = (2, .9999, .9999), \theta_E = (4, .9999, .9999)$ for A-,B-,D-,and E- respectively.

2.6 Transcriptome-wide RBP Prediction

Many of the proteins that bind to *XIST* and *KCNQ1OT1* bind throughout the transcriptome [29]. Furthermore, *XIST* clusters with thousands of transcripts in the human genome based on k-mer content [14]. Therefore, we hypothesized that *XIST*-like sub-sequences may be found throughout the transcriptome, both in lncRNAs as well as in pre-mRNAs. To test this hypothesis, we trained 4 HMMs on the A-,B-,D-, and E-repeats of human *XIST* using parameters in Table 3.5 (Methods 4.1, Transition parameters) and used *hmmSEEKR* to scan the set of all unspliced coding and non-coding transcripts in the human genome. The Viterbi parses for the A-,B-,D-, and E-repeat HMMs extracted sequences from the transcriptome that were significantly more similar to the query compared to the unparsed sequences (Figure 3.9). We then compared the results of our HMM against ENCODE eCLIP data for proteins that we have shown are enriched within each *XIST* repeat (Table 3.3).

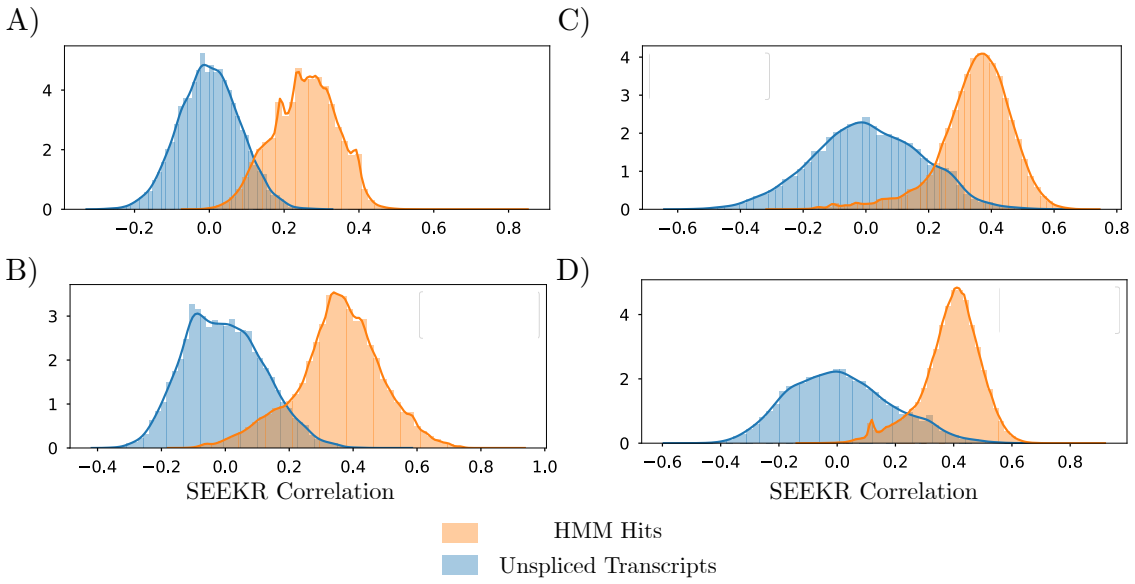

Figure 3.9: *hmmSEEKR* models trained on *XIST* tandem repeats extract sub-sequences with higher SEEKR correlation to the query than unparsed sequences using parameters listed in Table 3.5. A) A-repeat HMM hits relative to unparsed sequences, B) B-repeat HMM hits relative to unparsed sequences. C) D-repeat HMM hits relative to unparsed sequences. D) E-repeat HMM hits relative to unparsed sequences. x-axis represents the SEEKR correlation from the query to each transcript. Parameterizations as in Table 3.5 (Methods 4.1, Transition parameters).

The *hmmSEEKR* HMMs significantly out-performed randomized HMM parses for each of the proteins we tested. The A-repeat trained HMM successfully captured significantly more reads for

RBM15 (Figure 3.10 A left; $p \ll 10^{-100}$, chi-squared test) and SRSF1 (Figure 3.10 B; $p \ll 10^{-100}$, chi-squared test) than did the randomized hits. Furthermore, true HMM hits throughout the transcriptome had significantly more reads per hit than the shuffled parses for RBM15 eCLIP data as well as significantly more reads per hit than three RBPs that are not known to associate with *XIST*: UTP3, RBM22, and ZC3H8 (Figure 3.11 A; $p \ll 10^{-100}$, student's t-test) and SRSF1 (Figure 3.11 B; $p \ll 10^{-100}$, student's t-test).

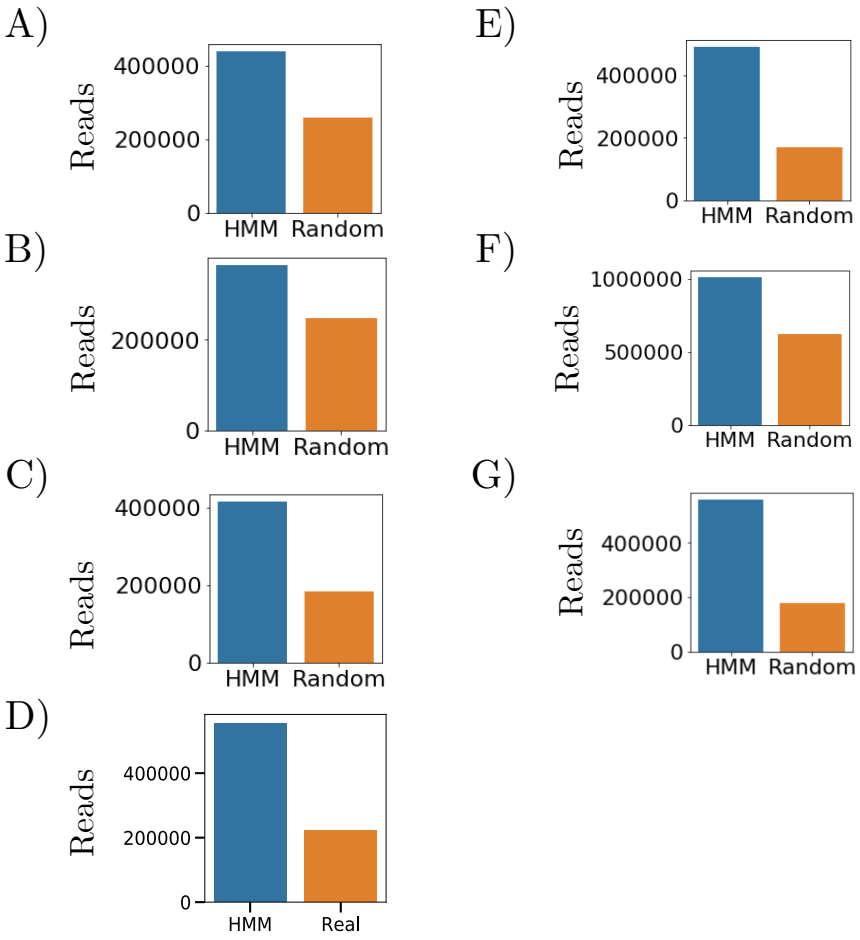

Figure 3.10: *hmmSEEKR* models trained on XIST tandem repeats predict RBP binding regions throughout the transcriptome. A-G) Total number of eCLIP reads assigned to HMM Viterbi parse hits compared to randomized shuffling of hits within each transcript for RBM15 (A), SRSF1 (B), HNRNPK B-repeat (C), HNRNPK D-repeat (D), MATR3 (E), TIA1 (F), PTBP1 (G) Parameterizations as in Table 3.5 (Methods 4.1, Transition parameters).

The B-repeat trained model had significantly more reads assigned from HNRNPK eCLIP data than the random parse (Figure 3.10 C, $p \ll 10^{-100}$, chi-squared test), and significantly more reads per hit than the randomized parses (Figure 3.11 C, $p \ll 10^{-100}$, student's t-test).

The B-repeat model is the only HMM that has a smaller mean of $\log_2$ read counts than UTP3 per hit (1.09, 1.24 respectively; $p < 10^{-52}$ student's t-test), but had significantly more reads per hit than RBM22 and ZC3H8 ($p \ll 10^{-100}$, student's t-test). The D-repeat HMM captured more reads than the B-repeat HMM (Figure 3.10 C-D), and overlapped significantly more reads than randomized parses (Figure 3.10 D, $p \ll 10^{-100}$, chi-square test) as well as more reads per hit than the randomized parses and RBM22, ZC3H8, and UTP3 (Figure 3.11 D, $p \ll 10^{-100}$, student's t-test).

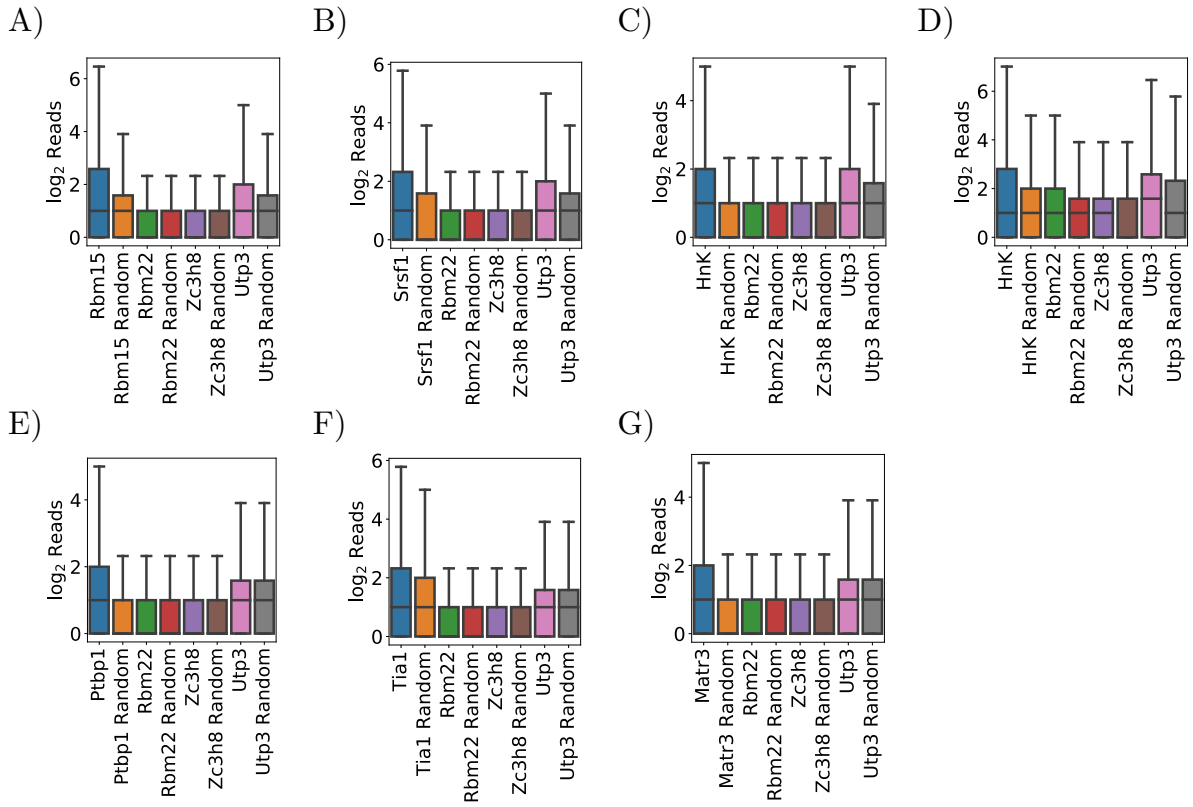

Figure 3.11: *hmmSEEKR* models trained on *XIST* tandem repeats predict binding sites of *XIST* associated RBPs throughout the transcriptome. A-G) $\log_2$ read count per hit distribution for the true HMM hits compared to the shuffled parses, for RBM15 (A), SRSF1 (B), HNRNPK B-repeat (C), HNRNPK D-repeat (D), MATR3 (E), TIA1 (F), PTBP1 (G). Also shown are the distribution of $\log_2$ read counts for non-*XIST* associated proteins RBM22, ZC3H8, and UTP3. Parameterizations as in Table 3.5 (Methods 4.1, Transition parameters).

Finally, the E-repeat trained HMM also successfully predicted associated protein binding better than randomized parses (Figure 3.11 E-G). MATR3 had significantly more reads assigned by the true HMM (Figure 3.10 E, $p \ll 10^{-100}$, chi-square test) and more reads per hit than random (Figure 3.11 E, $p \ll 10^{-100}$, student's t-test). The E-repeat trained HMM assigned significantly

more reads from TIA1 eCLIP (Figure 3.10 F, $p \ll 10^{-100}$, chi-squared test) and the number of reads per hit for the true HMM parses and the randomized parses was significantly higher (Figure 3.11 F, $p \ll 10^{-100}$, student's t-test). PTBP1 displayed the highest ratio of assigned reads between the true HMM and randomized parses over all comparisons (Figure 3.10 A-G) and assigned significantly more eCLIP reads than the randomized parses (Figure 3.10 G, $p \ll 10^{-100}$, chi-squared test) as well as more reads per hit compared to random (Figure 3.11 G, $p \ll 10^{-100}$, student's t-test). MATR3, PTBP1, and TIA1 all had significantly more reads per hit in the HMM parse compared to the RBPs that do not associate with *XIST* ($p \ll 10^{-100}$, student's t-test). These data show that domains with k-mer based similarity to *XIST* in the transcriptome are binding *XIST*-associated proteins in a manner consistent with the hypotheses in [2, 14].

3 Discussion

The non-coding portion of our genome is a vast and nebulous network of RNA transcripts whose functions are largely unknown – and if they are, their mechanism(s) of action have proven difficult to understand. A major obstacle to better understanding lncRNA function is the relatively unknown sequence-to-function relationship in lncRNAs. In particular, even the traditional principles of conservation analysis seem to provide little insight into lncRNAs [12, 16, 22]. Therefore, an entirely different toolbox is needed if computational prediction of functional lncRNAs are to be made.

Our lab has recently developed a k-mer based algorithm for quantification of sequence similarities between non-coding transcripts [14]. This method was built for comparison of full-length sequences, however it is clear from the relationship between *XIST* and *Rsx* that much of *XIST*'s sequence bears no resemblance to *Rsx* [2], and often only sub-sequences within a larger lncRNA may be conserved between species [12, 16]. A statistical model was necessary to identify where within a lncRNA sequence there are regions of non-linear sequence similarity between the lncRNA and a query domain that has some *a priori* known function, *e.g.* the tandem repeats of *XIST*.

Here, we have developed a python package, *hmmSEEKR*, that uses an HMM to parse sequences of interest into regions that either have k-mer based similarity to some known functional domain, or not. *hmmSEEKR* utilizes the underlying model of lncRNA sequence functionality in SEEKR [14] and hypothesizes that the functionality of lncRNA can be localized within func-

tional modules [1, 2, 12, 16, 17] in order to more precisely pin down the sequence relationship between two lncRNAs.

We found that *hmmSEEKR* was able to reproduce the non-linear sequence relationship between *XIST* and *Rsx* without the *a priori* extraction of tandem repeat domains from *Rsx* ([2], Figure 3.5). Additionally, *hmmSEEKR* identified regions of non-linear sequence similarity between *XIST* and *KCNQ1OT1* for each of the *XIST* functional domains that an HMM was trained on. This indicates that *KCNQ1OT1* contains the necessary sequence information bind the RBPs required for PRC mediated silencing, which we verified with statistical analysis of ENCODE eCLIP data [29]. Finally, we found that *hmmSEEKR* models trained on A-,B-,D-, and E-repeats within *XIST* overlapped with significantly more eCLIP reads for each query's associated proteins (Table 3.3) than randomly generated parses, and in all cases except TIA1 the eCLIP reads per hit were significantly higher for the *hmmSEEKR* parses than the randomized parses. *hmmSEEKR* was therefore able to predict functional binding locations for *XIST*-associated proteins throughout the transcriptome.

There may be numerous transcripts in our genome that contain the necessary information to function similarly to *XIST*, *Rsx*, and *KCNQ1OT1*. *hmmSEEKR* represents the first tool designed to consider functional analogy rather than homology when modeling function between RNA transcripts. Going forward, we hope to computationally predict and experimentally verify transcripts that *hmmSEEKR* identifies as containing the necessary information to silence through PRC recruitment.

4 Methods

4.1 Parameter Estimation

A hidden markov model is comprised of a stochastic transition matrix A, where each row vector ψ_i corresponds to a categorical distribution representing the probability of hidden state i transitioning to state j. The emission matrix E, where each row corresponds to a categorical distribution of k-mer frequencies. Finally, there is the initialization matrix π that describes the probability of starting at each hidden state at $t = 1$.

Emission Parameters

We defined the emission distribution for the query and null hidden states to be k-mer frequencies taken from a set of training sequences. For the query hidden state, we defined our training sets to be *XIST* repeats A,B,D, and E. For the null hidden state, we defined the set of training sequences to be the set of unspliced lncRNAs in the human transcriptome. A separate HMM was trained for each *XIST* repeat.

Some of the *XIST* repeats are quite short, for example *XIST* repeat B is only $\approx$ 200bp in length. Therefore, it is often the case that there are fewer nucleotides than there are parameters to estimate leading to zeros in the probability distribution. In reality, zero probability k-mers are both unlikely, as many of the motifs for RBPs that bind *XIST* have no non-zero probabilities, and also mathematically intractable, as log-space for probability calculations is required for long sequences.

To correct for this parameterization problem we employed pseudo-counts, specifically we use a +1 pseudo-count for all k-mers in the training set (*i.e.* initialize the counts array to 1). The hypothesis driving the pseudo-count is that despite the limited training data implying zero probability events, this only occurs because there is insufficient data to sample enough k-mers and their frequencies. Given the transition matrix A, each row i (representing a hidden state's emission distribution) within A is a categorical distribution ψ_i comprised of the frequencies for each possible k-mer. It can be shown that the *maximum a posteriori*, or MAP, estimate for the k-mer frequency distribution ψ_i is achieved by adding a pseudo-count of 1 to each k-mer [24]. This is achieved by placing a Dirichlet prior on the k-mer frequency distribution:

$$\psi_i \sim Dir(\alpha_i^{AAA} + 1, \alpha_i^{AAT} + 1, \ldots, \alpha_i^{GGG} + 1)$$
$$x_i \sim Cat(\psi_i^{AAA}, \psi_i^{AAT}, \ldots, \psi_i^{GGG})$$

The α parameters in the Dirichlet distribution would be the observed counts of each k-mer, and the +1 represents the pseudo-count. As the α for a given k-mer increases, the probability "concentration" around that k-mer increases – so when a sample of ψ_i is drawn – the k-mers with high α parameters are going to have the highest frequencies, and therefore most likely to be drawn when an observation x_i is drawn. That is to say, the best estimate of the k-mer frequencies, given the data available and the uncertainty in it, is achieved by adding a count of 1 to all

k-mers. Intuitively this makes sense, as this addition has the largest impact when there is very little data to work with (less than 4^k k-mers to train from), and very little impact when there is substantial data as is the case for the null hidden state, which is trained on many thousands of k-mers.

Transition Parameters and k

We used the SEEKR algorithm to calculate a score for how well the HMM performed at identifying query-like sequence regions based off their k-mer similarity to the original query. For a given set of parameters, the Kullback-Leibler diveregence was calculated between a SEEKR correlation distributions of parsed and unparsed sequences, by running *hmmSEEKR* on all transcripts in the human unspliced transcriptome and comparing the Viterbi "hits" to the original, unspliced transcripts. The KL-divergence is a metric of "distance" between two probability distributions.

The KL-Divergence is defined as [37]:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x} \tag{3.2}$$

Where D_{KL} is the KL-Divergence between two probability distributions P and Q. P represents the SEEKR score distribution of the HMM parse "hits" (contiguously "+" labeled k-mers within the sequence) to the query sequence, and Q represent the SEEKR score distribution of unparsed sequences to the query sequence, representing a 'reference'. If $P = Q$ then $D_{KL} = 0$. D increases as the probability distributions become more dissimilar to each other. $D_{KL}(P||Q) \neq D_{KL}(Q||P)$, so for consistency $D_{KL}(P||Q)$ was always calculated. Being comprised of unparsed sequences, $Q(x)$ just depends on k. In the real dataset, P and Q are histograms calculated from the data, and so the calculation of D_{KL} compares the frequencies of SEEKR correlations in each bin for the two distributions. The larger the difference between P and Q, the more successful the HMM was at parsing out highly correlated regions to the query.

The goal was to find a set of parameters, α $(+ \to +)$, β $(- \to -)$, and k, that maximize the expression in equation 3.3.

$$\underset{\alpha,\beta,k}{\operatorname{argmax}} \left[\mu_S \sum_{x \in X} P(x|\alpha,\beta,k) \log \frac{P(x|\alpha,\beta,k)}{Q(x|k)} \right] \tag{3.3}$$

Where μ_S is the average HMM score (log-likelihood of the sequence, Methods 4.2 "HMM Score") over all hits in the transcriptome, $P(x)$ is the distribution of SEEKR scores for parsed out hits, and $Q(x)$ is the distribution of SEEKR scores for full length unspliced transcripts in the human transcriptome:

Query	k	α	β
Repeat A	4	.9999	.9999
Repeat B	4	.9999	.9999
Repeat D	2	.75	.9999
Repeat E	3	.5	.9999

Table 3.5: Best values of k, the $+ \rightarrow +$ parameter α, and $- \rightarrow -$ parameter β as determined through a grid search based approach to identify sequences with the highest SEEKR correlation to the query.

We found that β must always be close to 1, whereas significant variance was found for the value of α amongst the different *XIST* queries (Table 3.5). We also found that there was often significant overlap of hits between the varying sets of parameters for each query, implying that there was general consistency regardless of general parameterization. (Note: Repeat A, $k = 5$ scored higher than $k = 4$, however, the HMM model for $k = 5$ has a positive expectation for the log-likelihood matrix between query and null. All score based sequence models generally require a negative expectation [20] to prevent spurious "hits" due to random chance, and simulations we performed indicated this was a problem).

4.2 Pseudo-code and algorithms

The functions that drive much of analysis in the *hmmSEEKR* package are found in the core-functions.py file. These include the algorithms behind HMM analysis including the forward, backward, Viterbi, and Baum-Welch algorithm implementations. In addition, several functions crucial for parsing the supplied DNA sequences and outputting in human readable format are found within this file.

k-mer counting

k-mer counting is at the heart of both SEEKR and hmmSEEKR. This calculation is one of the simplest, but potentially most time consuming portions of the analysis. This is largely because there is no way around reading along the supplied string(s), and counting every k-mer

as encountered. To speed k-mer counting up, **Algorithm 1** has been implented using the cython package within Python, which is a library that allows for C-like implementation of python code.

The pseudo-code in Algorithm 1 is implemented in kmers.py and kmers.pyx and takes a FASTA file as input. The program then saves a dictionary, or hash map, with k-mers as keys and their counts as values is created, and as each k-mer is encountered within the sequence, that k-mer's count is incremented by one.

Algorithm 1: Counting k-mers from supplied sequences

```
Result: Python dictionary of k-mer counts
read in fasta file
strFasta ← concactenate fasta strings with delimiter character
intL ← calculate total length excluding delimiter
dictKmerMap ← initialize dictionary of k-mers with initial counts of 1
for k-mer in sequence do
    if k-mer in dictKmerMap then
        dictKmerMap[k-mer]←increment k-mer by 1
    else
        do not increment
    end
end
Save binary file of python dictionary containing counts
```

Ambiguous Nucleotides

Transcripts containing regions of ambiguous sequence are of concern when building the viterbi parse, as ambiguous k-mers are not included in the emission distributions of the hidden state. Therefore only sequences of k-mers containing A,T,C, or G are allowed. To adjust for any ambiguous positions within the sequence, **Algorithm 2** stores two lists of indices. The first stores the position, or index, of each allowable k-mer, and the second stores the position of any k-mer containining an "N" within it. If there are any "N" within the sequence, the list of k-mers passed into the Viterbi algorithm are split into distinct sub-sequences, because the regions flanking the ambiguous nucleotides are not adjacent to each other. Finally, when mapping the Viterbi parse back to the original sequence, the two lists of indices generated above are used to merge the results back to their original order.

An example would best illustrate what this portion of the code is trying to achieve. For simplicity, let $k = 1$, and our observation be $X = A, A, N, N, T, T$. Counting indices from 1, the

allowable k-mer indices in X are $\{1, 2, 5, 6\}$. The ambiguous k-mer indices are $\{3, 4\}$. As the sequence X is fragmented by two Ns at position 3 and 4, the flanking sequences are passed to the Viterbi algorithm to be parsed separately: $\{\{A, A\}, \{T, T\}\}$. The grouping algorithm then sorts the viterbi parse back in with the ambiguous nucleotides, $\{+, +, N, N, -, -\}$.

Algorithm 2: Generate unambiguous observed sequence

```
Result: Sequence of unambiguous k-mers
listO←initialize empty list to contain k-mers
listIdx←initialize empty list to contain k-mer indices
listAmbigIdx←initialize empty list to contain ambig k-mer locations
listKmers←list of k-mers that can be constructed from ATCG
intIndex←0
for k-mer in sequence do
    if k-mer in listKmers then
        Append k-mer to listO
        Append intIndex to listIdx
    else
        Append intIndex to listAmbigIdx
    end
    intIndex+1
end
return listO listIdx listAmbigIdx
```

Identifying hits from the Viterbi parse

Within *hmmSEEKR* a Viterbi parse is a sequence of state labels that have been assigned to each k-mer from a DNA sequence, *e.g.* if the DNA sequence was "AAAACCCC", and $k = 4$, then the k-mer sequence is $\{AAAA, AAAC, AACC, ACCC, CCCC\}$, and a hypothetical Viterbi parse would be a list such as $\{-, -, -, +, +\}$, where "$-$" represents the null state, and "$+$" represents the query state. **Algorithm 3** takes the list $\{-, -, -, +, +\}$ and converts it into a grouped list, *e.g.* $\{\{-, -, -\}, \{+, +\}\}$. This also exactly defines what term a hit within our HMM, consecutively labeled "$+$" regions with a sequence from the viterbi parse.

HMM Score

The score for the HMM is calculated for each "hit" within the transcript, as defined in the prior section. We assume that the set of state labels from the Viterbi algorithm are now known. This allows, for a hit X of length L, the likelihood that X was emitted from the query state over its entire length.

Algorithm 3: Group HMM hits

Result: Viterbi backtrack grouped into consecutive states
condition←Swap condition, *i.e.* '-' encountered
Key←Stores a FLAG that switches value when condition is met
previousNuc←Previous nucleotide
for *nucleotide in sequence* **do**
  currNucBool←TRUE/FALSE is *nucleotide* in *condition*
  prevNucBool←TRUE/FALSE is *previousNuc* in *condition*
  if *currNucBool*≠*prevNucBool* **then**
    swap *Key*
  else
    Key
  end
  previousNuc←nucleotide
end

$$P(X = \{x_1, \ldots, x_L\}|Y = \{+, \ldots, +\}) = \prod_{i=1}^{L} p(x_i|+)$$

This likelihood can be compared to the likelihood of X had it been emitted from the null state, and a likelihood ratio can be calculated.

$$S' = \frac{\prod_{i=1}^{L} p(x_i|+)}{\prod_{i=1}^{L} p(x_i|-)}$$

Due to the lengths of most sequences, these calculations are moved to log-space, yielding the formula for the score reported by *hmmSEEKR*. These calculations are implemented in Algorithm 4.

$$S = \sum_{i=1}^{L} [\log_2 p(x_i|+) - \log_2 p(x_i|-)]$$

Viterbi Parse

The premise behind the Viterbi algorithm is outlined in the introduction. Briefly, the algorithm finds the sequence of hidden states that maximizes the joint probability function of the HMM defined in equation 1.4, given the observations X, the transition matrix A, the emission matrix E, and the initial probabilities of each hidden state π. The viterbi algorithm is imple-

Algorithm 4: Log-likelihood

```
arrLLR←empty array with an element for each hit
hits←list of sequences that HMM called hits
Query←Frequency distribution of kmers in query
Null←Frequency distribution of kmers in null
for hit in hits do
    intLLRQuery← 0
    intLLRNull← 0
    for kmer in hit do
        intLLRQuery+ log P(kmer|Query)
        intLLRNull+ log P(kmer|Null)
    end
    arrLLR[hit]←intLLRQuery-intLLRNull
end
```

mented in *hmmSEEKR* as in **Algorithm 5**.

At each step in the Viterbi algorithm, the state transition that maximizes the probability at time t is recorded, for each state. That is, at time t, the "+" state has a most likely transition from either "+" or "-" from time $t-1$, and the "-" state at time t has a most likely transition from either "+" or "-" from time $t-1$ in the sequence. To identify which was the most likely, a back-trace must be calculated after all these calculations have been made, and this is implemented in **Algorithm 6**.

Algorithm 5: Viterbi parse

O←Sequence of kmers
E←Emission matrix
A←Transition matrix
π←Starting probability of each state
states← List of states *i.e.* [-,+]
dictViterbi←List of dicts with cumulative probability at each time t for each state

dictViterbi[1][query] means the cumulative probability at time $t = 1$ at state=query

dictMaxState←List of dicts containing the state at time $t - 1$ that maximized transition to each state at time t

Initial probabilities at each state at time $t = 1$
for *state* ***in*** *states* **do**
  dictViterbi[1][state]← $\log P(state|\pi) + \log P(O_1|state)$
end

Calculate probability of being in each state at time t and find which state transition maximizes this probability
for *t* ***in*** *range(2,length of O)* **do**

  Add new dictionary element to dictViterbi and dictMaxState
  for *state* ***in*** *states* **do**

    selectedState← start with the first state to calculate prob
    currMaxProb← $\log P(state|selectedState) + dictViterbi[t-1][selectedState]$

    for *checkState in remaining states* **do**

      tempProb← $\log P(state|checkState) + dictViterbi[t-1][checkState]$

      if *tempProb>currMaxProb* **then**
        set currMaxProb to tempProb
      else
        keep currMaxProb the same
        selectedState←checkState
      end
    end
    currMaxProb←currMaxProb+ $\log P(O_t|state)$
    dictViterbi[t][state]←currMaxProb
    dictMaxState[t][state]←selectedState
  end
end

Algorithm 6: Backtrack viterbi

backtrack←empty list to append maximizing states to
dictViterbi←List of dicts with cumulative probability at each time t for each state

dictViterbi[1][query] means the cumulative probability at time $t = 1$ at state=query

dictMaxState←List of dicts containing the state at time $t - 1$ that maximized transition to each state at time t

maxProbability←largest probability at time $t = T$
maxState←state associated with maxProbability
maxPriorState←dictMaxState[T][maxState], get state that transitioned to maxState at $t = T - 1$

Append maxPriorState to backtrack
for *t=T-2 to t=1* **do**
    Append dictMaxState[t+1][maxPriorState] to backtrack
    maxPriorState←dictMaxState[t+1][maxPriorState]
end
return backtrack with order flipped to start at $t = 1$

Forward Algorithm

The forward algorithm calculates the likelihood of the data, $P(X)$, as well as the probability of the sequence $x_{1:t}$ ending at state i at time t. The algorithm was implemented through the pseudo-code below in *hmmSEEKR*.

Algorithm 7: Forward Algorithm

$X \leftarrow$Sequence of kmers
$Y \leftarrow$Sequence of hidden states
$E \leftarrow$Emission matrix
$A \leftarrow$Transition matrix
$\pi \leftarrow$Starting probability of each state
states$\leftarrow$ List of states *i.e.* [-,+]

$\alpha_i(t) \leftarrow$forward probabilities, indexed by time t and state i *i.e.* the probability of being in state i at time t
$T \leftarrow$length of X

Initialize probabilities at $t = 1$
$\alpha_i(t = 1) \leftarrow \pi(i) + \log P(X_t|Y_t = i)$

for *t in T* **do**
  Append new entry to α
  for *state "i" in states* **do**
    currP$\leftarrow${+:0,-:0} : dict with entry for each state, indexed by state name
    for *state "j" in states* **do**
      $A_{ij}(t) \leftarrow P(Y_t = i|Y_{t-1} = j)$
      currP[j]$\leftarrow \alpha_j(t-1) + A_{ij}(t)$
    end
    $\alpha_i(t) \leftarrow$ logsumexp $(currP) + \log P(X_t|Y_t = i)$
  end
end

Backward Algorithm

The backward algorithm calculates the probability of the observed sequence $P(X)$, as well as the probability of sequence $x_{t+1:T}$ starting in state i at time t. The algorithm was implemented as below in *hmmSEEKR* and is primarily used in the Baum-Welch algorithm.

Algorithm 8: Backward Algorithm

$X \leftarrow$Sequence of kmers
$Y \leftarrow$Sequence of hidden states
$E \leftarrow$Emission matrix
$A \leftarrow$Transition matrix
$\pi \leftarrow$Starting probability of each state
states$\leftarrow$ List of states *i.e.* [-,+]

$\beta_i(t) \leftarrow$backward probabilities, indexed by time t and state i *i.e.* the probability of being in state i at time t
$T \leftarrow$length of X

Initialize probabilities at $t = T$
$\beta_i(t = T) \leftarrow 0$; probability of 1 in log space

for *T to t=1* **do**
  Append new entry to β
  for *state "i" in states* **do**
    currP$\leftarrow${+:0,-:0} : dict with entry for each state, indexed by state name
    for *state "j" in states* **do**
      $A_{ij}(t) \leftarrow P(Y_{t+1} = j | Y_t = i)$
      $E_j(t+1) \leftarrow P(X_{t+1} | Y_{t+1} = j)$
      currP[j]$\leftarrow \beta_j(t+1) + A_{ij}(t) + E_j(t+1)$
    end
    $\beta_i(t) \leftarrow \text{logsumexp}\,(currP)$
  end
end
return β inverted s.t. $1 \rightarrow T$ rather than $T \rightarrow 1$

Baum-Welch Algorithm

This algorithm updates the transition matrix parameters. The premise of this algorithm is outlined in the introduction. Briefly, the BW algorithm calculates the expected number of "observations" of each hidden state i, as well as the expected number of "observed" transitions from each hidden state i to all possible states j (which may include i, and in *hmmSEEKR* does).

Algorithm 9: Baum Welch Parameter Update

$X \leftarrow$Sequence of kmers
$Y \leftarrow$Sequence of hidden states
$E \leftarrow$Emission matrix
$A \leftarrow$Transition matrix
$\pi \leftarrow$Starting probability of each state
states$\leftarrow$ List of states *i.e.* [-,+]

$\alpha(t) \leftarrow$forward probabilities
$\beta(t) \leftarrow$backward probabilities

$\gamma_i(t) \leftarrow \frac{P(Y_t=i,X|A,E,\pi)}{P(X_t)}$; probability of being in state i at time t
$\epsilon_{ij}(t) \leftarrow \frac{P(Y_t=i,Y_{t+1}=j,X|A,E,\pi)}{P(X_t)}$; probability of being in state i at time t and state j at time t+1

for *each time "t" in sequence* X **do**
  for *state "i" in states* **do**
    $\gamma_i(t) \leftarrow \alpha_i(t) + \beta_i(t) - \text{logsumexp}\left(\alpha_i(t) + \beta_i(t)\right)$
    for *state "j" in state* **do**
      $A_{ij}(t+1) \leftarrow \log P(Y_{t+1} = j | Y_t = i)$
      $E_j(t+1) \leftarrow \log P(X_{t+1} | Y_{t+1} = j)$
      numerator$\leftarrow \alpha_i(t) + A_{ij}(t) + \beta_j(t+1) + E_j(t+1)$
      denominator$\leftarrow \log P(X_t)$

      $\epsilon_{ij}(t) \leftarrow$numerator-denominator
    end
  end
  Append new entry to γ list of dicts
  Append new entry to ϵ list of dicts
end
Sum γ and ϵ over all values of T for all states i and j. The sum below represents the expected number of transitions from state i to state j over the sequence.

$$A^*_{ij} = \frac{\sum_t \epsilon_{ij}(t)}{\sum_t \gamma_i(t)}$$

Do not update emission matrix E.

return A^*

Program	Input	Purpose
kmers.py	fasta file	Count k-mers within fasta files provided
train.py	counts files from kmers.py for query and null	Generate matrices that define the HMM
mSEEKR.py	fasta file for sequence(s) of interest, path to output of train.py	retrieve the viterbi parse of the sequence of interest to identify potential functional domains
bw.py	HMM training sequences, initial parameterizations	provide MLE of transition parameters

Table 3.6: Individual programs within the *hmmSEEKR* package.

4.3 Python implementation

hmmSEEKR has been implemented as a python package in order to rapidly train HMMs and scan sequences of interested for potential function domains. Given the general algorithms defined above, which roughly correspond to individual functions within the *hmmSEEKR* package, a thorough guide is provided for how the algorithms have been implemented.

Installation

Table 3.6 provides an overview of the different python programs within the *hmmSEEKR* package. The first step is to ensure that Python3.6 or greater is installed, as well as the SEEKR package and hmmSEEKR repository which can be retrieved entering the following commands:

```
> pip install seekr
> git clone https://github.com/spragud2/mSEEKR
> cd mSEEKR/
> python setup.py build_ext --inplace
```

k-mer frequencies

To count k-mers for a set of sequences, a single fasta file containing any number of sequences is required. The program counts the k-mers in all sequences, and then calculates the average over all of them to provide a single distribution of k-mer frequencies. Additionally the program is capable of multi-processing over different values of k, such that 2-,3-,4-,...-mers can be calculated simultaneously.

Parameter	Function
–fasta	Path to fasta file
–name	Output file name
–dir	Output directory
-k	Comma delimited list of values of k *e.g.* 2,3,4,5
-a	Alphabet to use *e.g.* ATCG
-n	Number of processors to use

Table 3.7: Parameters for kmers.py

The program counts k-mers as defined in Algorithm 1. The following python code reads in the parameters defined above. The provided fasta file is converted into a singular string with a delimiter character $, and the total length of the string excluding the delimiter is calculated. Finally, the script kmers.pyx, which contains the implementation of Algorithm 1, is spooled onto the number of processors passed, and the results are compiled into a dictionary.

```
> python kmers.py --fasta ./fastaFiles/gencode.vM17.lncRNA_transcripts.fa -k 2,3,4 --
name mm10Trscpts -n 3
```

The results are then saved into a binary file containing a python dictionary containing k-mer frequencies for each value of k specified in the arguments.

```
# Read in specified values of k, and the alphabet
kVals = [int(i) for i in args.k.split(',')]
a = args.a.upper()

#SEEKR fasta reader module
F = Reader(args.fasta)
fS = F.get_seqs()

#Join sequences together using $ delimiter character
fString = '$'.join(fS).upper().strip()
lenFString = sum([len(i) for i in fS if '$' not in i])

# Need to figure out how to deal with very long fasta files (~ 2-3X the
    size of the transcriptome in mice)
# if lenFString >= 2147483647:
#     fString='$'.join(fS[::10]).upper()

#Split jobs onto processors and call kmers.pyx cython file
with pool.Pool(args.n) as multiN:
    jobs = multiN.starmap(kmers.main,product(*[[fString],kVals,[a]]))
    dataDict = dict(jobs)

#Save data
kDir = args.dir
if not kDir.endswith('/'):
    kDir+='/'
pickle.dump(dataDict,open(f'{kDir}{args.name}.skr','wb'))
```

HMM Creation

To create the HMM files necessary for running the mSEEKR.py program, the train.py defines the matrices A, E, π that are defined in Algorithms 5, 7, 8, and saves them in a single binary file. The emission probabilities are equivalent to the k-mer frequencies calculated from kmers.py. Due to restrictions of available training data for our own experiments, the user can manually provide the self-transition parameters for the query and null hideen states.

If training sequences are available, the –bw flag can be passed to run the Baum-Welch algorithm to obtain an MLE for the transition parameters. If the –bw flag is passed, the –iter argument must be passed. The results from the –bw are then passed directly into the main train.py program. To better check the results of a Baum-Welch operation it is recommended to run the bw.py program separately with a variety of initial parameterizations. This allows for inspection of whether local minima or a potential global maximum have been reached. The results from the previous experiment can then be manually passed to train.py –qT and –nT arguments.

Parameter	Function
–query	Path to kmer counts of query
–null	Path to kmer counts of null
–qT	Query→Query transition probability
–nT	Null→Null transition probability
–qPrefix	Name of query for file path
–nPrefix	Name of null for file path
–dir	Directory to put HMM model
-k	Comma delimited list of values of k *e.g.* 2,3,4,5
-a	Alphabet to use *e.g.* ATCG

Table 3.8: train.py parameters

hmmSEEKR creates directories for the save files in a predefined fashion to ensure proper retrieval of the correct matrices when running mSEEKR.py. The following code checks to see if the directory specified in –dir exists, and if not, creates the directory.

```
   > python train.py --query ./counts/mouseA.skr --null ./counts/mm10Trscpts.skr -k 2,3,4
 --qPrefix mouseA --nPrefix mm10Trscpts --qT .9999 --nT .9999 --dir ./markovModels/
```

```
if not args.dir.endswith('/'):
    args.dir+='/'
newDir = f'{args.dir}{args.qPrefix}_{args.nPrefix}/'
```

```
if not os.path.exists(newDir):
    os.mkdir(newDir)
else:
    flag = True
    while flag:
        usrIN = input(f'Directory {newDir} exists, continue? y/n: ').strip
            ().lower()
        if usrIN == 'y':
            flag = False
        elif usrIN == 'n':
            print('Initiating self-destruct sequence')
            sys.exit()
        else:
            print('Please enter y or n')
```

The script then loads the k-mer counts specified in –query and –null and passes and begins looping through the values of k specified in the -k argument. The following code loops through each value of k, loads the k-mer frequencies, and creates the matrices A, E, π in the form of a python dictionary. The 2-state dimensions of this HMM are hard-coded into the script.

```
 Load k-mer counts
qCount = pickle.load(open(args.query,'rb'))
nCount = pickle.load(open(args.null,'rb'))

# Loop through specified values of k
# Check if they exist in the counts file,
# and call corefunctions.HMM to generate the HMM matrices
for k in kVals:
    if (k in qCount.keys()) and (k in nCount.keys()):
        qKCount = qCount[k]
        nKCount = nCount[k]
        kDir = newDir+f'{k}/'
        if not os.path.exists(kDir):
            os.mkdir(kDir)
        A,E,states,pi = corefunctions.HMM(qKCount,nKCount,k,args.a,args.qT
            ,args.nT)
        kmers = [''.join(p) for p in itertools.product(alphabet,repeat=k)]
        # queryMkv = corefunctions.transitionMatrix(qKCount,k,alphabet)
        # nullMkv = corefunctions.transitionMatrix(nKCount,k,alphabet)
        # lgTbl = corefunctions.logLTbl(queryMkv,nullMkv)
    else:
        print(f'Missing {k}-mer counts in count file... skipping')

    # np.savetxt(f'{kDir}logtbl.mkv',lgTbl)
    pickle.dump({'A':A,'E':E,'pi':pi,'states':states},open(f'{kDir}hmm.mkv
        ','wb'))
```

The resulting python dictionaries are then saved into a binary file in a dictionary within the directory matching the following pattern:

```
--dir/--qPrefix_--nPrefix/-k/hmm.mkv
```

4.4 Viterbi Parsing

The primary script within the *hmmSEEKR* is mSEEKR.py. This script takes the matrices A, E, π created from train.py and scans a sequence, or set of sequences, and calculates the most likely parse, or Viterbi path, through each sequence. Therefore a sequence, *e.g.* ATCG, is converted into an equal length string of state lebels, *e.g.* - - + -. The program then extracts consecutively + (query) labeled nucleotides, and reports each such region as a *hit*.

```
> python mSEEKR.py --db ./fastaFiles/mm10kcn.fa -n 8 --prefix test --model ./markovModels
/mouseA_mm10Trscpts -k 3 --fasta
```

Parameter	Function
–model	Path to .mkv file output from train.py or bw.py
-k	Length of short motif (k-mer)
–db	Sequence database to use, currently accepts FASTA format
–prefix	Output file name prefix
-a	Alphabet to use *e.g.* ATCG
-n	Number of processors to use (default=1)
–fasta	Include sequence in output

Table 3.9: mSEEKR.py parameters

The following code section from mSEEKR.py loads in the specified HMM model from the –model argument, and ensures that the path has been provided in the correct format, *i.e.* ending with a "/".

```
if not model.endswith('/'):
    model +='/'

kDir = model+f'{args.k}'+'/'
modelName = model.split('/')[-2]
# Check if file exists and open if so, else skip this iteration of the
   loop

hmm = pickle.load(open(kDir+'hmm.mkv','rb'))


# Explicitly determine k from the size of the log matrix and the size of
   the alphabet used to generate it
k = int(log(len(hmm['E']['+'].keys()),len(args.a)))
kmers = [''.join(p) for p in product(alphabet,repeat=k)] # generate k-mers
target = Reader(args.db)
targetSeqs,targetHeaders = target.get_seqs(),target.get_headers()
targetMap = defaultdict(list)
```

The sequences to be parsed are then spooled onto the number of threads provided in the -n argument, and that is executed by this code:

```
#Pool processes onto number of CPU cores specified by the user
with pool.Pool(args.n) as multiN:
    jobs = multiN.starmap(hmmCalc,product(*[list(zip(targetHeaders,
        targetSeqs))]))
    dataDict = dict(jobs)
#Check if no hits were found
# if not all(v == None for v in dataDict.values()):
```

Each sequence is then prepared for input into the Viterbi algorithm. First, the sequences are passed to the function `kmersWithAmbigIndex`, which implements the pseudo-code outlined in Algorithm 2. The next step is the calculation of the Viterbi parse, as outlined in Algorithm 5.

```
''' hmmCalc
Run several functions including viterbi algorithm, log-likelihood, and
   generate output dataframes
Input: fasta file information
Output: dataframe object
'''
def hmmCalc(data):
    tHead,tSeq = data
    O,oIdx,nBP = corefunctions.kmersWithAmbigIndex(tSeq,k)
    A,E,states,pi= hmm['A'],hmm['E'],hmm['states'],hmm['pi']
    bTrack = corefunctions.viterbi(O,A,E,states,pi)
    #Zip the indices of unambig k-mers with their viterbi derived HMM
       state labels
    coordBTrack = list(zip(oIdx,bTrack)) # [(1,'-'),(2,'+',...(n,'+'))]
    mergedTrack = coordBTrack + nBP # add back in ambig locations
    mergedTrack.sort(key=itemgetter(0)) # sort master list by index
    hmmTrack = [i[1] for i in mergedTrack] # fetch just state label from
       mergedTrack ['-','+',...,'+']
    groupedHits = corefunctions.groupHMM(hmmTrack) #
       ['-----','++++++++++','-','++++','------------']

    # Return sequences of HMM hits, and their start and end locations in
       the original sequence
    seqHits,starts,ends = corefunctions.formatHits(groupedHits,k,tSeq)
    if (seqHits):
        df = corefunctions.hitOutput(seqHits,starts,ends,k,E,tHead,tSeq)
        return tHead,df
```

The parsed sequences are then combined with any ambiguous sequence regions from the original supplied sequence such that the original sequence is reconstructed, but with hidden state labels instead of nucleotides.

Finally the regions that have been designated as "hits" are compiled in a dataframe within the `formatHits` and `hitOutput` functions. Additionally, this is where the HMM score is calculated as in Algorithm 4.

4.5 bw.py - Transition parameter optimization

A key feature of *hmmSEEKR* is that it searches target transcripts for a very specific query that is defined *a priori*. *hmmSEEKR* obtains the query k-mer frequency estimates using the maximum likelihood estimate plus a pseudocount from the training sequences. The transition parameters, however, are less well defined or have no clear calculation from the data. Therefore, we opted to use an iterative expectation-maximization scheme while keeping the emission parameters, *i.e.* the k-mer frequencies, fixed.

To do this, a custom implementation of the Baum-Welch algorithm was written as part of the *hmmSEEKR* package. The algorithm works by alternating between the expectation of $P(Y|X,\theta)$, and then updating the parameters in a way that is guaranteed to increase the likelihood of the model, relative to the previous set of parameters, up to a local maximum.

Within the context of an HMM and the transition parameters, this corresponds to calculating the expected number of observations belonging to each hidden state i in the training sequence, as well as the expected number of hidden state transitions $i \rightarrow j$, where j can be any allowable transition from the transition matrix A. From here, these expectations are used to update the parameters θ. This is done by taking the partial derivative of the complete log-likelihood function with respect to each parameter and solving for 0. These update rules are outlined in the introduction as well as in algorithm 9.

Parameter	Function
–prior	Path to binary .mkv file output from train.py
-k	Length of short motif (k-mer)
–db	Sequence database to use, currently accepts FASTA format
–createfile	Create new file rather than overwrite
–its	Number of iterations to run BW algorithm

Table 3.10: bw.py parameters

Example usage:

```
> python bw.py -k 4 --db ./fastaFiles/xist.fa --prior markovModels/mouseA_mm10Trscpts
/4/hmm.mkv --its 3 -cf
```

The algorithm is implemented in python as in algorithms 7, 8, 9. Initially, the bw.py script reads in an initially trained model that contains guesses for the parameters α and β. The script then checks the sequence for any ambiguous nucleotides, and if any exist, the sequence is split

into contiguous chunks of unambiguous sequence. From here, a dictionary is created that will track the values of the parameters α and β over each iteration of the BW algorithm.

```
fa = Reader(args.db)
seqs = fa.get_seqs()[0]
model = args.prior

k = args.k

# Identify the location of any ambiguous nucleotides (N)
O,oIdx,nBP = corefunctions.kmersWithAmbigIndex(seqs,k)
# Load in train.py output
hmm = pickle.load(open(args.prior,'rb'))

'''
A - transition matrix (dictionary)
E - emission matrix (dictionary)
pi - initial state probabilities (always 50/50)
states - list of states (query,null)
'''
A,E,pi,states = hmm['A'],hmm['E'],hmm['pi'],hmm['states']

data = defaultdict(list)
data['alpha'].append(A['+']['+'])
data['beta'].append(A['-']['-'])
```

Following this, the main loop of the BW algorithm begins. First, the forward and backward probabilities are calculated for the entire training sequence. The updated parameters α and β are then calculated. Last, these parameters are normalized to sum to one (to be a proper probability distribution), and the loop continues until the specified number of iterations is completed.

```
for i in tqdm(range(args.its)):
    a = corefunctions.fwd(O,A,pi,states,E) # forward probabilities
    b = corefunctions.bkw(O,A,pi,states,E) # backwards probabilities
    A = corefunctions.update(a,b,O,states,A,E) # update parameter values

    # The above step yields probabilities that are not yet normalized (do
       not sum to 1)
    # That is, for prior state "i" (can be either query or null)
       transitioning to either query (+) or null (-): p(-|i) + p(+|j) != 1
    # Here, we calculate the marginal probability for transitioning from
       state i to either state i or j (normalizing constant)
    # Then, divide each probability p(-|i) and p(+|i) by the marginal
       probability
    # Now, p(-|i) + p(+|i) = 1
    for i in states:
        # marginal probability (normalizing constant)
        marg = logsumexp(list(A[i].values())) # cannot simply sum in log-
           space, logsumexp to sum log-space numbers without over/under-
           flow (see wiki page)
        A[i]['+']-=marg # log-space, so this represents the division by
           the normalizing constant
        A[i]['-']-=marg # as above
    data['alpha'].append(A['+']['+'])
    data['beta'].append(A['-']['-'])
```

From here, two files are saved. One contains the values of α and β over each iteration of the BW algorithm. This file should be checked for convergence of the parameters. Second, a new binary file containing the updated HMM is saved, and can be passed to mSEEKR.py.

```
# Data dictionary tracks the iterations of BW algorithm for manual
    inspection if necessary

arr = np.array(list(data.values()))
arr = 2**arr
arr = arr.T
np.savetxt(os.path.dirname(args.prior) +'/hmm_BWparameters.txt',arr,fmt='
    %.9f')
if args.createfile:
    bn = os.path.basename(args.prior)
    bn = bn.split('.')[0]
    bn+='_MLE'
    bn = os.path.dirname(args.prior) +'/'+ bn
    pickle.dump({'A':A,'E':E,'pi':pi,'states':states},open(f'{bn}.mkv','wb
        '))
elif not args.createfile:
    pickle.dump({'A':A,'E':E,'pi':pi,'states':states},open(f'{args.prior}'
        ,'wb'))
```

4.6 Xist Enriched Proteins

All ENCODE eCLIP Experiments (non-controls) in K562 cells were selected and downloaded from the ENCODE database (bigWig file type).

`xargs -L 1 curl -O -L < encodefileurls.txt\\for i in *bigWig; do ./bigWigToBedGraph` i`i.bedGraph;done`

We then calculated the ratio of average eCLIP signal within Xist Repeats to the average signal outside repeats.

We defined X as a set of size $L = 19275$, where each entry corresponds to a basepair in *XIST*. Each nucleotide within this set, x_i, was given a value depending on whether or not eCLIP signal was present from the bigWig files retrieved from ENCODE. If there was no signal, a value of 0 was given to x_i, otherwise the read density from the bigWig file was assigned to x_i.

We calculated the eCLIP signal enrichment of each protein within each *XIST* repeat by using the transcript coordinates of each repeat in *XIST* to define a set I, which contains all individual nucleotide positions within a given *XIST* repeat. Set I^c is the complement of I, relative to X, and contains all basepairs in *XIST* that are not in I.

If N is the size of I, *i.e.* the size of the given repeat in *XIST*, then the ratio of average signal R is:

$$R = \frac{L - N}{N} \frac{\sum_{i \in I} X_i}{\sum_{j \in I^c} X_j}$$

4.7 *Rsx* HMM Analysis

Human *XIST* repeats A,B,D,E were used as queries within 4 separate *hmmSEEKR* models. For the null model, human unspliced lncRNAs (GENCODE V26) were used. A model was trained for each value of $k \in \{2, 3, 4, 5, 6\}$ and for the transition parameters all combinations of the following values were tested, $\alpha, \beta \in \{.5, .75, .9, .99, .999, .9999\}$. The koala *Rsx* transcript [38] was then scanned using each of these HMMs. For measuring performance of the HMM, precision was defined as the number of nucleotides from the HMM Viterbi parse that fell within the correct *Rsx* repeat as found in [2]. Recall was defined as the percentage of each *Rsx* repeat recovered from the associated *XIST* trained HMM. The F_1 score was then calculated as

$$F_1 = \left(\frac{2}{\texttt{precision}^{-1} + \texttt{recall}^{-1}} \right) \tag{3.4}$$

And the set of parameters for each *XIST* query with the highest F_1 score was chosen (Table 3.11.

Query	k	α	β
Repeat A	4	.9999	.9999
Repeat B	4	.9999	.9999
Repeat D	2	.9999	.9999
Repeat E	4	.9999	.9999

Table 3.11: Best values of k, the $+ \rightarrow +$ parameter α, and $- \rightarrow -$ parameter β as determined through a grid search based approach to identify sequences with the highest SEEKR correlation to the query.

4.8 *KCNQ1OT1* Analysis

HMM Training

HMMs were trained on each of the human *XIST* A-,B-,D-, and E- repeats [6] with parameters $\theta = (k, \alpha, \beta)$. The A-repeat had parameters $\theta_A = (4, .9999, .9999)$, B-repeat $\theta_B = (4, .9999, .9999)$, D-repeat $\theta_D = (2, .9999, .9999)$, E-repeat $\theta_E = (3, .9999, .9999)$. The unspliced human KCNQ1OT1 gene (GENCODE V26) was provided as the "target" sequence to be parsed.

Randomization

Randomized parses were generated for each query's results by taking each hit i after scanning *KCNQ1OT1* and generating a random point between $[1, L - l_i + 1]$ (where l_i is the length of hit i). The hit was then "moved" to this new location within the transcript. This process was repeated for each hit.

eCLIP Read overlap

To calculate eCLIP signal, we used featureCounts (`-s 2 -F SAF -p` options) to overlap the reads from each proteins BAM file with the genomic coordinates of the HMM hits, and randomized parses.

4.9 Transcriptome Search

The eCLIP files from the RBPs in Table 3.3 were sourced from ENCODE, with the BAM files corresponding to the GRCH38 build of the human genome pulled from the following experiments:

- RBM15 - ENCFF739LLZ
- SRSF1 - ENCFF696TJG
- HNRNPK - ENCFF894NKS
- TIA1 - ENCFF080VML
- MATR3 - ENCFF162SAS
- PTBP1 - ENCFF765BPN
- RBM22 - ENCFF045LAO
- ZC3H8 - ENCFF409GUM
- UTP3 - ENCFF826UUS

ENCODE chromatin association RNA-seq data (experiment ID: ENCFF957VGD) was used to identify expressed transcripts that are enriched around chromatin. The following programs

were then used to identify these expressed transcripts, parse sequences of interest, and overlap the reads from the BAM files above with the HMM parses:

- mSEEKR/v1.0.8
- RSEM/1.2.31
- deeptools/3.0.0
- subread/2.0.0

The unspliced/spliced ratio for each transcript was done using RSEM. A fasta file with transcript isoforms of interest was generated. For this analysis, a single spliced isoform (primary) and the unspliced transcript were included in the fasta file from the GENCODE V26 annotation gff file. The following commands were then run:

```
module add rsem/1.2.31
module add samtools
module add deeptools/2.5.2
module add bowtie/1.2.2
sbatch --mem 50g -t 5:00:00 -N 1 -n 24 --wrap='bowtie-build --threads 24 v26_combo.fa hgT'
sbatch --mem 50g --wrap='rsem-prepare-reference . hgT'
```

After the reference was built for RSEM, the `rsem-calculate-expression` program was called to quantify the TPM of the unspliced and spliced isoforms of each transcript, where EXPNAME was replaced with the name of the ENCODE experiment being quantified.

```
sbatch -N 1 --mem 50g -t 5:00:00 -J rsemChr -n 24 --wrap='rsem-calculate-expression -p 24 --paired-end READS1.fastq READS2.fastq hgT EXPNAME'
```

Following this, transcripts with unspliced isoforms above a threshold TPM ($> \log_2$ TPM 0) were used as input into *hmmSEEKR*. Parameterizations for the A-,B-,D-, and E-repeat HMMs were as in Table 3.5, with the null models trained on unspliced human lncRNAs (GENCODE V26).

A SAF (Simple Annotation Format) file was then created from the *hmmSEEKR* output. A SAF file is similar to BED6 input with re-arranged columns. We then used feature counts to

quantify the read overlaps between the *hmmSEEKR* SAF file and the ENCODE BAM files for each protein.

In tandem with the real *hmmSEEKR* parse, a randomized parse for each transcript was created (as in the KCNQ1OT1 analysis). For each hit within a transcript a random point was chosen with uniform probability within the transcript from position 1, to the final coordinate less the length of the hit. This randomly chosen point was then designated as the start point for the random hit, with length equal to the original hit from *hmmSEEKR*. From here, a SAF file was created for all randomized hits and overlapped with the ENCODE eCLIP data.

We then calculated the total number of reads retrieved by the true HMM and the randomized parses, as well as the average number of reads per hit. Statistics were calculated using a chi-square test for total reads retrieved, with a null hypothesis of 1:1 ratio between the true HMM and the randomized parses, and a student's t-test of the average number of reads per hit with a null hypothesis of no difference between HMM and random.

REFERENCES

[1] N. Brockdorff, "Local tandem repeat expansion in Xist RNA as a model for the functionalisation of ncRNA," *Non-coding RNA*, 2018, ISSN: 2311553X.

[2] D. Sprague, S. A. Waters, J. M. Kirk, J. R. Wang, P. B. Samollow, P. D. Waters, and J. M. Calabrese, "Nonlinear sequence similarity between the Xist and Rsx long noncoding RNAs suggests shared functions of tandem repeat domains," *RNA*, 2019, ISSN: 14699001.

[3] G. Pintacuda, G. Wei, C. Roustan, B. A. Kirmizitas, N. Solcan, A. Cerase, A. Castello, S. Mohammed, B. Moindrot, T. B. Nesterova, and N. Brockdorff, "hnRNPK Recruits PCGF3/5-PRC1 to the Xist RNA B-Repeat to Establish Polycomb-Mediated Chromosomal Silencing," *Molecular Cell*, 2017, ISSN: 10974164.

[4] X. Wang, K. J. Goodrich, A. R. Gooding, H. Naeem, S. Archer, R. D. Paucek, D. T. Youmans, T. R. Cech, and C. Davidovich, "Targeting of Polycomb Repressive Complex 2 to RNA by Short Repeats of Consecutive Guanines," *Molecular Cell*, 2017, ISSN: 10974164.

[5] J. Zhao, B. K. Sun, J. A. Erwin, J. J. Song, and J. T. Lee, "Polycomb proteins targeted by a short repeat RNA to the mouse X chromosome," *Science*, 2008, ISSN: 00368075.

[6] C. J. Brown, "The human XIST gene: analysis of a 17 kb inactive X-specific RNA that contains conserved repeats and is highly localized within the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 527–542, 1992 10, ISSN: 0092-8674.

[7] N. Brockdorff, "The product of the mouse Xist gene is a 15 kb inactive X-specific transcript containing no conserved ORF and located in the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 515–526, 1992 10, ISSN: 0092-8674.

[8] J. Grant, S. K. Mahadevaiah, P. Khil, M. N. Sangrithi, H. Royo, J. Duckworth, J. R. McCarrey, J. L. Vandeberg, M. B. Renfree, W. Taylor, G. Elgar, R. D. Camerini-Otero, M. J. Gilchrist, and J. M. Turner, "Rsx is a metatherian RNA with Xist-like properties in X-chromosome inactivation," *Nature*, 2012, ISSN: 00280836.

[9] Y. Hoki, N. Kimura, M. Kanbayashi, Y. Amakawa, T. Ohhata, H. Sasaki, and T. Sado, "A proximal conserved repeat in the Xist gene is essential as a genomic element for X-inactivation in mouse," *Development*, 2009, ISSN: 09501991.

[10] B. Moindrot, A. Cerase, H. Coker, O. Masui, A. Grijzenhout, G. Pintacuda, L. Schermelleh, T. B. Nesterova, and N. Brockdorff, "A Pooled shRNA Screen Identifies Rbm15, Spen, and Wtap as Factors Required for Xist RNA-Mediated Silencing," *Cell Reports*, 2015, ISSN: 22111247.

[11] H. Sunwoo, D. Colognori, J. E. Froberg, Y. Jeon, and J. T. Lee, "Repeat E anchors Xist RNA to the inactive X chromosomal compartment through CDKN1A-interacting protein (CIZ1)," *Proceedings of the National Academy of Sciences of the United States of America*, 2017, ISSN: 10916490.

[12] T. B. Nesterova, S. Y. Slobodyanyuk, E. A. Elisaphenko, A. I. Shevchenko, C. Johnston, M. E. Pavlova, I. B. Rogozin, N. N. Kolesnikov, N. Brockdorff, and S. M. Zakian, *Characterization of the genomic Xist locus in rodents reveals conservation of overall gene structure and tandem repeats but rapid evolution of unique sequence*, 2001.

[13] M. E. Royce-Tolland, A. A. Andersen, H. R. Koyfman, D. J. Talbot, A. Wutz, I. D. Tonks, G. F. Kay, and B. Panning, "The A-repeat links ASF/SF2-dependent Xist RNA processing with random choice during X inactivation," *Nature Structural and Molecular Biology*, 2010, ISSN: 15459993.

[14] J. M. Kirk, S. O. Kim, K. Inoue, M. J. Smola, D. M. Lee, M. D. Schertzer, J. S. Wooten, A. R. Baker, D. Sprague, D. W. Collins, C. R. Horning, S. Wang, Q. Chen, K. M. Weeks, P. J. Mucha, and J. M. Calabrese, "Functional classification of long non-coding RNAs by k-mer content," *Nature Genetics*, 2018, ISSN: 15461718.

[15] M. Jiang, J. Anderson, J. Gillespie, and M. Mayne, "uShuffle: A useful tool for shuffling biological sequences while preserving the k-let counts," *BMC Bioinformatics*, 2008, ISSN: 14712105.

[16] K. C. Pang, M. C. Frith, and J. S. Mattick, *Rapid evolution of noncoding RNAs: Lack of conservation does not mean lack of function*, 2006.

[17] H. Hezroni, D. Koppstein, M. G. Schwartz, A. Avrutin, D. P. Bartel, and I. Ulitsky, "Principles of Long Noncoding RNA Evolution Derived from Direct Comparison of Transcriptomes in 17 Species," *Cell Reports*, 2015, ISSN: 22111247.

[18] R. Johnson and R. Guigó, "The RIDL hypothesis: Transposable elements as functional domains of long noncoding RNAs," *RNA*, 2014, ISSN: 14699001.

[19] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *Journal of Molecular Biology*, 1997, ISSN: 00222836.

[20] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, 1990, ISSN: 00222836.

[21] T. J. Wheeler and S. R. Eddy, "Nhmmer: DNA homology search with profile HMMs," *Bioinformatics*, 2013, ISSN: 14602059.

[22] P. Johnsson, L. Lipovich, D. Grandér, and K. V. Morris, *Evolutionary conservation of long non-coding RNAs; Sequence, structure, function*, 2014.

[23] P. Rice, L. Longden, and A. Bleasby, *EMBOSS: The European Molecular Biology Open Software Suite*, 2000.

[24] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 1989, ISSN: 15582256.

[25] L. Pachter, M. Alexandersson, and S. Cawley, "Applications of generalized pair Hidden Markov models to alignment and gene finding problems," *Journal of Computational Biology*, 2002, ISSN: 10665277.

[26] J. Henderson, "Finding genes in DNA with a Hidden Markov Model," *Journal of Computational Biology*, 1997, ISSN: 10665277.

[27] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles, J. Lagarde, L. Veeravalli, X. Ruan, Y. Ruan, T. Lassmann, P. Carninci, J. B. Brown, L. Lipovich, J. M. Gonzalez, M. Thomas, C. A. Davis, R. Shiekhat-

tar, T. R. Gingeras, T. J. Hubbard, C. Notredame, J. Harrow, and R. Guigó, "The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression," *Genome Research*, 2012, ISSN: 10889051.

[28] D. Cirillo, M. Blanco, A. Armaos, A. Buness, P. Avner, M. Guttman, A. Cerase, and G. G. Tartaglia, *Quantitative predictions of protein interactions with long noncoding RNAs: To the Editor*, 2016.

[29] E. L. Van Nostrand, G. A. Pratt, A. A. Shishkin, C. Gelboin-Burkhart, M. Y. Fang, B. Sundararaman, S. M. Blue, T. B. Nguyen, C. Surka, K. Elkins, R. Stanton, F. Rigo, M. Guttman, and G. W. Yeo, "Robust transcriptome-wide discovery of RNA-binding protein binding sites with enhanced CLIP (eCLIP)," *Nature Methods*, 2016, ISSN: 15487105.

[30] C. Chu, Q. C. Zhang, S. T. Da Rocha, R. A. Flynn, M. Bharadwaj, J. M. Calabrese, T. Magnuson, E. Heard, and H. Y. Chang, "Systematic discovery of Xist RNA binding proteins," *Cell*, 2015, ISSN: 10974172.

[31] D. Dominguez, P. Freese, M. S. Alexis, A. Su, M. Hochman, T. Palden, C. Bazile, N. J. Lambert, E. L. Van Nostrand, G. A. Pratt, G. W. Yeo, B. R. Graveley, and C. B. Burge, "Sequence, Structure, and Context Preferences of Human RNA Binding Proteins," *Molecular Cell*, 2018, ISSN: 10974164.

[32] D. Ray, H. Kazan, K. B. Cook, M. T. Weirauch, H. S. Najafabadi, X. Li, S. Gueroussov, M. Albu, H. Zheng, A. Yang, H. Na, M. Irimia, L. H. Matzat, R. K. Dale, S. A. Smith, C. A. Yarosh, S. M. Kelly, B. Nabet, D. Mecenas, W. Li, R. S. Laishram, M. Qiao, H. D. Lipshitz, F. Piano, A. H. Corbett, R. P. Carstens, B. J. Frey, R. A. Anderson, K. W. Lynch, L. O. Penalva, E. P. Lei, A. G. Fraser, B. J. Blencowe, Q. D. Morris, and T. R. Hughes, "A compendium of RNA-binding motifs for decoding gene regulation," *Nature*, 2013, ISSN: 00280836.

[33] D. P. Patil, C. K. Chen, B. F. Pickering, A. Chow, C. Jackson, M. Guttman, and S. R. Jaffrey, "M6 A RNA methylation promotes XIST-mediated transcriptional repression," *Nature*, 2016, ISSN: 14764687.

[34] C. Stork, Z. Li, L. Lin, and S. Zheng, "Developmental Xist induction is mediated by enhanced splicing," *Nucleic Acids Research*, 2019, ISSN: 13624962.

[35] A. Pandya-Jones, Y. Markaki, J. Serizay, T. Chitiashvilli, W. Mancia, A. Damianov, C. Chronis, B. Papp, C.-K. Chen, R. McKee, X.-J. Wang, A. Chau, H. Leonhardt, S. Zheng, M. Guttman, D. L. Black, and K. Plath, "An Xist-dependent protein assembly mediates Xist localization and gene silencing," *bioRxiv*, 2020.

[36] K. M. Seibt, T. Schmidt, and T. Heitkam, "FlexiDot: Highly customizable, ambiguity-aware dotplots for visual sequence analyses," *Bioinformatics*, 2018, ISSN: 14602059.

[37] B. C. Brookes and A. N. Kolmogorov, "Foundations of the Theory of Probability," *The Mathematical Gazette*, 1951, ISSN: 00255572.

[38] R. N. Johnson, D. O'Meally, Z. Chen, G. J. Etherington, S. Y. Ho, W. J. Nash, C. E. Grueber, Y. Cheng, C. M. Whittington, S. Dennison, E. Peel, W. Haerty, R. J. O'Neill, D. Colgan, T. L. Russell, D. E. Alquezar-Planas, V. Attenbrow, J. G. Bragg, P. A. Brandies, A. Y. Y. Chong, J. E. Deakin, F. Di Palma, Z. Duda, M. D. Eldridge, K. M. Ewart, C. J. Hogg, G. J.

Frankham, A. Georges, A. K. Gillett, M. Govendir, A. D. Greenwood, T. Hayakawa, K. M. Helgen, M. Hobbs, C. E. Holleley, T. N. Heider, E. A. Jones, A. King, D. Madden, J. A. Graves, K. M. Morris, L. E. Neaves, H. R. Patel, A. Polkinghorne, M. B. Renfree, C. Robin, R. Salinas, K. Tsangaras, P. D. Waters, S. A. Waters, B. Wright, M. R. Wilkins, P. Timms, and K. Belov, "Adaptation and conservation insights from the koala genome," *Nature Genetics*, 2018, ISSN: 15461718.

CHAPTER 4: SEEKR K-MERS

1 Introduction

K-mer counting is a seemingly relatively straightforward process. Given a sequence $X = \{x_1, \ldots, x_L\}$ of length L, for each position $i \in 1 \leq i \leq (L - k + 1)$ within the sequence, a sub-string is sliced from $[i, i + k)$. Where "[" indicates inclusive, and ")" indicates non-inclusive. A hash-map (python dictionary), then saves the count for each k-mer as encountered in X. In many applications, the frequency of each k-mer is then calculated by dividing each k-mer count by the total number of k-mers, *i.e.* $L - k + 1$.

SEEKR differs from this approach in that the algorithm adjusts for the non-uniformity of k-mers within the genome. Some k-mers are more, or less, frequent than others solely due to mononucleotide frequency variations within the genome [1, 2]. Therefore, the enrichment for already common k-mers, or depletion of rare k-mers, is not informative unless it is known that they are significantly more enriched or depleted than background. To account for this, SEEKR calculates a z-score for each k-mer rather than a frequency.

The z-score is calculated by first calculating the k-mer frequencies within some reference set of sequences, *e.g.* the transcriptome. From this reference set of k-mer frequencies, SEEKR calculates, for each k-mer "j", the mean frequency of each k-mer μ_j and the standard deviation of each k-mer σ_j.

For each k-mer j within a sequence X, X_j (the frequency of j in X), has a z-score X_j^* [3]:

$$X_j^* = \frac{X_j - \mu_j}{\sigma_j} \tag{4.1}$$

We observed that k-mer similarities between XIST and RSX that these z-scores were approximately log-normally distributed [4]. For the analysis in [4], we transformed z-scores as outlined in 2.3.4. However, we sought to perform a more rigorous analysis of different methodologies for adjusting k-mer counts. This chapter outlines the methods that we used and their results.

2 Results and *k*-mer counting methods

2.1 *k*-mer counts are log-normally distributed

We observed that a challenge when estimating the correlation between highly repetitive regions of *XIST* and other *cis*-repressive lncRNAs was that the *k*-mer counts were heavily right-skewed [4]. Pearson correlation is primarily intended to calculate the correlation between two normally distributed random variables. Spearman correlation can be used when this condition is not met, for a strictly monotonic relationship, but we found that the mode of the *k*-mer count distribution for any sequence or sub-sequence was 0, yielding artificially high correlations between any two given sequences [4].

To correct for the right skewed distribution of *k*-mers across all sequences, we first confirmed that *k*-mer counts within lncRNAs are log-normally distributed (Figure 4.1). The left panel of Figure 4.1 shows the marginal count distribution of *k*-counts summed over all possible *k*-mers, over all annotated human transcripts (Methods 4.3.1). We then log-transformed the data and observed an approximately normal distribution (Figure 4.1), with the exception being the mode of the distribution was still a count of 0. This indicated that *k*-mer frequencies amongst all transcripts were log-normally distributed and that correcting for this in the SEEKR algorithm may yield better correlation estimates amongst transcripts.

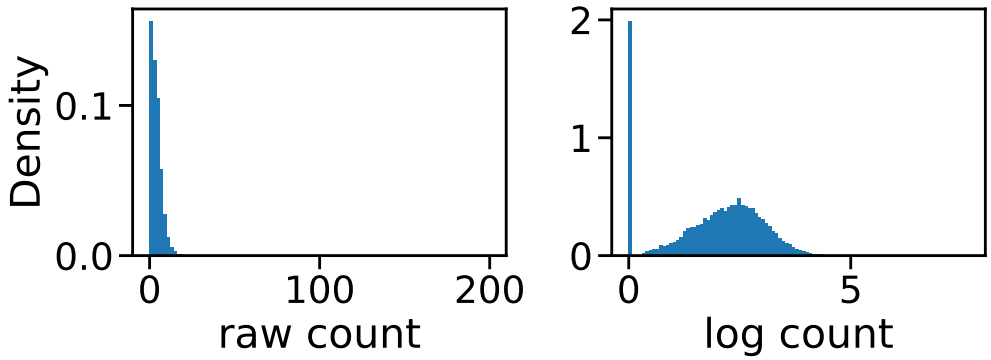

Figure 4.1: *k*-mer counts are log-normally distributed. A) The frequency of *k*-mer counts over all *k*-mers, over all annotated transcripts in the human genome. B) The distribution in (A), log-transformed.

2.2 k-mer counting methodologies

We then asked if a k-mer counting strategy could be devised that yielded more consistent estimates of correlations between two sequences, in a way that is sensible and mathematically tractable. The primary difficulty is the addition of a z-score calculation step within SEEKR, which transforms the data in a way that suppresses the influence of naturally frequent or depleted k-mers within a reference set, *e.g.* the transcriptome.

We found in [4], that the z-scores themselves were log-normally distributed, and so we log-transformed the z-scores after shifting all values by equation 4.2 [4].

$$M^*_{ij} = \log_2 \left(M_{ij} + |\min M| + 1 \right) \tag{4.2}$$

Where M is the SEEKR z-score matrix, i is the row, j is the column (M_{ij} is then the coordinates of a single data point in M), and $\min M$ is the smallest value in the matrix M. This translation of the data allows for the calculation of the logarithm. An issue with this methodology, however, is that there is no clear best choice of value to add to each data point in M, due to the log-transform. The goal, therefore, was to add the smallest possible value to each data-entry without destroying the linear relationship between row vectors. Each row within a SEEKR matrix represents the vector of k-mer frequencies within a single sequence, and the correlation step in SEEKR calculates the Pearson correlation between all pair-wise combinations of rows within M.

Below, we iterate through a series of algorithms for generating the final SEEKR matrix from which correlations will be calculated, and the resulting correlations.

Method 1. Kirk et al. 2018

The first method we outline is the original algorithm published in *Kirk et al.*. Here, we first count the occurrence of each k-mer in a sequence. This count is then normalized to a per-kB count by dividing each k-mer count by the length of the sequence and multiplying by 1000. This process is then repeated for a reference set of sequences, *e.g.* the transcriptome, and the mean frequency and standard deviation for each k-mer over the reference set are calculated. From there, each frequency count in the SEEKR frequency matrix is transformed using Equation 4.1

[3].

	MXA	MXB	MXC	HXD	MXE
KR1	-0.048	0.266	0.001	0.146	-0.117
KR2	0.085	-0.089	-0.111	-0.024	0.147
KR3	-0.041	-0.119	-0.009	-0.021	0.267
KR4	0.180	-0.114	-0.081	0.087	0.294

Table 4.1: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated as in [3], 2018.

From here, the pair-wise correlations of each sequence's k-mer profile, *i.e.* the row vector of k-mer z-scores, are calculated. For each of the methods outlined in this chapter, the tandem repeats of *XIST* were compared against the tandem repeats of *Rsx* as in [4]. The correlations for this algorithm are shown in Table 4.1.

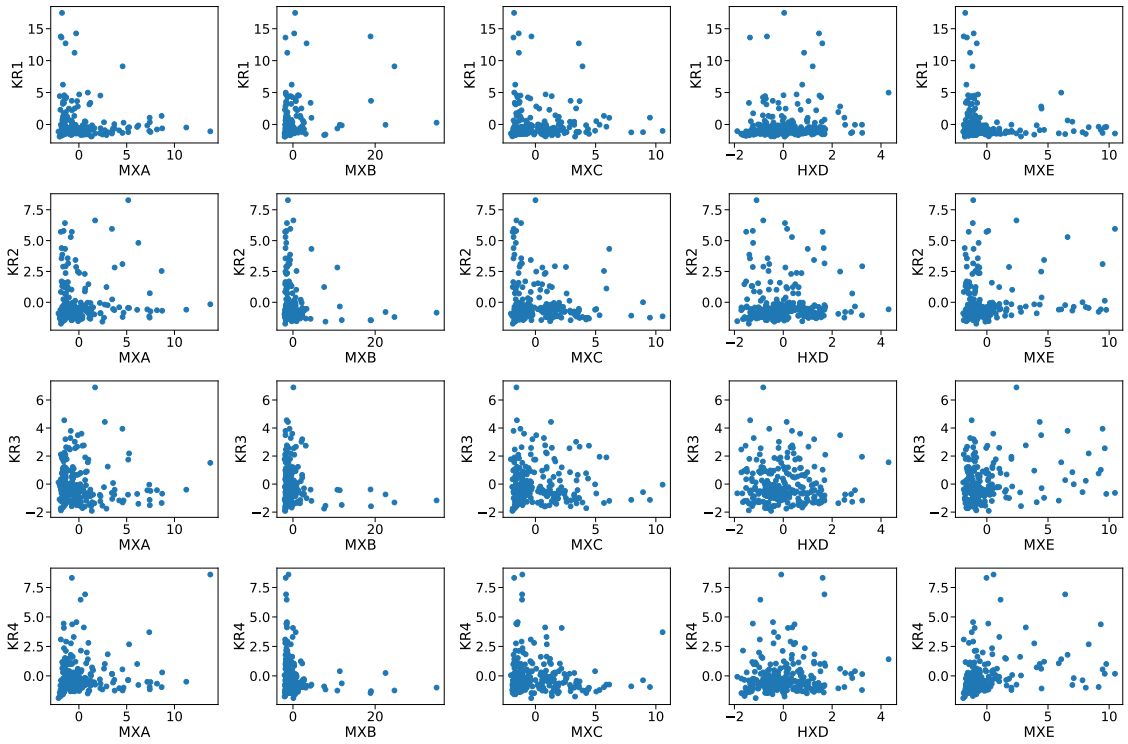

Figure 4.2: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 1. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

	MXA	MBX	MXC	HXD	MXE
KR1	-0.021	0.325	0.038	0.193	-0.163
KR2	0.084	-0.122	-0.122	-0.013	0.145
KR3	-0.077	-0.093	0.014	-0.023	0.249
KR4	0.210	-0.114	-0.068	0.112	0.398

Table 4.2: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated as in [4], 2019.

Method 2. Sprague et al. 2019

The second method outlined is the algorithm published in [4]. The k-mers are counted exactly as in [3], however we additionally calculated the log_2 of the z-scores using Equation 4.2. Tandem repeat comparisons that were highly correlated as in Method 1 saw large increases in correlation using this method, whereas uncorrelated regions saw little change (Table 4.2).

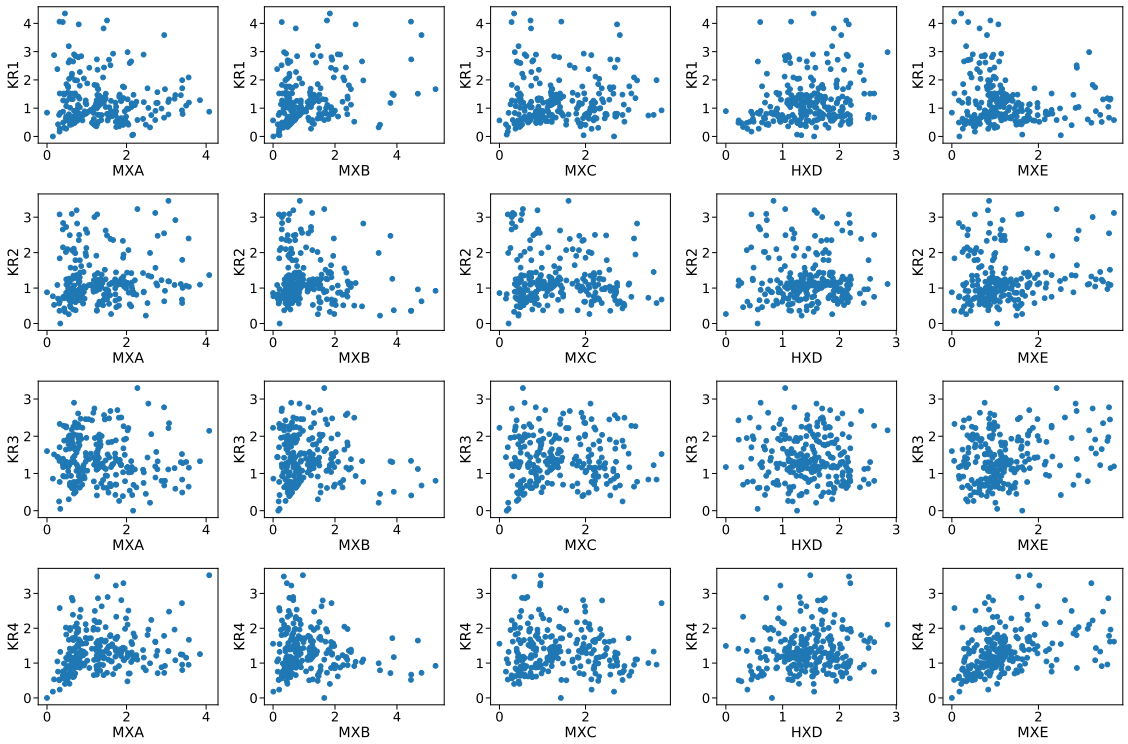

Figure 4.3: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 2. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

Method 3. Row-wise addition to z-scores

A concern when translating data, as in Equation 4.2, is the unequal effect it has when calculating the log of the data. Typically, Pearson correlation is translation invariant, *i.e.* the addition of any constant to all elements of a vector does not affect its correlation with another vector. This is because the calculation of the Pearson correlation involves 0-mean centering each vector, which negates the addition or multiplication of any constant. Another intuition for this is that moving a scatter plot around on a graph changes the location of the data points, but it doesn't change the line of best fit between the data.

	MXA	MXB	MXC	HXD	MXE
KR1	-0.017	0.327	0.042	0.197	-0.163
KR2	0.088	-0.124	-0.120	-0.009	0.147
KR3	-0.079	-0.090	0.016	-0.022	0.246
KR4	0.214	-0.110	-0.066	0.115	0.404

Table 4.3: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated as in *Sprague et al.*, 2018., except that instead of the minimum of the z-score matrix being added, the minimum of each row was added to all values within the row.

However, the inclusion of a log-transformation step alters the linear relationship of the data depending on how far the data point is from 0. Therefore, it is ideal to add the smallest possible constant to each entry in the matrix. *Sprague et al.* added the minimum of the entire matrix to each z-score, however adding the minimum of each row to all values within that row allows for the addition of the smallest possible value such that the log can be taken, without altering the relationship between the k-mer counts within a given sequence. The results were similar to those in Method 2, except with further increases in correlation between high correlated domains such as KR4-MXE and KR1-MXB (Table 4.3).

Method 4: Column-wise addition to z-scores

To illustrate the point that the value added to the SEEKR matrix must be constant within a row, here we repeat what was done in Method 3, but instead of adding the minimum of each row to all values within that row, we take the minimum of each column and add that value to each element within the column.

As an example, consider the hypothetical SEEKR z-score matrix in Table 4.5. After applying

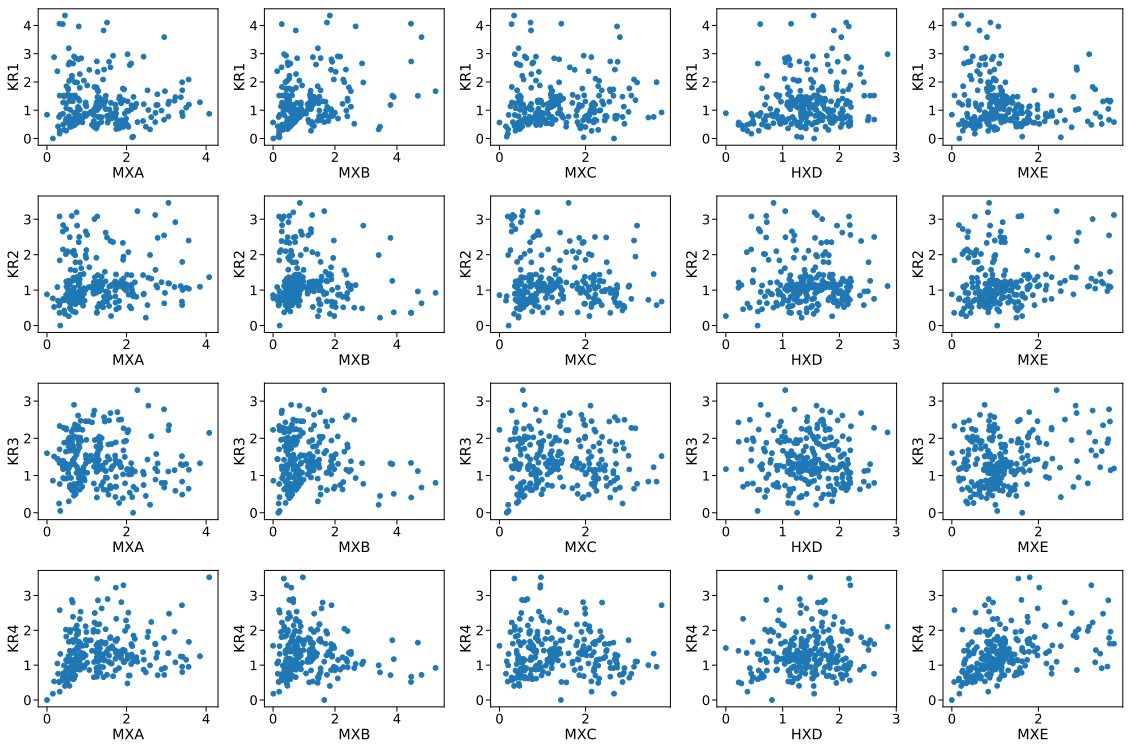

Figure 4.4: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 3. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

the transformation described in this section, all values are greater than 0 which allows for the log-transform of the data (Table 4.6. However, the k-mer frequencies within each sequence have become jumbled. Consider the relationship between A and T in seq1. Prior to the transformation, T was clearly depleted relative to A in seq1, with z-scores of -2 and 0 respectively. After the transformation, the SEEKR matrix is now indicating that A and T are equally frequent in seq1. Likewise, G was depleted relative to A in both seq1 and seq2, but after the operation is now enriched relative to A. This operation therefore does not make sense, as we are only trying to move the data to be positive, such that we can calculate the log. While the correlations look promising in Table 4.4, the scatter plots of the k-mer counts show that significant aberration has been introduced to the data relative to Methods 1,2 and 3 (Figure 4.5).

Method 5: Pseudo-count to raw counts

A more natural transformation of the data is to take the logarithm of the raw count data, which is guaranteed to be ≥ 0. A potential downside with this approach is that adding 1 to raw

	MXA	MXB	MXC	HXD	MXE
KR1	-0.048	0.348	0.093	0.284	-0.098
KR2	0.057	-0.046	0.034	0.201	0.261
KR3	-0.061	-0.015	0.144	0.173	0.347
KR4	0.147	-0.094	0.044	0.241	0.451

Table 4.4: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated as in *Sprague et al.*, 2018., except that instead of the minimum of the z-score matrix being added, the minimum of each column was added to all values within the column.

	A	T	C	G
seq1	0	-2	5	-3
seq2	0	1	3	-2

Table 4.5: A hypothetical SEEKR z-score matrix.

	A	T	C	G
seq1	1	1	7	1
seq2	1	4	5	2

Table 4.6: The same matrix as in Table 4.5 but transformed by adding the minimum of each column to all values within that column, then adding 1 such that all values are greater than 0.

counts has a differential effect on sequences of different lengths, *i.e.* adding +1 to the count of all k-mers in a sequence that is 200bp long has a larger effect on the frequencies than a 10,000bp sequence. Therefore, we tried two different approaches. In Method 5, we added +1 to the raw counts of the data, length normalized the counts, and then took the log-transform, followed by z-score calculation as in *Kirk et al.* (Table 4.7).

Method 6: Length normalized pseudo-count

The second method was to add 1 after length normalizing the k-mer counts per kB, effectively making the pseudo-count equivalent between transcripts of different lengths, and then going through SEEKR as before. The results are shown in Table 4.8. This latter method of adding a pseudo-count posterior to length normalization is the method we chose to continue using with SEEKR going forward, as the mathematics are straigt forward and are best suited to the data at hand, as well as yielding strong results between regions known to share similar function and k-mer frequencies (Figure 4.7).

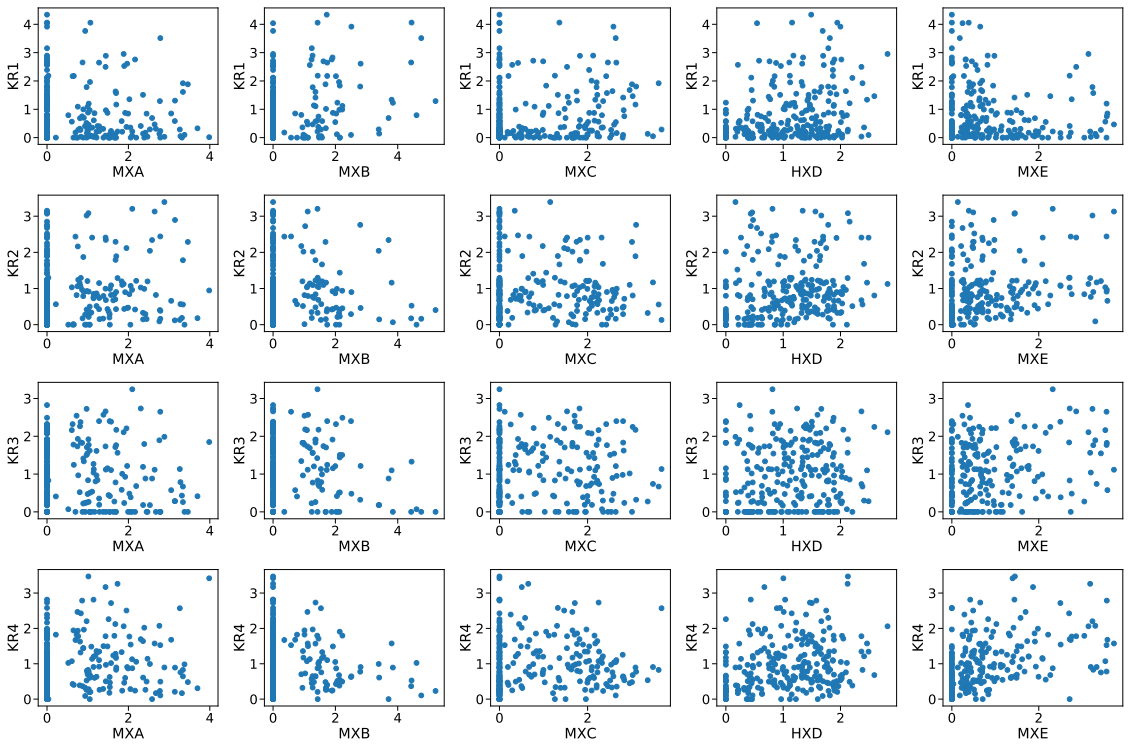

Figure 4.5: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 4. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

	MXA	MXB	MXC	HXD	MXE
KR1	-0.068	0.232	0.024	0.181	-0.179
KR2	-0.074	-0.295	-0.087	0.071	0.126
KR3	-0.170	-0.170	0.063	0.001	0.205
KR4	0.119	-0.175	-0.014	0.148	0.387

Table 4.7: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated adding a pseudo-count to the raw count data (Method 5).

	MXA	MXB	MXC	HXD	MXE
KR1	-0.013	0.335	0.059	0.200	-0.163
KR2	0.039	-0.152	-0.080	-0.007	0.135
KR3	-0.102	-0.067	0.076	-0.025	0.220
KR4	0.249	-0.078	0.015	0.150	0.431

Table 4.8: SEEKR Pearson correlations between the tandem repeats of *Xist* and *Rsx*. Correlations were calculated using a length normalized pseudo-count (Method 6).

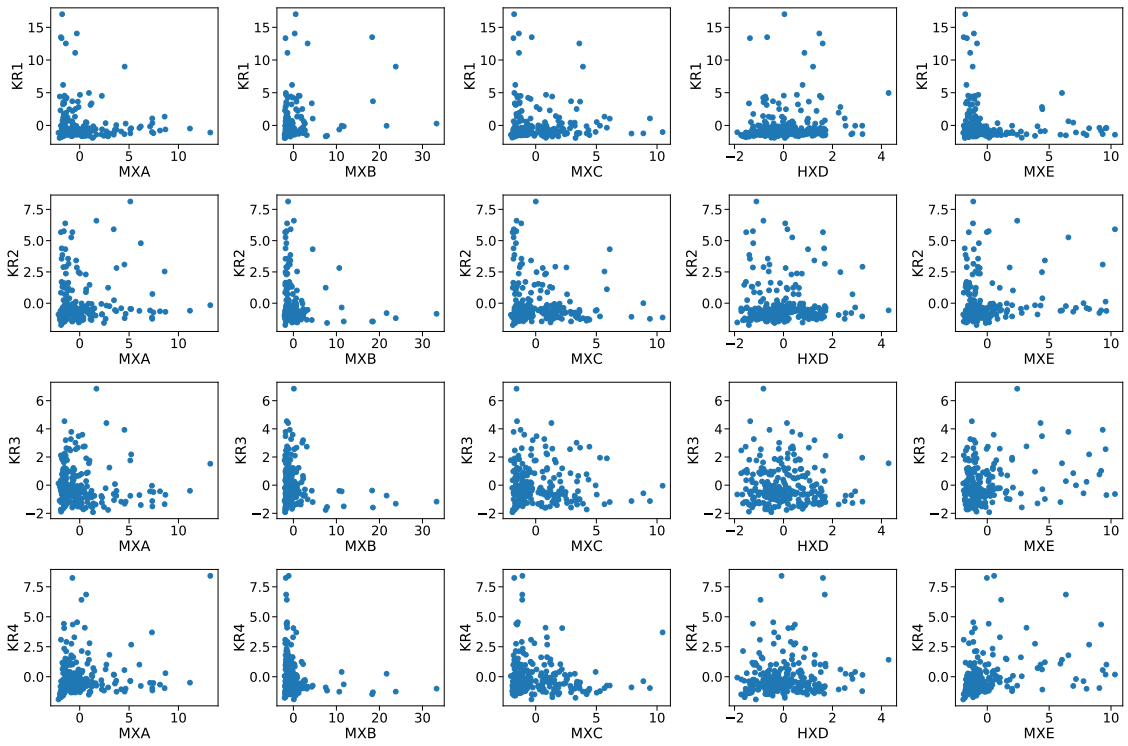

Figure 4.6: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 5. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

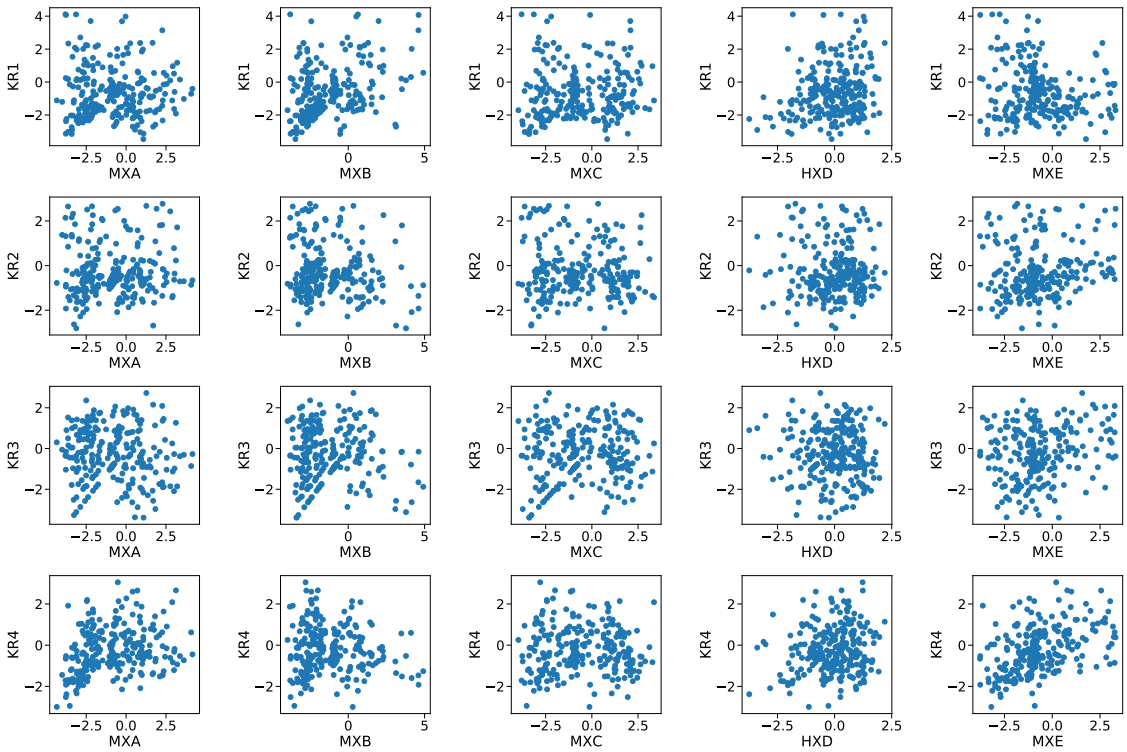

Figure 4.7: Scatter plots of SEEKR z-scores for all pair-wise comparisons between MXA,MXB,HXD,MXE and KR1,KR2,KR3, and KR4 using Method 6. Each dot represents a k-mer z-score for both transcripts being compared (x,y axis).

3 Discussion

Several methods for counting k-mers in SEEKR were analyzed. We observed the methodology in [3] could be improved, but the calculation in [4] was not optimal and a touch convoluted. We found that k-mers are log-normally distributed in the genome (Figure 4.1), and that adding a pseudo-count of 1 posterior to length normalization of the raw count data (Method 6), provided the sharpest distinction between correlated domains in *Xist* and *Rsx*, without erroneously increasing correlation between domains that we originally did not find to have sequence based relationships in [4] as well as linear relationships between the data that are well suited for Pearson correlation calculations (Figure 4.7).

4 Methods

4.1 k-mer count distributions

For a given sequence, *e.g.*"seq1", the number of occurrences for each k-mer was calculated. This yields a vector representing $\{c_{AAA}, c_{AAT}, \ldots c_{GGG}\}$, where c_{kmer} is the number of observed counts for that k-mer. We then calculated the frequency distribution of counts in a sequence, over all k-mers. *E.g.*, if the k-mers AAA and GGG had counts $c_{AAA} = 5$ and $c_{GGG} = 5$, and no other k-mers had a count of 5 in "seq1", then we would record a frequency of 2 for the number of times we observed a count of 5 in "seq1". These observed frequencies were calculated for each transcript and then summed, *e.g.* if we observed a count of 5 twice in "seq1" and 10 times in "seq2", then the marginal frequency of a count of 5 over those two sequences is 15.

$$P(\texttt{frequency}) = \sum_{\texttt{k-mers}} \sum_{\texttt{sequences}} P(\texttt{frequency,sequences,k-mers})$$

4.2 SEEKR Data Sources

For each of the analyses above, we used the following transcript annotations. For *XIST*, we used Mouse A-repeat (MXA) [5], mouse B-repeat (MXB) [5], human D-repeat (HXD) [6], and mouse E-repeat (MXE)[5]. For the *Rsx* repeats, we used the koala *Rsx* 1,2,3, and 4 repeats as defined in [4]. For the reference set of sequences within SEEKR, used the GENCODE (M14) annotated set of spliced lncRNA transcripts in the mouse transcriptome [7].

REFERENCES

[1] B. Haubold, "Alignment-free phylogenetics and population genetics," *Briefings in Bioinformatics*, 2014, ISSN: 14774054.

[2] K. Yang and L. Zhang, "Performance comparison between k -tuple distance and four model-based distances in phylogenetic tree reconstruction," *Nucleic Acids Research*, 2008, ISSN: 03051048.

[3] J. M. Kirk, S. O. Kim, K. Inoue, M. J. Smola, D. M. Lee, M. D. Schertzer, J. S. Wooten, A. R. Baker, D. Sprague, D. W. Collins, C. R. Horning, S. Wang, Q. Chen, K. M. Weeks, P. J. Mucha, and J. M. Calabrese, "Functional classification of long non-coding RNAs by k-mer content," *Nature Genetics*, 2018, ISSN: 15461718.

[4] D. Sprague, S. A. Waters, J. M. Kirk, J. R. Wang, P. B. Samollow, P. D. Waters, and J. M. Calabrese, "Nonlinear sequence similarity between the Xist and Rsx long noncoding RNAs suggests shared functions of tandem repeat domains," *RNA*, 2019, ISSN: 14699001.

[5] N. Brockdorff, "The product of the mouse Xist gene is a 15 kb inactive X-specific transcript containing no conserved ORF and located in the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 515–526, 1992 10, ISSN: 0092-8674.

[6] C. J. Brown, "The human XIST gene: analysis of a 17 kb inactive X-specific RNA that contains conserved repeats and is highly localized within the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 527–542, 1992 10, ISSN: 0092-8674.

[7] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles, J. Lagarde, L. Veeravalli, X. Ruan, Y. Ruan, T. Lassmann, P. Carninci, J. B. Brown, L. Lipovich, J. M. Gonzalez, M. Thomas, C. A. Davis, R. Shiekhattar, T. R. Gingeras, T. J. Hubbard, C. Notredame, J. Harrow, and R. Guigó, "The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression," *Genome Research*, 2012, ISSN: 10889051.

CHAPTER 5: CONCLUSION

The non-coding portion of our genome represents a significant fraction of the mammalian transcriptome [1]. Small non-coding RNAs are relatively well understood [1], however they make up only a fraction of annotated non-coding RNA transcripts. Despite representing a large fraction of known transcripts in the mammalian genome, we have solid understanding of only a handful of lncRNAs. *XIST* was discovered in 1992 [2], and yet we are still learning how exactly Xist functions within the cell [3, 4].

A significant frustration is the unclear sequence to function relationship in lncRNAs. This relationship has long been understood in protein coding genes [5] and has been facilited by numerous bioinformatics tools that can rapidly predict relationships between protein coding genes [6–10]. Relatively speaking though very little is known about lncRNAs. It is known that the tandem repeats of *XIST* encode a significant amount of its function [11–15], despite much of the sequence of *XIST* being relatively poorly conserved [11]. This has led to the notion that lncRNAs may derive a significant amount of their function from small sub-sequences [16, 17].

Our lab developed a k-mer based algorithm for quantifying sequence similarity between two lncRNAs [18]. This revealed large communities of lncRNAs that are related based on their k-mer content, and these communities were predictive of cellular function [18]. However, the functional analog to *XIST* in marsupials, *Rsx* was anti-correlated with *XIST* in this analysis. This observation led to the implementation of a modular approach to lncRNA sequence analysis – *i.e.*, breaking the sequences up into known or hypothesized functional domains and comparing only those regions. It was only after this that the degree of similarity between *XIST* and *Rsx* could be realized [17]. We then developed a statistical model using an HMM framework that was capable of identifying functional domains within lncRNAs that lack any sort of obvious tandem repeat domains as in *XIST* and *Rsx*.

We now have a set of tools that can answer how and where the function is encoded within lncRNAs. There remain significant ways that they can be improved and areas in which further

understanding is needed within lncRNAs. A primary shortcoming of SEEKR is that we do not consider secondary structure of the RNA when modeling its potential function. RNA molecules are able to form a wide variety of secondary structures in a sequence dependent manner that are known to be functional within the cell [19, 20]. Despite this, we are able to capture extensive RBP binding events in *XIST* and *KCNQ1OT1* with high precision and without considering structure at all. While structure is certainly important and has been shown to be conserved in lncRNAs [21], these results imply that overly focusing on secondary structure as the primary mechanism of lncRNA function may not tell the whole story.

Another limitation of *hmmSEEKR* is the assumption of a 2-state HMM and the requirement for the *a priori* definition of a query. Extending the logic behind what the *XIST* queries represent, it is possible to generate a query-of-interest through a Bayesian sampling algorithm [22]. One way to view the sequences of the functional domains of *XIST* is as a mixture model of k-mers that ultimately correspond to subset of differing proteins that recognize that sequence region. Given a set of weights from which to sample RBP PWMs, an artificial query can be generated by sampling k-mers from the PWMs with frequencies determined by the weights given to each protein. A further extension of the HMM model removes the assumption on a 2-state system entirely by utilizing a hierarchical Dirichlet process HMM (HDP-HMM) [23]. Put briefly, the HDP-HMM models the number of hidden states within a sequence without any prior information on what the hidden states are. This allows the model to determine how many functional domains may be within a sequence, what the k-mer content of those functional domains is, and where these domains are within a sequence.

lncRNAs represent a significant challenge in molecular biology – both computationally and experimentally. However, they represent approximately a third of annotated transcripts in the human genome, so it is crucial that models of their function are developed. Given the complex nature of lncRNAs, and their roles as hubs within the cell, I expect that future developments in the field will extend beyond simply modeling the sequence of a lncRNA, as well as their secondary and tertiary structure. Rather, integrative models, that model the complex interactions between RNA, protein, and DNA, will push the field further. With proteins and mRNAs, this was unnecessary to prediction the function of a transcript – because the protein was effectively the molecule performing a single function. With lncRNAs, again, their role is much more subtle and

a piece in a larger puzzle. I also believe that as more knowledge is gained about lncRNA function, more advanced models can be proposed that incorporate more prior information. Perhaps there are sequence features that, due to lack of training data knowledge, are simply undetectable for the moment. Finally, further developments in understanding conservation of non-coding regions of the genome, and development of models that consider conservation of sequence features other than linear alignment, will be crucial to understanding the nature of lncRNAs in the cell.

REFERENCES

[1] J. L. Rinn and H. Y. Chang, "Genome Regulation by Long Noncoding RNAs," *Annual Review of Biochemistry*, 2012, ISSN: 0066-4154.

[2] C. J. Brown, "The human XIST gene: analysis of a 17 kb inactive X-specific RNA that contains conserved repeats and is highly localized within the nucleus.," *Cell (Cambridge)*, vol. 71, no. 3, pp. 527–542, 1992 10, ISSN: 0092-8674.

[3] M. D. Schertzer, K. C. Braceros, J. Starmer, R. E. Cherney, D. M. Lee, G. Salazar, M. Justice, S. R. Bischoff, D. O. Cowley, P. Ariel, M. J. Zylka, J. M. Dowen, T. Magnuson, and J. M. Calabrese, "lncRNA-Induced Spread of Polycomb Controlled by Genome Architecture, RNA Abundance, and CpG Island DNA," *Molecular Cell*, 2019, ISSN: 10974164.

[4] S. T. Da Rocha and E. Heard, *Novel players in X inactivation: Insights into Xist-mediated gene silencing and chromosome conformation*, 2017.

[5] J. C. Whisstock and A. M. Lesk, *Prediction of protein function from protein sequence and structure*, 2003.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, 1990, ISSN: 00222836.

[7] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, 1981, ISSN: 00222836.

[8] T. J. Wheeler and S. R. Eddy, "Nhmmer: DNA homology search with profile HMMs," *Bioinformatics*, 2013, ISSN: 14602059.

[9] H. Yi and L. Jin, "Co-phylog: An assembly-free phylogenomic approach for closely related organisms," *Nucleic Acids Research*, 2013, ISSN: 03051048.

[10] J. Qi, B. Wang, and B. I. Hao, "Whole Proteome Prokaryote Phylogeny Without Sequence Alignment: A K-String Composition Approach," *Journal of Molecular Evolution*, 2004, ISSN: 00222844.

[11] T. B. Nesterova, S. Y. Slobodyanyuk, E. A. Elisaphenko, A. I. Shevchenko, C. Johnston, M. E. Pavlova, I. B. Rogozin, N. N. Kolesnikov, N. Brockdorff, and S. M. Zakian, *Characterization of the genomic Xist locus in rodents reveals conservation of overall gene structure and tandem repeats but rapid evolution of unique sequence*, 2001.

[12] X. Wang, K. J. Goodrich, A. R. Gooding, H. Naeem, S. Archer, R. D. Paucek, D. T. Youmans, T. R. Cech, and C. Davidovich, "Targeting of Polycomb Repressive Complex 2 to RNA by Short Repeats of Consecutive Guanines," *Molecular Cell*, 2017, ISSN: 10974164.

[13] Y. Hoki, N. Kimura, M. Kanbayashi, Y. Amakawa, T. Ohhata, H. Sasaki, and T. Sado, "A proximal conserved repeat in the Xist gene is essential as a genomic element for X-inactivation in mouse," *Development*, 2009, ISSN: 09501991.

[14] J. Zhao, B. K. Sun, J. A. Erwin, J. J. Song, and J. T. Lee, "Polycomb proteins targeted by a short repeat RNA to the mouse X chromosome," *Science*, 2008, ISSN: 00368075.

[15] G. Pintacuda, G. Wei, C. Roustan, B. A. Kirmizitas, N. Solcan, A. Cerase, A. Castello, S. Mohammed, B. Moindrot, T. B. Nesterova, and N. Brockdorff, "hnRNPK Recruits PCGF3/5-PRC1 to the Xist RNA B-Repeat to Establish Polycomb-Mediated Chromosomal Silencing," *Molecular Cell*, 2017, ISSN: 10974164.

[16] N. Brockdorff, "Local tandem repeat expansion in Xist RNA as a model for the functionalisation of ncRNA," *Non-coding RNA*, 2018, ISSN: 2311553X.

[17] D. Sprague, S. A. Waters, J. M. Kirk, J. R. Wang, P. B. Samollow, P. D. Waters, and J. M. Calabrese, "Nonlinear sequence similarity between the Xist and Rsx long noncoding RNAs suggests shared functions of tandem repeat domains," *RNA*, 2019, ISSN: 14699001.

[18] J. M. Kirk, S. O. Kim, K. Inoue, M. J. Smola, D. M. Lee, M. D. Schertzer, J. S. Wooten, A. R. Baker, D. Sprague, D. W. Collins, C. R. Horning, S. Wang, Q. Chen, K. M. Weeks, P. J. Mucha, and J. M. Calabrese, "Functional classification of long non-coding RNAs by k-mer content," *Nature Genetics*, 2018, ISSN: 15461718.

[19] N. A. Siegfried, S. Busan, G. M. Rice, J. A. Nelson, and K. M. Weeks, "RNA motif discovery by SHAPE and mutational profiling (SHAPE-MaP)," *Nature methods*, 2014, ISSN: 15487105.

[20] Y. Wan, M. Kertesz, R. C. Spitale, E. Segal, and H. Y. Chang, *Understanding the transcriptome through RNA structure*, 2011.

[21] P. Johnsson, L. Lipovich, D. Grandér, and K. V. Morris, *Evolutionary conservation of long non-coding RNAs; Sequence, structure, function*, 2014.

[22] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite Mixture Models," *Annual Review of Statistics and Its Application*, 2019, ISSN: 2326-8298.

[23] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric Hidden semi-Markov models," *Journal of Machine Learning Research*, 2013, ISSN: 15324435.