ENHANCING THE G4-CATCHALL ALGORITHM THROUGH DETECTION OF SECONDARY STRUCTURES WITHIN EXTREME LOOPS

BERÇİN SÖNMEZ

Thesis for the Master's Program in Bioengineering

Graduate School
Izmir University of Economics
Izmir
2023

ENHANCING THE G4-CATCHALL ALGORITHM THROUGH DETECTION OF SECONDARY STRUCTURES WITHIN EXTREME LOOPS

BERÇİN SÖNMEZ

THESIS ADVISOR: ASSOC. PROF. DR. OSMAN DOLUCA

A Master's Thesis
Submitted to
the Graduate School of Izmir University of Economics
the Department of Bionengineering

Izmir
2023

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behaviour at every stage from the planning of the thesis to its defence. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behaviour, and that all statements not cited are my own.

Name, Surname:
BERÇİN SÖNMEZ

Date:
18.10.2023

Signature:

ABSTRACT

ENHANCING THE G4-CATCHALL ALGORITHM THROUGH DETECTION OF SECONDARY STRUCTURES WITHIN EXTREME LOOPS

Sönmez, Berçin

Master’s Program in Bioengineering

Advisor: Assoc. Prof. Dr. Osman DOLUCA

October, 2023

G-quadruplexes are high-order structures formed by non-canonical, telomeric nucleic acids, such as single-stranded guanine (G)-rich DNA and RNA sequences. G-quadruplexes represent a class of non-canonical nucleic acid structures that arise from guanine-rich sequences. Many algorithms have been developed to predict G-quadruplex-forming sequences in DNA and RNA. The expectation is that an improved version of the G4-CATCHALL algorithm can be created to predict G-quadruplex sequences in DNA and RNA. It is aimed to integrate the detection of secondary structures in extreme cycles known to contain potential G-quads, and to use an expanded parameter set to increase the accuracy of the prediction. It is expected that the performance of the improved algorithm will be evaluated in both synthetic and experimental datasets and will contribute to the original G4-CATCHALL algorithm. Evaluating the performance of the improved algorithm on both synthetic and experimental datasets, it is expected to be a valuable tool in identifying potential G-quadruple forming sequences, which is important for contributing to the original G4-

CATCHALL algorithm and making potential therapeutic applications more understandable.

Keywords: G-quadruplex, G-quadruplex prediction, Motif prediction, extreme loops, G4Catchall, RNA and DNA topology,

ÖZET

AŞIRI DÖNGÜLER İÇİNDEKİ İKİNCİL YAPILARIN TESPİT EDİLMESİYLE G4-CATCHALL ALGORİTMASININ GELİŞTİRİLMESİ

Sönmez, Berçin

Biyomühendislik Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Osman DOLUCA

Ekim, 2023

G-dörtlü yapılar, tek sarmallı guanin (G) açısından zengin DNA ve RNA dizileri gibi kanonik olmayan, telomerik nükleik asitler tarafından oluşturulan yüksek dereceli yapılardır. G-dörtlü gruplar, guanin bakımından zengin dizilerden ortaya çıkan bir kanonik olmayan nükleik asit yapıları sınıfını temsil eder. DNA ve RNA'da G-dörtlü oluşturan dizileri tahmin etmek için birçok algoritma geliştirilmiştir. Beklenti, DNA ve RNA'daki G-dörtlü dizilerini tahmin etmek için G4-CATCHALL algoritmasının geliştirilmiş bir sürümünün oluşturulabilmesidir. Potansiyel G-dörtlüleri içerdiği bilinen aşırı döngülerdeki ikincil yapıların tespitini entegre etmek ve tahminin doğruluğunu artırmak için genişletilmiş bir parametre seti kullanmak amaçlanmaktadır. Geliştirilen algoritmanın performansının hem sentetik hem de deneysel veri setlerinde değerlendirilmesi ve orijinal G4-CATCHALL algoritmasına katkı sağlaması beklenmektedir. İyileştirilmiş algoritmanın performansının hem sentetik hem de deneysel veri kümeleri üzerinde değerlendirilmesi, orijinal G4-CATCHALL algoritmasına katkıda bulunmak ve potansiyel terapötik uygulamaları daha anlaşılır kılmak için önemli olan potansiyel G-dörtlü oluşturma dizilerini belirlemede değerli bir araç olması bekleniyor.

Anahtar Kelimeler: G-dörtlüleri, G-dörtlü tahmini, Motif tahmini, aşırı döngüler, G4Catchall, RNA ve DNA topolojisi.

This thesis dedicated to my parents.

For their endless love, support and encouragement.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Assoc. Prof. Dr. Osman DOLUCA, for their invaluable guidance and unwavering support throughout my master’s program. Their expertise, forbearance, and inspiration have enabled me to persevere in this research and compose this thesis.

I am appreciative of my parents' unending love, steadfast courage, and faith in my skills. Their consistent support, empathy, and patience have been a wellspring of fortitude and incentive. I acknowledge their sacrifices in facilitating my academic aspirations and sustaining my research odyssey. Their unflagging support has been the bedrock of my accomplishments, and I consider myself truly fortunate to have them as my pillars of strength.

During this process, I express my profound gratitude to my esteemed companions, Damla KÖROĞLU and İlayda TOPTAŞ for their unwavering affection and support. Without their constant support and inspiration, I would not have completed this difficult path successfully.

Ultimately, I extend my gratitude to my feline companion, Simba, for his unwavering patience and invaluable emotional encouragement throughout the duration of my scholarly endeavor. His steadfast companionship allowed me to persevere and remain in good spirits, particularly during the trying and anxiety-inducing periods.

TABLE OF CONTENTS

ABSTRACT iv
ÖZET vi
ACKNOWLEDGEMENTS ix
TABLE OF CONTENTS x
LIST OF TABLES xi
LIST OF FIGURES xii
CHAPTER 1: INTRODUCTION 1
1.1. Overview of G-quadruplex Structures and Their Biological Significance....... 1
1.2. Brief Summary of G4Catchall Algorithm and Its Limitations 2
1.3. Importance of Detecting Secondary Structures Within Extreme Loops 4
1.4. Possible Hairpin-G4 Hybrid Structure Formation 6
1.5. Paths of Secondary Structure G4 Compatation 7
CHAPTER 2: LITERATURE REVIEW 10
2.1. Previous Research on G-quadruplex and Their Prediction 10
2.2. Previous Research on the G4Catchall Algorithm and Its Limitations 13
2.3. Previous Research on the Detection of Secondary Structures within Extreme Loops 14
CHAPTER 3: METHODLOGY 15
3.1. How RNAFold is Working 15
3.2. Explanation of the Datasets Used in the Study 16
3.3. Overview of the Enhanced G4Catchall Algorithm with the Inclusion of the Secondary Structures within Extreme Loop 17
CHAPTER 4: RESULTS 28
CHAPTER 5: DISCUSSION 32
CHAPTER 6: CONCLUSION 33
REFERENCES 34

LIST OF TABLES

Table 1. Confusion Matrix 31
Table 2. Recall(Sensitiviy) Analyses 31

LIST OF FIGURES

Figure 1. Structures of G-quadruplex 1
Figure 2. A typical G4 structure is shown schematically. 6
Figure 3. Synthetic sequences. 28
Figure 4. Synthetic sequences testing on RNAfold. 29
Figure 5. Window size plots of profiles 30

CHAPTER 1: INTRODUCTION

1.1 Overview of G-quadruplex Structures and Their Biological Significance

Nucleic acid structures, counting G-quadruplexes, play a crucial part in different organic forms such as gene expression, genome solidness, and cellular capacities. G-quadruplexes are shaped by guanine-rich sequences, where four guanine bases come together to create a steady square-planar course of action through Hoogsteen base blending (Sun et al., 2019; Zhang et al., 2009). The arrangement of G-quadruplexes includes the stacking of G-tetrads, which are encourage stabilized by cations, regularly potassium particles, and mediating circles.

Numerous genomic domains, including promoter fragments and telomeres, have been demonstrated in many studies to contain G-quadruplexes. Studies have appeared that G-quadruplexes have impacts on transcriptional action, telomere support, DNA replication and numerous cellular capacities (Pavlova et al.,2021).

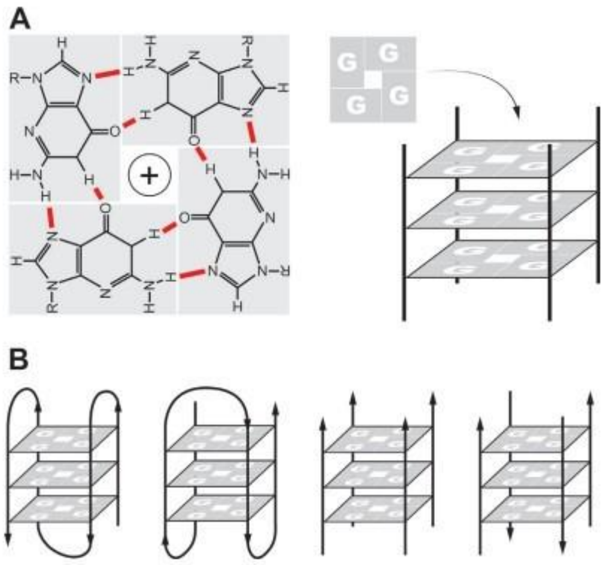


Figure 1. Structures of G-quadruplex. A) G-quadruplex form in DNA or RNA sequences B) Discrimination of G-quadruplex structures (Rhodes and Lipps, 2015).

In the studies, many techniques have been used to examine the structural properties of G-quads in detail. These techniques have generally emerged as experimental studies such as biophysical analysis, X-ray crystallography, NMR spectroscopy, and computational modeling (Rhodes and Lipps, 2015). Through the discoveries of the investigates, more detailed and broad information was obtained approximately the stability of G-quadruplexes and their interactions with proteins and small molecules.

Understanding the key part that G-quadruplexes play in biological processes and having a broad information of their complex structure and potential applications is fundamental.This investigate zone proceeds to advance, with progressing examinations into the challenges, open questions, and future headings within the field.By uniting the existing information and motivating advance investigation, analysts point to harness the potential of G-quadruplexes for different therapeutic and diagnostic applications (Puig Lombardi and Londoño-Vallejo, 2020).

1.2. Brief Summary of G4Catchall Algorithm and Its Limitations

The G4catchall algorithm may be a computational tool that predicts the arrangement of G-quadruplex structures in DNA sequences. It was created based on the examination of experimentally approved G-quadruplex sequences, and it employments a set of rules to predict the probability of G-quadruplex arrangement in a given DNA sequence. The algorithm considers different components, counting the length and composition of the DNA sequence, the presence of G-runs, and the stability of the predicted G-quadruplex structure.

The G4Catchall algorithm moreover has certain limitations. A case of this limitation is its failure to assess the region of G-quadruplexes. In expansion, the algorithm does not take into consideration the impact of flanking sequences or the impact of cellular variables on G-quadruplex arrangement. Subsequently, encourage exploratory approval is essential to affirm the predicted G-quadruplex structures and their potential biological importance.

The G4catchall algorithm and other G-quadruplex (G4) detection algorithms play a vital part in distinguishing and characterizing G4 structures in nucleic acid sequences. It utilizes a machine learning approach that combines different sequence-based highlights, such as G4-specific motifs, stability, and loop characteristics, to classify potential G4-forming locales.

Other algorithms utilized for G4 prediction incorporate QGRS Mapper, Quadparser, G4NN, and G4Hunter. Quadparser and QGRS Mapper are two algorithms commonly utilized for G-quadruplex (G4) prediction. Quadparser recognizes G4 motifs by applying a predefined set of rules. At the same time, QGRS Mapper utilizes a pattern-matching approach utilizing regular expressions to distinguish G4s based on G-rich sequences and loop lengths. Both algorithms are included within the recognizable of G4 structures, with Quadparser putting accentuation on sequence patterns and thermodynamic stability, whereas QGRS Mapper centers on the acknowledgment of G4 motifs through pattern coordinating. Both G4Hunter and G4NN utilize different strategies to distinguish G4s, with the previous using a probabilistic model and the other one utilizing an artificial neural network strategy. It is vital to note that the adjustment between susceptibility and specificity could be a critical figure to consider (Kikin, D'Antonio and Bagga, 2006; Huppert, 2005).

Whereas Quadparser, G4NN, G4Hunter, and QGRS Mapper have been important in recognizing G-quadruplexes (G4s), they are not without limitations. One common limitation is their dependence on sequence-based highlights, which can lead to wrong positives or the neglecting of G4s with atypical highlights. Moreover, certain algorithms have restrictions on sequence length or particular nucleotide contexts, limiting their feasibility in certain genomic regions. The delicate adjustment between susceptibility and specificity must moreover be taken into mind. A few algorithms prioritize susceptibility, allowing for the detection of a wide extend of potential G4-forming regions, but with a higher wrong positive rate. On the other hand, other algorithms prioritize specificity, result in lower wrong positive rates but possibly lost a few true G4 structures (Bedrat, Lacroix and Mergny, 2016).

In expansion to the previously mentioned limitations, the field of G4 prediction faces challenges related to agreement and structural assessment. Firstly, the lack of agreement among distinctive algorithms presents changefulness in G4 distinguishing proof, as each algorithm utilizes significant models, parameter settings, and training datasets. Subsequently, there can be incompatibility in G4 prediction results, making it troublesome to determine a specific G4 landscape in a given sequence (Garant, Perreault and Scott, 2017).

Moreover, the dependence on sequence-based highlights in G4 prediction algorithms neglects substantial structural aspects of G4s. G4 structures are impacted by variables past the primary sequence, such as DNA topology, flanking sequences, and the chromatin environment (Chen et al., 2021). These structural components can considerably affect G4 arrangement and stability but are not continuously incorporated into sequence-based prediction algorithms. Subsequently, the current approaches may not completely capture the complexity of G4 structures and their functional implications.

Tending to these limitations requires progressions in G4 prediction algorithms that organize both sequence-based and structural highlights. By joining a more comprehensive understanding of G4 arrangement and considering additional components past the nucleotide sequence, ready to improve the accuracy and reliability of G4 prediction strategies and achievement more profound insights into their biological importance.

G4catchall and other G4 prediction algorithms ensure important tools for initializing potential G4 structures in nucleic acid sequences. In any case, their performance may be affected by the trade-off between susceptibility and specificity, dependence on sequence-based highlights, and the complexity of G4 arrangement completely different biological contexts. Advance progressions and integration of experimental information are essential to progress the precision and reliability of G4 prediction algorithms.

1.3 Importance of Detecting Secondary Structures Within Extreme Loops

The importance of detecting secondary structures within extreme loops lies in the fact that these structures can play a crucial role in various biological processes. Extreme loops are the longest unpaired stretches in RNA structures and can harbor functional elements such as binding sites, cleavage sites, and regulatory elements. Secondary structures within extreme loops have been shown to be involved in RNA stability, localization, and function (Risitano, 2004).

For example, studies have shown that RNA structures with stable hairpins within extreme loops have increased stability and protection against RNase degradation. In expansion, it has been appeared that the presence of hairpins within extreme loops can balance RNA-protein intuitive and can serve as acknowledgment sites for RNA-binding proteins.

Moreover, RNA structures with G-quadruplexes within extreme loops have been appeared to play a part in translational regulation and splicing. G-quadruplexes within extreme loops have been appeared to be recognized by RNA-binding proteins and to play a part in elective splicing.Therefore, detecting secondary structures within extreme loops can provide important insights into RNA function and regulation

(Zhang, Harvey and Cheng, 2019).

As a result, understanding the secondary structures found in extraordinary loops can give critical modern bits of knowledge into how RNA functions and how its regulation works.

Predicting secondary structures within extreme loops is important as these regions often exhibit unique characteristics and functional roles. Extreme loops, moreover known as huge loops or long loops, are locales in nucleic acids where the length of the loop surpasses a certain limit or shows unusual highlights compared to ordinary loops Here's why extreme loops are important and how various tools are employed to study them:

Structural diversity: Extreme loops exhibit diverse structural properties, including complex folding patterns, bulges, internal loops, and higher-order structural motifs. Understanding the structural characteristics of extreme loops provides insights into their stability, flexibility, and potential functional roles.

Regulatory functions are of most extreme importance in various biological processes, counting but not restricted to transcriptional regulation, RNA processing, and post-transcriptional alterations. In specific, extreme loops play a vital part in these processes. They can act as regulatory components, enhancers, or silencers, influencing gene expression and cellular capacities (Mattick et al., 2023).

The regions that show a tall prevalence of extreme loops are ordinarily those of non-coding RNAs, which play a vital role in gene regulation, cellular signaling, and disease treatment. These loops are known to help with the creation of functional RNA structures, counting riboswitches, ribozymes, and RNA-binding protein recognition sites (Radecki et al., 2021, Statello et al., 2021).

Extreme loops, which function as binding sites for RNA-binding proteins (RBPs) and other regulatory molecules, improve RNA-protein interactions. The area, stability, and protein recognition of RNA are altogether impacted by these interactions. The mechanics of RNA-protein complexes can be way better caught on by characterizing these interactions, which too sheds light on how they affect functional results.

Numerous inherited diseases and disorders have been connected to extreme loop modifications or structural changes. Such mutations may disturb the stability or functionality of RNA structures within extreme loops, ultimately resulting in anomalous gene expression or impeded cellular processes. This highlights the significant role of extreme loops in disease associations. Recognizing and understanding these structural modifications can help in disease determination and therapeutic mediations (Weskamp and Barmada, 2018).

Several computational tools and algorithms are available for studying extreme loops and predicting their secondary structures. Some commonly used tools include RNAstructure, RNAStructureFold, NUPACK, and ViennaRNA Package. These tools utilize various approaches such as thermodynamic calculations, comparative genomics, and machine learning algorithms to predict RNA secondary structures and analyze extreme loops (Zadeh et al., 2010; Lorenz et al., 2011; Reuter and Mathews, 2010).

Through utilization of these techniques, researchers can attain a higher level of comprehension regarding the structural and functional attributes of anomalous loops, discern their operational mechanisms within biological processes, and scrutinize their plausible ramifications on disease pathways.

1.4 Possible Hairpin-G4 Hybrid Structure Formation

A hairpin structure and a G-quadruplex (G4) structure coexisting or interacting with one another within the same RNA or DNA molecule is referred to as the creation of hairpin-G4 hybrid structures. In order to create a stem-loop structure, complementary sections of a nucleic acid strand must couple together intramolecularly, creating hairpins. The stacking of guanine (G) nucleotides into planar structures known as G-quartets, on the other hand, results in the formation of G-quadruplexes, which are non-canonical nucleic acid structure (Ravichandran et al., 2021).

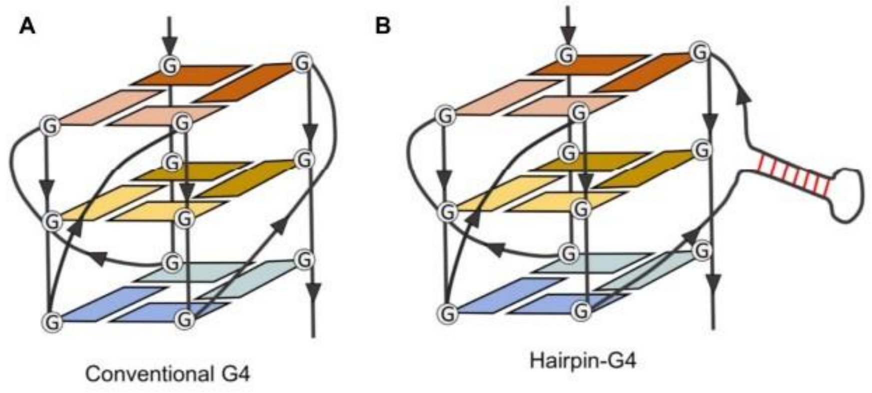


Figure 2. A) A typical G4 structure is shown schematically. B) A G4-hairpin (Ravichandran et al., 2021).

There are numerous processes via which hairpin-G4 hybrid structures might develop. If there is no direct connection between the two structures, the coexistence of G4 and hairpin structures in diverse positions on the same molecule offers one possible explanation for this phenomenon. The formation of a hairpin shape with a G-rich sequence in the loop region is another plausible possibility. The hairpin stem of this structure can fold into a G-quadruplex structure while doing so. In this instance, the G4 and hairpin structures are physically linked together inside the same molecule.

The potential roles that hairpin-G4 hybrid structures could play in a variety of biological processes has sparked interest in them. They might contribute to the control of translation, mRNA processing, gene regulation, and other cellular processes. The accessibility of protein and other molecule binding sites can be influenced by the coexistence of hairpin and G4 structures, which can have an impact on gene expression and cellular processes (Agarwal et al., 2014).

Utilizing both experimental and computational methods is necessary to understand the creation and characteristics of hairpin-G4 hybrid structures. X-ray crystallography, nuclear magnetic resonance (NMR), and spectroscopic approaches are examples of experimental techniques that can provide structural information. Numerous computational methods and techniques have been developed to forecast the generation of hairpin-G4 hybrid structures based on an analysis of sequence and thermodynamics. RNAfold, an application for folding RNA, is one such instrument (Winnerdy et al., 2019).

All things considered, the potential emergence of hairpin-G4 hybrid structures complicates our knowledge of RNA and DNA structure and function. Examining these hybrid structures can help us understand the various ways that nucleic acids act and how they are regulated.

1.5 Paths of Secondary Structure G4 Compatation

Different folding mechanisms can result in the development of secondary structures in G-quadruplex (G4) DNA or RNA molecules. The sequence and length of the G4-forming area, the presence of cations or other molecules that stabilize the structure, the thermodynamic stability of various intermediate states, and other variables all play a role in determining the precise folding process (Grün and Schwalbe, 2021).

There are general phases involved in the production of G4 secondary structures, even though the precise folding pathway for a given G4 structure can be complex and reliant on particular sequence properties. A folding mechanism that can compact G4 structures has been developed, and it can be summed up as follows (You et al., 2017):

Initial unstructured state: The guanine-rich portion of the G4-forming sequence is single-stranded and devoid of any distinct secondary structure.

Nucleation: The creation of a G-quartet, a planar arrangement of four guanine bases bound together by Hoogsteen hydrogen bonding, usually marks the beginning of the folding process. In order to create the initial G-quartet, neighboring guanine bases must be stacked during this nucleation process.

Propagation: Guanine bases can stack on top of one another to produce a G-quadruplex core after the initial G-quartet has been formed. The consecutive addition of guanine bases during this propagation stage results in the development of many G-quartets, usually in a planar configuration.

Normally, loop regions arise to connect the guanine lengths between the G-quartets. These loops may adopt various conformations and have varying lengths and sequences. The growth of loops supports the overall stability and structural variation of G4 secondary structure.

Stabilization: The binding of cations, such as potassium (K+) or sodium (Na+), in the central channel created by the G-quartets frequently increases the stability of the G4 structure. The oxygen atoms of guanine bases can coordinate with cations, providing electrostatic stability and encouraging the creation of a compact G4 structure.

Higher-order structures can sometimes be created by further folding or stacking G4 motifs with one another. These structures may result from interactions between G4 sequences at the molecular level or through the assembly of G4-quadruplexes with extra stems or loops.

It's important to remember that a number of variables, including the presence of other DNA-binding proteins, the concentration of cations, and the existence of other secondary structures nearby, might affect the folding pathway and speed of G4 compaction. Our knowledge of the folding processes and dynamics of G4 structures is aided by experimental methods like NMR spectroscopy, X-ray crystallography, and single-molecule investigations.

Overall, the development and growth of G-quartets, the formation of loops, the coordination of cations, and potential higher-order folding events all contribute to the compacting of G4 structures. Understanding these routes is essential for determining the functional functions and physical characteristics of G4 structures. The sequence and environmental factors may change the particular folding mechanism.

CHAPTER 2: LITERATURE REVIEW

2.1 Previous Research on G-quadruplex and Their Prediction

Motifs consisting of non-canonical and guanine-rich sequences play important roles in gene regulation and stabilization. As a result of the researches, these motifs are called G-quartets. Guanine tetrads are stacked together in these arrangements and held together by Hoogsteen hydrogen bonds, making steady secondary structures. Various nucleic acid atoms, counting DNA, RNA, and telomeric DNA, have been appeared to incorporate G-quadruplexes, which have been connected to critical physiological functions such DNA replication, translation, and translation. Also, G-quadruplexes are charming candidates for therapeutic mediation due to their affiliation with a number of human diseases, counting cancer and neurodegenerative disorders (Kwok and Merrick, 2017).

For the reason of explaining their functional roles and comprehending their suggestions in disease pathways, it is significant to precisely predict the structures of G-quadruplexes. In spite of giving point by point structural information, Nuclear magnetic resonance (NMR) spectroscopy and X-ray crystallography are strenuous, time-consuming, and not agreeable to high-throughput examination. Thus, computer-based prediction strategies have risen as important adjuncts for quickly distinguishing putative G-quadruplex themes in nucleotide sequences (Das et al., 2021).

For the purpose of predicting G-quadruplex structures in nucleotide sequences, a variety of methods have been used. The availability of sequence data, the desired level of accuracy, and the available computational resources are only a few of the variables that influence which approach is chosen. The taking after are a few of the typical strategies:

One way is the sequence-based procedure, which predicts G-quadruplex-prone regions by taking into thought sequence characteristics such the presence of G-rich motifs and specific nucleotide sequences. This method frequently uses machine learning or statistical analysis to find probable G-quadruplex motifs based on sequence properties (Monti et al., 2021). The structural-based method is an alternative strategy that predicts the presence of G-quadruplexes in nucleotide sequences by taking into account their structural characteristics, such as loop lengths, loop sequences, and the number of G-tetrads. This approach frequently uses algorithms that examine the stability and folding potential of G-quadruplex structures using structural principles and energy estimates. Sequence-based and structural-based elements are used in hybrid approaches to increase the accuracy of G-quadruplex prediction. By combining sequence data with structural limitations, they trust to capture both neighborhood sequence patterns and global structural highlights that impact the creation of G-quadruplexes (Chambers et al., 2015). For G-quadruplex prediction, machine learning methods including support vector machines (SVM), random forests, and deep learning models have become more and more popular (Zhang et al., 2023). In order to predict the behavior of fresh sequences, these approaches attempt to learn patterns and relationships from the training data, which might comprise different aspects like sequence motifs, physicochemical characteristics, and structural parameters. For the prediction of G-quadruplexes, a variety of web-based servers and applications have been created with user-friendly user interfaces. These platforms frequently include additional features including the display of predicted structures and integration with other bioinformatics tools, as well as a variety of prediction techniques. The choice of a particular strategy should be based on the individual research issue, available resources, and desired accuracy. It ought to be noted that each technique has its one of a kind qualities and limits. Also, combining diverse procedures or consolidating information from a few sources can make strides the exactness and constancy of expectations (Brázda et al., 2019).

G4NN is a computational tool created to capture potential RNA G-quads . The implementation of a new measure based on abstract sequence similarity is suggested by the current research. This score is computed through the utilization of a rudimentary artificial neural network (ANN), denominated as G4NN. Notably, G4NN is trained on the sequences that are housed within the G4RNA database.The G4RNA screener was developed as a motif-independent tool based on the structural features of the G-quadruplexes (Garant, Perreault and Scott, 2017).

The developed tool evaluates the probability of G-quad formation on the basis of factors such as stability of G-quads, loop length and loop array composition. The generated algorithms are tested using data containing experimentally validated G-quadruplets and non-G-quadruplets.

Results from these test sequences used and the experimentally proven G-quads sequences demonstrated the accuracy of g4NN in detecting G-quads. Additionally, G4NN has demonstrated its capacity to improve predictive abilities for G-quads while also spotting unconventional G-quads through comparison with previously examined methods.

Researchers typically use the web-based server QGRS Mapper to carry out sequence analysis and identify probable G-quadruplex groups. The parameters of the QGRS Mapper have been developed taking loop length and stability into account, just like many other tools. In addition, it also includes sequence conservation analysis in order to evaluate G-quads that may occur among different species.

QGRS Mapper evaluates the accuracy of its performance over experimentally validated G-quad arrays. The web server also performs graphs of the predicted G-quads and detailed visualization of the predicted structure.It allowed researchers to customize the estimation parameters on the sequences they wanted to examine according to the subject they wanted to investigate (Kikin, D'Antonio and Bagga, 2006).

QGRS Mapper is an essential web server for researchers interested in the research and development of G-quads. The software furnishes a facile and user-friendly interface, together with resilient prediction algorithms and comprehensive visualization functionalities, thereby rendering the identification and analysis of G-quadruplex structures in nucleotide sequences undemanding.

While there are many methods and tools developed for the prediction of G-quads, the limitations of these methods make it very difficult for researchers to move their studies forward, and that's why G4Hunter was introduced to improve these limitations.

The G4Hunter algorithm includes many new features and a machine learning approach to improve the accuracy and reliability of the prediction of G4-quads.To evaluate a particular DNA sequence's tendency to generate G-quadruplex structures, G4Hunter makes use of a broad range of sequence and structural information. It considers context around probable G-quadruplex motifs as well as variables like guanine content, loop length, and stability.Utilizing a sizable dataset of G-quadruplexes with experimental validation, assess G4Hunter's performance. They show that, in terms of sensitivity, specificity, and total prediction accuracy, G4Hunter surpasses other prediction techniques.

The advancement of G4Hunter provides a significant contribution to the G-quadruplex research domain through the provision of a sophisticated algorithm for predicting G-quadruplex structures. Furthermore, it offers novel perspectives into the prevalence of these structures and their prospective biological importance.

2.2 Previous Research on the G4Catchall Algorithm and Its Limitations

The G4Catchall algorithm is a tool developed for the prediction and analysis of structures of G-quadruplexes in both DNA and RNA sequences. It aims to identify and characterize potential G4 motifs based on their sequence patterns and structural properties.

The algorithm implemented in this study employs a regular expression matching strategy, akin to other existing algorithms for G4 detection. The approach utilizes a distinctive pattern, denoted as the G4 Catchall motif, which is specifically tailored to effectively detect a wide range of G4 structures characterized by varying loop lengths and arrangements.

Gx1NL1Gx2NL2Gx3NL3Gx4

In this motif, the 'G' represents a guanine base, 'x' denotes a range of guanine repeats (typically 2 to 5), 'N' represents any nucleotide (A, T, C, or G), and 'L' denotes a range of loop lengths (typically 1 to 7 nucleotides). The subscripts 1, 2, 3, and 4 indicate different occurrences of G-runs and loop sequences.

By using this motif, the G4 Catchall algorithm searches for sequences that match the pattern in the input DNA or RNA sequence. It identifies potential G4 motifs by detecting sequences with multiple guanine runs separated by loop sequences of varying lengths. The algorithm does not consider the folding stability or in vivo likelihood of the predicted G4 structures; it provides a binary "yes/no" output based on pattern matching.

One of the challenges in G4 detection is handling nested structures, where multiple G4 motifs can overlap within a long sequence. The G4 Catchall algorithm offers two techniques for managing overlapping matches to handle this:

Counting only identical motifs that do not overlap In this method, the algorithm exclusively counts instances of the same G4 motif that do not overlap.This avoids overestimating the number of G4 structures in cases where nested structures may be present.

Counting overlapping motifs with different loop sequences: Here, the algorithm counts overlapping motifs with different loop sequences as separate G4 structures. This allows for the identification of distinct G4 motifs within nested structures.

The G4 Catchall algorithm provides a flexible framework for G4 detection and analysis, allowing researchers to customize the motif parameters and analyze different types of G4 structures. The use of this method is thought to be very helpful in the early assessment and detection of potential G4 patterns in DNA and RNA sequences.

2.3 Previous Research on the Detection of Secondary Structures within Extreme Loops

The examination of the affect of loops on quadruplex stability could be a significant calculate to comprehend the structural and functional characteristics of G-quadruplexes. Mediating sequences, moreover assigned as loops, represent the regions between continuous G-tracts inside the quadruplex structure, comprising non-G nucleotides. These loops apply a significant impact on determining the flow and stability of the quadruplex (Arora and Maiti, 2009).

Investigate has uncovered that the length and sequence composition of loops exerts a significant impact on the stability of G-quadruplex structures. In common, shorter loops contribute to more prominent stability, as they lead to more tightly packing of the G-tetrads. Alternately, longer loops present a degree of adaptability and may disturb the stacking intuitive between G-tetrads, coming about in diminished stability. By the by, particular loop sequences showing specific patterns and intuitive can upgrade the in general stability of the quadruplex structure (Guédin et al., 2010).

The stability of G-quadruplexes is affected by different components concerning the nature of loop residues, counting their character and situating inside the loop. The interest of loop residues in hydrogen bonding or other interactions with the G-tetrads can move forward stability. On the other hand, the nearness of loop residues that cause steric clashes or destabilizing interactions can decrease stability.

Besides, the stability and folding of G-quadruplex structures can be altogether affected by the nearness of bulges, which are unpaired nucleotides inside the loops. Bulges cause disjunction within the stacking of G-tetrads and modify hydrogen holding patterns, coming about in varieties in stability and structural elements.

CHAPTER 3: METHODLGY

3.1 How RNAFold is Working?

An RNA molecule's secondary structure can be predicted by the software program RNAfold using the main sequence. The most stable RNA molecule folding patterns are determined using thermodynamic methods.

The RNAfold algorithm is based on the ideas of minimum free energy (MFE) folding. It rates the stability of several RNA molecule folding scenarios and chooses the folding pattern with the lowest free energy as the most likely one. Base pair creation is taken into account in the free energy calculation, along with other structural elements including loops, bulges, and interactions between internal and exterior base stacking (Lorenz et al., 2011).

In the RNAfold prediction method, potential secondary structures are investigated in the combinatorial space and their stabilities are assessed. It looks for folding patterns that enhance base pairing while reducing the structure's overall free energy. The algorithm determines the most stable folding configuration by taking into account both local and global interactions.

RNAfold can handle multiple sequence alignments as well as individual RNA sequences. Base pairing probabilities, energy readings, and other structural annotations are all information regarding the expected secondary structure that is provided. The projected structures can also be graphically represented by RNAfold.

It's crucial to remember that RNAfold's calculations contain a number of assumptions and simplifications. The presumption stated above assumes that the folding process takes place within the parameters of physiological circumstances and that the RNA molecule in issue is submerged in an aqueous solution. In some situations, especially for complex RNA molecules or under unusual experimental circumstances, these hypotheses may restrict the precision of predictions (Wayment-Steele et al., 2022).

Overall, RNAfold is a popular technique for predicting RNA secondary structure and has greatly advanced our knowledge of RNA folding and structure-function correlations.

3.2 Explanation of the Datasets Used in the Study

All the potential ways to produce G4 were combined into one synthetic sequence. However, the human genome serves as our real sequence, and information about the human genome will be used to compare G-quadruplexes there. From a recent work (Chambers et al. others, 2015), the coordinates of the G-quadruplex-forming domains experimentally discovered in the human genome will be obtained. The relevant dataset including the sequences can be found in the GEO database under accession number GSE63874.

This dataset included several samples that were examined. These studies were conducted with Na, K, and PDS present. Among small molecule G4-linkers, pyridostatin (PDS) and its derivatives, such as PyPDS, are well recognized for their high affinity for G-quadruplexes (G4s). The precise mechanism by which they identify and attach to G4s is still unknown.

Although pyridostatin (PDS) is widely acknowledged as a powerful inducer and stabilizer of G-quadruplexes (G4s), it remains unclear what genes this substance specifically targets. In light of this situation, the instances created via PDS will be used as a dataset. Then, in case studies involving PDS, the 150 base pair single-ended Fastq reads (Read-1) were aligned to the human genome (hg19) using the bwa meme alignment tool. The equivalent chromosomal regions were assigned to the matching Read-1 and Read-2 files.

The generated alignment files in BAM format were translated to BED files and then handled by bedtools. Related actions 1) using the BAM to BED format converter (bamToBed); 3) group BED files only to maintain the best alignments (groupBy -g 4 -c 5 -o max); 4) extraction of fasta sequences corresponding to BED intervals (bedtools getfasta -s). 2) expanding BED files by 30 base pairs in both directions (slopBed -s -r 30); 3) grouping BED files only to keep the best alignments (groupBy -g 4 -c 5 -o max).

R was then used to compare the Read-1 and Read-2 files using the generated fasta sequence files and the original Fastq file. Any poly-A sequence with a tail exceeding 9 bases underwent a targeted clipping process. For each pair of readings, baseline discrepancies and differences in quality ratings were calculated. The average single base mismatch values for all reads spanning each matched genomic region were calculated.

Hg19 is the genomic construct that was employed in this analysis.

3.3 Overview of the Enhanced G4Catchall Algorithm with the Inclusion of the Secondary Structures within Extreme Loops

The G4 Catchall algorithm has been modified, and the Enhanced G4 Catchall method now takes into account secondary structures created within the extreme loops of G-quadruplex (G4) motifs. This improvement enables a more thorough examination of G4 structures, accounting for structural features other than the primary sequence pattern.

The extreme loops of G4 motifs, sometimes referred to as lateral loops, were treated as flexible areas lacking precise structural information in the initial G4 Catchall method. It has been shown, nevertheless, that these loops occasionally take on secondary structures like hairpins or bulges, which can affect the stability and usefulness of G4 structures.

With the help of the improved approach, secondary structures within the extreme loops of G4 motifs may now be found and described. By carefully examining the possibilities for base pair creation and stem-like structures within the areas of the loop, this procedure is completed. In order to determine the likelihood of secondary structure creation, the algorithm analyzes the loop sequences for the presence of complementary nucleotide pairs and calculates a secondary structure score (Arora and Maiti, 2009).

The quantity and stability of base pairs, the length and placement of the loop sequences, and the sequence context around the loops are some of the variables used to calculate the secondary structure score. Higher scores suggest a greater likelihood that secondary structures will grow inside the extreme loops.

The improved algorithm predicts G4 motifs more precisely by taking into account both the primary sequence pattern and probable secondary structure elements by including the examination of secondary structures. The accuracy of G4 forecasts improves as a result, and it also helps us understand a wider variety of G4 configurations.

The presence and functional significance of these structures must still be confirmed through experimental validation, despite the fact that the improved method offers useful insights into the putative secondary structures within G4 motifs. The program acts as a computational tool to direct future research and determine which areas should be prioritized for experimental analysis.

The algorithm is programmed using Python 3.9.2 in such a way that the search rules such as the number of G-paths and the limitations of loops, etc. can be specified by parameters.It was designed to create a folding profile for an RNA sequence using the Biopython module and the ViennaRNA software package.

The function CallRNAfold runs with the parameters filename and temp, where temp is set to 37. The aforementioned function accepts two parameters.

The term "filename" refers to the name of the file that contains RNA sequences and structures.

Temperature (optional): Signifies the specific temperature at which the prediction is to be conducted. The default value for the temperature is set at 37 degrees Celsius.

The variable "directory" is used to store the path to the directory where the ViennaRNA Package is installed.

Execute the command to open a file named "output.txt" in write mode.The method close() is utilized in this line to open a file named "output.txt" in write mode ("w") and promptly terminate the file connection. This process involves either generating or erasing the contents of the file in order to facilitate its future utilization.

If the file "temp_output.fold" exists in the operating system's path. The function os.remove("temp_output.fold") is used to delete the file named "temp This condition evaluates the presence of a file named "temp_output.fold" within the current directory. If the condition is met, this line of code will delete or remove it. The purpose of this step is to verify the absence of a pre-existing file with an identical name prior to the generation of a new file through the RNAfold prediction process.

The line `print(os.getcwd())` is responsible for outputting the current working directory to the console. This feature can be advantageous for the purpose of debugging, as it provides visibility into the specific directory from which the script is being executed.

The line of code employs the subprocess.run() function to perform the RNAfold program. The following is an analysis of the parameters that are supplied to it:

The provided command-line arguments correspond to the execution of the RNAfold program. The aforementioned items encompass:

Please provide the file path of the RNAfold executable, including the directory in which it is located.

The "noconv" option in RNAfold allows for the exclusion of the conversion of non-standard characters present in the input file.

The "-i" option is used to designate the input file for the RNAfold program. The filename parameter represents the specific file that is supplied as an argument to the function.

The temperature for the prediction can be set using the "--temp" parameter. The value is transformed into a string using the str() function.

The parameter "--outfile=temp_output.fold" is used to designate the name of the output file in which RNAfold will store the outcomes of its predictions.

The output file "temp_output.fold" is opened for reading.The method `read()` is utilized in this line to open the file named "temp_output.fold" in read mode ("r"). The contents of the file are then read and stored in the variable called `output`. The retrieval of the RNAfold prediction findings is performed by the script.

The output is represented as rows.The function "split("\n")" is utilized to divide the contents of the output variable into a collection of strings, where each string corresponds to a line extracted from the file. Each line of the text is delimited by newline letters ("\n").

The given line of code, `description = rows[2]`, is responsible for extracting the third line, identified by its index 2, from the list of rows. Subsequently, the extracted line is assigned to the variable named "description". This particular line commonly encompasses details pertaining to the secondary structure of RNA.

The score is calculated as a floating-point number obtained by extracting the numerical value from a string that matches the regular expression pattern "\([ ]*[+-]?[0-9]*[.]?[0-9]+The regular expression [0-9]+\.? can be used to match one or moreThe last element of the second row, obtained by applying a regular expression pattern to the string, is extracted and then its first and The provided code snippet utilizes regular expressions, namely the re.findall() function, to do a search for numeric values enclosed within parentheses within the description line. The program retrieves these numbers, turns the final one into a floating-point data type, and subsequently assigns it to the variable named "score". The aforementioned value denotes the numerical representation of the RNA secondary structure's score.

The value of DBN is obtained by removing any numerical values included in parentheses from the string in the third element of the rows list. The provided code snippet utilizes regular expressions, namely the re.sub() function, in order to eliminate numerical numbers included within parentheses from the description line. The outcome, denoting the Dot-Bracket Notation (DBN) structure, is allocated to the DBN variable.

The value of the element at index 2 in the rows array is obtained by removing any leading or trailing whitespace from the DBN string. The third line (description) in the rows list is modified by this code snippet, which updates it with the cleaned DBN structure. The cleaning process involves removing any leading and trailing whitespace.

The provided line of code executes a loop that iterates over the elements in the "rows" list and prints each element to the console. The utilization of this approach can prove to be advantageous in the process of debugging and comprehending the material encapsulated within the "temp_output.fold" file.

Ultimately, the function yields a tuple that encompasses the score and the changed list of rows. The score denotes the prediction score for the RNA secondary structure, whereas the rows correspond to the lines extracted from the "temp_output.fold" file.

Within the confines of the context manager, the function iterates over the split_seq_pos list. For every element within this enumerated collection (comprising a tuple consisting of a level and positions):

The program generates a header line that incorporates data pertaining to the present level, temperature, and score. The formatting of this header is as follows: >Level: {level} | Temperature: {temperature} | Score: {score:.2f}. The header serves the purpose of providing contextual information for the written material.

The header line is written to the output file using the output_file.write() method.

Subsequently, the system generates a textual output consisting of the recorded positions. The positions are concatenated into a string using the method ', '.join(map(str, positions)), so enhancing their legibility and facilitating their writing to the file.

Upon completing the header and positions for a particular level, the function proceeds to generate a blank line by executing the command output_file.write("\n"). The presence of an empty line functions as a demarcation point, effectively delineating several tiers of data within the output file, hence enhancing visual clarity.

The iteration persists until all levels within the split_seq_pos have been executed.

Ultimately, the context manager assumes the responsibility of properly closing the output file.

In brief, this function processes structured data pertaining to split sequences, corresponding temperature values, and scores, and transforms them into a coherent manner that can be easily comprehended. Subsequently, the formatted data is written to an output file. In order to enhance clarity, an empty line is utilized to separate each level of split sequences inside the output file.The file named "temp_output.fold".

The function initializes a list named "ranges" with no elements, which will be used to hold the temperature values that are generated.

The temp variable is assigned the value of the starting temperature (start).

The program enters a while loop that iterates until the value of the variable "temp" becomes smaller than the value of the variable "end". The iteration construct is responsible for generating the temperature values within the defined range.

Within the iteration:

The current value of the variable "temp" is appended to the list "ranges" using the "ranges.append()" method.The function "append(temp)" is executed. The temperature is appended to the existing list.

The temp variable is incremented by the step value (temp += step), so facilitating its convergence towards the final temperature.

Upon completion of the loop, the function will return the ranges list, which encompasses the temperature values that have been generated.

In brief, this function generates a sequence of temperature values commencing from a designated temperature, increasing by a specified increment, and concluding just prior to a specified concluding temperature.

The function utilizes a context manager (with open(output_filename, "w") as output_file) to open the output_filename in write mode ("w"). This guarantees that the output file is appropriately managed and closed subsequent to the writing process.

The algorithm sequentially processes each temperature measurement within the specified temperature range. For any given temperature:

Within the iteration of the loop pertaining to a particular temperature:

The temporary filename is generated using the format "output_{temp_value}.fasta". The given filename is associated with the output file that corresponds to the particular temperature in question.

The context manager is used to access the temporary file associated with the current temperature in read mode ("r").

The program generates a header line in the output file, denoting the temperature value as "Temperature: 10.0" for example. The purpose of this header is to facilitate the identification of the data that corresponds to each temperature.

The content of the temporary file is replicated into the output file. This process amalgamates the data extracted from individual files corresponding to specific temperature ranges into a unified dataset.

In conclusion, the temporary file is eliminated through the utilization of the os.remove(temp_filename) function, so facilitating the process of tidying up and liberating disk space.

The aforementioned procedure is iteratively executed for every temperature value within the temperature_range. As a consequence, an amalgamated output file is generated, encompassing data extracted from each individual output file corresponding to a certain temperature.

group_numbers Function:

This function is responsible for grouping a list of numbers into consecutive ranges. For example, given a list of numbers like [1, 2, 3, 5, 6, 7, 10], it will group them into ranges such as "1-3, 5-7, 10".

Parameters:

lst: A list of integers that you want to group into ranges.

The function initializes an empty list called groups to store the grouped ranges.

It also initializes start and end variables with the first element of the input list lst.The function then iterates through the elements of lst starting from the second element (index 1).

For each element, it checks whether it is consecutive to the previous element. If it is (i.e., the difference between the current element and the previous one is 1), it updates the end variable to the current element.

If the current element is not consecutive to the previous one, it checks whether the start and end are the same (indicating a single number) or different (indicating a range). Based on this, it appends the appropriate representation (either the single number or a range) to the groups list.

Finally, it joins all the groups into a comma-separated string and returns it.The function takes a list of numbers, groups them into consecutive ranges, and returns a string representing these ranges as comma-separated values.

In the context of computer programming, the statement "if __name__ == "__main__":" is commonly used to designate the main entry point of a program.

The aforementioned Python construct is a frequently used method to ascertain if the script is being executed directly as the primary program or if it is being imported as a module into another script. When the script is designated as the primary program, the code contained within this block will be executed.

The user's input string is stored in the variable "input_str".Kindly provide the initial temperature, final temperature (excluding), and temperature increment (separated by commas):

The user is prompted to provide a string of values related to temperature, separated by commas. The input() function is employed to obtain user input, and the resulting string is assigned to the variable input_str. The values of start_temp, end_temp, and temp_step are then extracted from input_str by splitting it at each comma and converting the resulting substrings to floating-point numbers.

This step involves the processing of the user input acquired in the preceding stage. This action performs the following functions:

The method input_str.split(",") divides the input string into many values by using a comma as the delimiter. This results in a list of values, such as ["start_temp", "end_temp", "temp_step"].

The function map(float, ...) is utilized to transform every element within the list into a floating-point number, which corresponds to a decimal number. The conversion of user inputs from strings to numerical values is a common practice due to the fact that user inputs are often represented as strings.

The variables start_temp, end_temp, and temp_step are allocated the corresponding values from the list.

The function "generate_temperature_ranges" is used to calculate a range of temperatures based on the provided parameters: the starting temperature (start_temp), the ending temperature (end_temp), and the temperature step (temp_step).

This portion of the code is responsible for reading RNA sequences from an input file, processing them, and preparing temporary files for further analysis.

inputfile is set to "inputs.fa," which is assumed to be the input file containing RNA sequences. You should replace this with the actual path to your input file.Two empty lists, inputnames and inputseqs, are initialized. These lists will be used to store the names and sequences of the RNA molecules read from the input file.

The script opens the input file ("inputs.fa") in read mode using a context manager (the with open(inputfile, "r") as f: statement).It then iterates through the lines of the input file using a for loop

For each line in the input file:

If the line starts with ">", it is assumed to be the name or description of the RNA sequence. In this case, a new empty string is appended to the inputseqs list, and the name is appended to the inputnames list.

If the line does not start with ">", it is assumed to be part of the RNA sequence. It is added to the most recent (last) element in the inputseqs list after stripping any leading or trailing white spaces.

After processing all lines in the input file, the script prints the length of the inputseqs list, which represents the number of RNA sequences read from the input file

Next, the script enters a loop over the RNA sequences in inputnames. For each sequence, it does the following:It creates a temporary output file named "example.fasta" using a with open(tempfile, "w") as f: statement.It writes the name (from inputnames) and the corresponding sequence (from inputseqs) into the temporary file.

A new variable split_output_filename is set to "split_seq_pos_output.txt."The script creates this file and immediately closes it. This file will be used to store information related to split sequences later in the code.

The aforementioned code invokes the generate_temperature_ranges method, passing the values acquired via user input as arguments. The function computes a series of temperature values by utilizing the given initial, final (non-inclusive), and incremental values. The variable temperature_range stores the range of temperatures that have been obtained.

In the context of the given temperature range, the variable "temp_value" iterates through each element.

The provided code snippet demonstrates a for loop that sequentially iterates over each temperature value within the temperature_range. The code block that follows will be executed for each temperature value.

The function CallRNAfold was invoked with the arguments "example.fasta" and temp_value, and the resulting values were assigned to the variables scr and o1. Within the iteration of the for loop, a function named CallRNAfold is runs by this particular line of code. The function is passed two arguments, namely "example.fasta". The provided input represents a filename or file path denoting a file titled "example.fasta."

The variable "temp_value" represents a temporary value. The present temperature value is derived from the temperature_range that is being iterated in the loop.

The function CallRNAfold is observed to execute computations pertaining to predictions of RNA folding and yields two values, denoted as scr and o1, which are subsequently allocated to variables.

The variables Seq and DBN are assigned the second and third lines from the o1 list, respectively. These lines correspond to the sequence and the dot-bracket notation of the predicted structure.

The empty lists splitSequences and splitSeqPos are initialized to store the split sequences and their positions.

The dictionaries openSequence and openSeqPos are initialized to store the open sequences and their positions at different levels. These dictionaries will help keep track of the sequences and positions as we iterate through the dot-bracket notation.

A loop iterates over each position in the dot-bracket notation (DBN) to identify the different levels and the corresponding open sequences. It examines each character in the dot-bracket notation to determine its type (open parenthesis, dot, or closing parenthesis) and takes appropriate actions.

If an open parenthesis is encountered, the code increases the curlvl variable, indicating the current level. It also appends "/" to the open sequence and initializes the position list for that level. This prepares to store the upcoming nucleotides for the open sequence.

If a dot is encountered, the code appends the corresponding nucleotide to the open sequence and adds its position to the position list for that level. The current method involves grouping together nucleotides and their matching locations within the open sequence at this time.

If a closing parenthesis is found, the program continues by decreasing the variable curlvl, which denotes a decrease in the level's magnitude.

It appends the open sequence and its positions to the splitSequences and splitSeqPos lists, respectively. This marks the end of the current open sequence at the current level. The dictionaries are cleared of the open sequence and position list corresponding to the current level once the relevant calculations have been completed in preparation for any upcoming open sequence at a lower level.

Any open sequences and their corresponding positions that weren't closed before the dot-bracket notation ended are then added to splitSequences and splitSeqPos, respectively, when this loop is finished. These open spots and sequences serve as a good indicator for those that are still open.

The code then iterates over the split sequences and checks if any of them contain "e". If they do, it further splits them at "e" and prints the position ranges of the secondary structures within the extreme loops. This step helps identify and analyze the extreme loops within the predicted structure.

The name of the output file is "split_seq_pos_output.txt".

The aforementioned code assigns the value "split_seq_pos_output.txt" to the variable output_filename. The designated file name for the storage of certain data will be determined.

The function "write_split_seq_pos_to_file" is designed to write the values of "temp_value", "splitSeqPos", and "scr" to a file specified by "output_filename".

The aforementioned line invokes the execution of a function named "write_split_seq_pos_to_file" by passing the subsequent arguments:

The variable "temp_value" represents the current temperature value that is being processed within the loop.

The splitSeqPos data structure encompasses position information.

The term "scr" refers to a numerical value that represents a score.

The output_filename refers to the designated name of the file in which data will be written.

The function is observed to generate and record data pertaining to position information, temperature, and score into the designated output file.

The output filename is defined as "output_{temp_value}.fasta".

The output_filename is assigned a formatted string that incorporates the current temperature value by replacing {temp_value}. As an illustration, when the value of temp_value is 37, the resulting output_filename is transformed into "output_37.fasta".

The production of the FASTA file is being initiated.

The subsequent code snippet initiates the opening of the output_filename in write mode ("w") by employing a context manager (with open(...)). The following code segment is responsible for the generation of an output FASTA file.

In this code block, a for loop is used to iterate through the elements of the splitSequences. It is probable that this particular data structure has pertinent information pertaining to RNA sequences.

Within the loop, the variable "i" serves as an index, while the variables "level" and "sequence" are extracted from each element of the "splitSequences" collection.

The coordinates are obtained from the variable splitSeqPos[i][1], which is presumably the positional information associated with the sequence.

The description string is structured in a manner that includes details pertaining to the level, temperature, and score.

The information is written to the output file in FASTA format using the output_file.write() method. > The user's text can be enhanced to have a more academic tone without adding any additional information.

The description and sequence are additionally outputted to the console for the purpose of display.

The aforementioned procedure yields a multitude of FASTA files, with each file corresponding to a distinct temperature value.

The process of combining output files.

Following the execution of the loop responsible for generating the output files, the code proceeds to invoke the merge_output_files function in order to consolidate all of these output files into a singular file denoted as "merged_output.fasta." The anticipated behavior of this function is the aggregation of output files that correspond to a certain range of temperature values into a single file.

CHAPTER 4: RESULT

In the written algorithm, Window approach was preferred in order to predict more clearly the G-quadruplexes that may occur in extreme loops. On the generated synthetic sequence, primarily Window size range was evaluated in three different ways small, middle, and large.

The tool we wanted to do is find and detect base ranges that can create hairpins in a string. But while taking the deltaG values as a base for this hairpin formation, we also need to set a threshold for it. And we used for Window approach to find out how to determine what was trash. Because when it divides each array into windows and looks at DeltaG, the values that come out change according to the size of the windows. The proximity and distance of the distances between the arrays change the deltaG calculation and the probability of detecting the locations where hairpins may occur. When it takes the window interval small, it can do a much more detailed calculation, but it is likely to miss the corresponding sequences that may create hairpins. However, when it makes the window gap large, many possibilities are missed again because the area becomes larger. We thought that more than one range scan should be done for this.

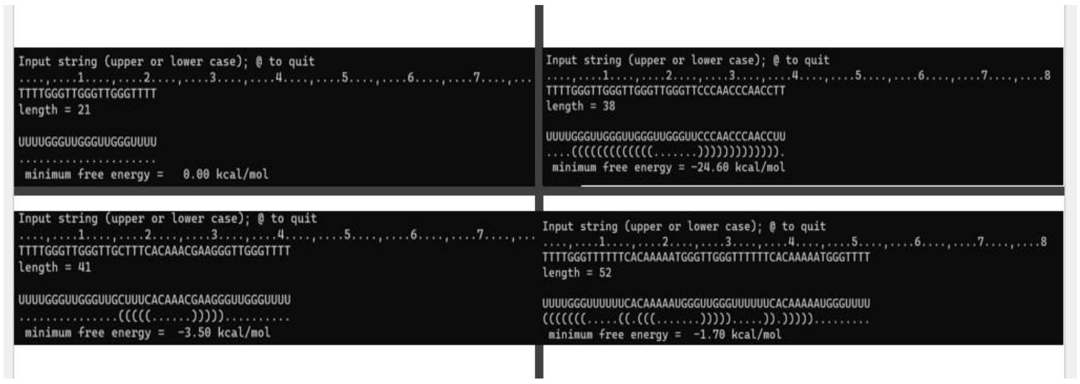
A) TTTTGGGTTGGGTTGGGTTTT

B) TTTTGGGTTGGGTTGCTTTCACAAACGAAGGGTTGGGTTTT

C)TTTTGGGTTTTTTCACAAAAATGGGTTGGGTTTTTTCACAAAAATGGGTTTT

D)TTTTGGGTTGGGTTGGGTTGGGTTCCCAACCCAACCTT

Figure 3. Synthetic sequences.



```
Input string (upper or lower case); @ to quit
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,...
TTTTGGGTTGGGTTGGGTTTT
length = 21

UUUUGGGUUGGGUUGGGUUUU
.....................
 minimum free energy =   0.00 kcal/mol
```

```
Input string (upper or lower case); @ to quit
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8
TTTTGGGTTGGGTTGGGTTGGGTTCCCAACCCAACCTT
length = 38

UUUUGGGUUGGGUUGGGUUGGGUUCCCAACCCAACCUU
....(((((((((((((.......))))))))))))).
 minimum free energy = -24.60 kcal/mol
```

```
Input string (upper or lower case); @ to quit
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,...
TTTTGGGTTGGGTTGCTTTCACAAACGAAGGGTTGGGTTTT
length = 41

UUUUGGGUUGGGUUGCUUUCACAAACGAAGGGUUGGGUUUU
...............(((((......)))))..........
 minimum free energy =  -3.50 kcal/mol
```

```
Input string (upper or lower case); @ to quit
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8
TTTTGGGTTTTTTCACAAAAATGGGTTGGGTTTTTTCACAAAAATGGGTTTT
length = 52

UUUUGGGUUUUUUCACAAAAAUGGGUUGGGUUUUUUCACAAAAAUGGGUUUU
(((((((.....((.(((.......))))).....)).)))))..........
 minimum free energy =  -1.70 kcal/mol
```

Figure 4. Synthetic sequences testing on RNAfold. Cannot find G-quadruplexes with test sequences with RNAFold.

The algorithm splits the given array into parts and assigns points to the resulting windows based on their DeltaG values. In order to observe how the intervals change according to their size, we examined the window size selections in three ways, 30, 50 and 90, and we obtained separate graphs for each window size. Trashold has changed constantly with window size. We created an average graph by dividing the window size value for each value in the profile. According to the graph obtained, based on the scores of the secondary structures that may occur regionally, the formation of hairpins, but the decrease in the scores when trashold is increased was observed. The graph gives us the average of 50 when window size 90 is selected, and the average of 30 when 50 is selected. Where secondary structures are high tells us they can form hairpins.

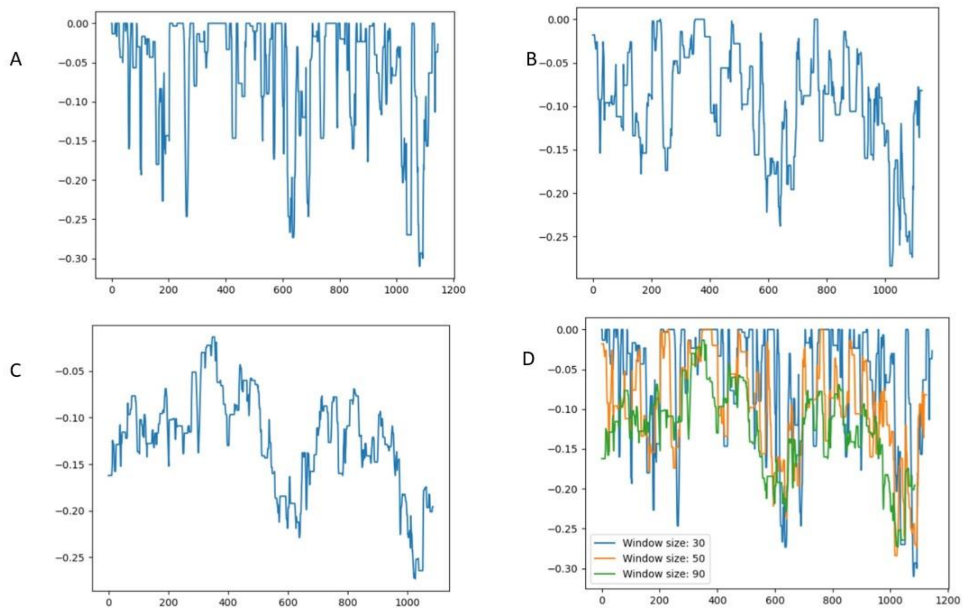
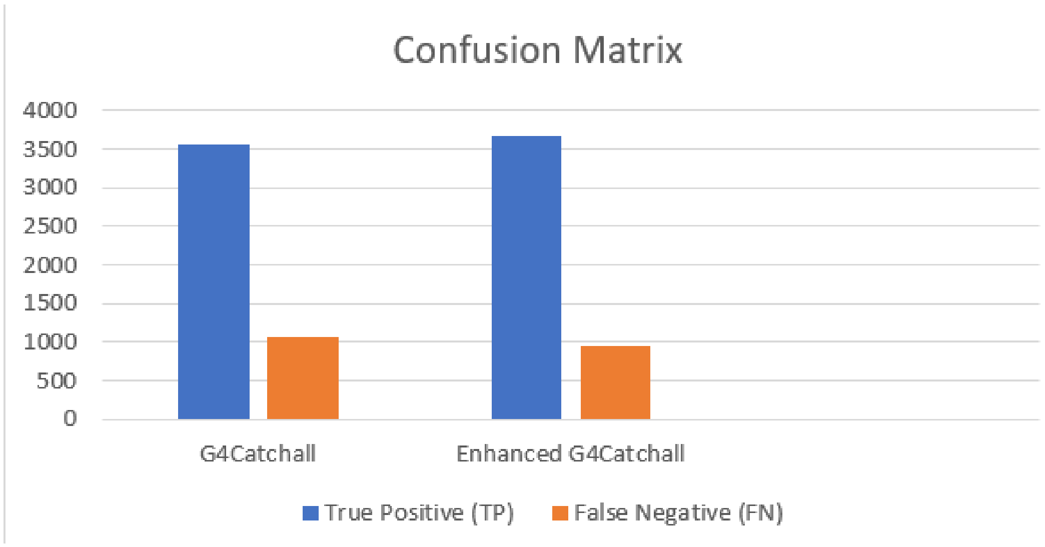


Figure 5. Window size plots of profiles A) Window size :30 B) Window size : 50 C) Window size: 90 D) Display 3 window size plots together.

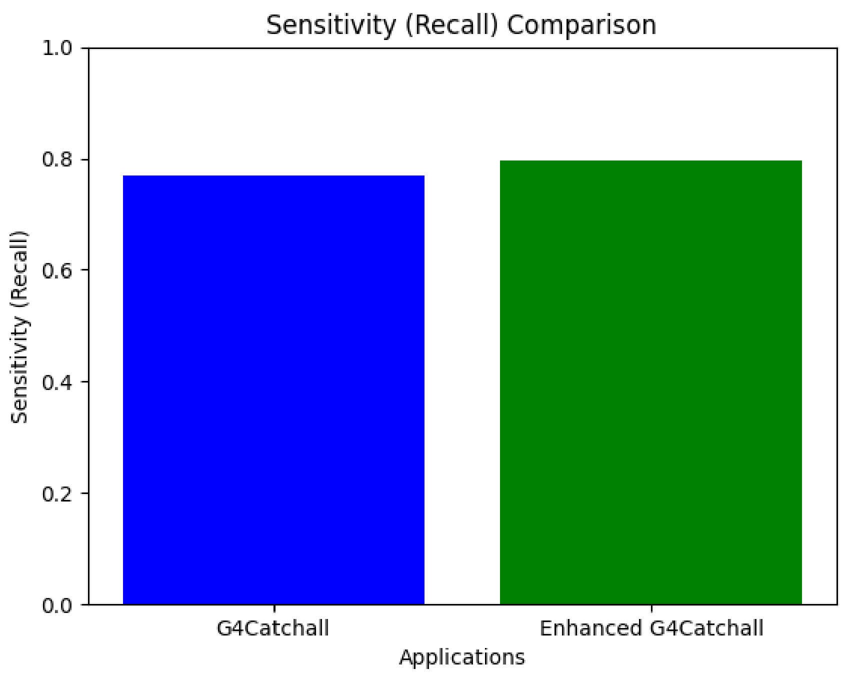
While continuing with the Window approach, 2 questions arose. The first one is which trashold/window to choose and the other one is that the peaks that can be caught when the window size is kept small will take the average of each peak instead of giving a clear one as the window will take the average of the regions as you get bigger. If the same trashold is used, the areas to be selected will be different and the ends that can form G4 will not come close to each other when the hairpin is opened. Besides the advantages of both window size ranges, the lack of a common trashold has led to the abandonment of the window approach. We decided to look at the part from the beginning to the end of each hairpin to see if the ends converge in the window. The algorithm was able to find the scoring for each window size, so we thought converting it to scoring for each hairpin would help avoid any gaps in between. Using a new strategy, we aimed to improve the algorithmic detection of experimentally discovered G-fours. We tried to achieve this by finding a whole directory, which we call the layer approach, and determining the level by starting to count when the hairpin is formed by examining the directory one by one (when the brackets are opened). . Levels are created according to DeltaG values in each opened series, and the chance of finding G4 in each level changes. In addition to just performing RNA folding calculations, the user can specify the temperature range. In this way, you can scan according to temp and observe the formations in between. The files obtained according to Temp were combined into a single file for ease of analysis. The final output was then given to the Enhanced G4Catchall algorithm. Then, a comparison was made with the results obtained for both applications on Hg19 chr21, which has been experimentally proven to have G-quadruplex. As a result of this comparison, out of 4617 entries, G4Catchall detected 3553, while Enhanced G4Catchall detected 3677. However, the reason why the negatives cannot be viewed is that the RNAfold, which the algorithm uses as a parameter, is too long and cannot be read. Confusion Matrix interpretation was attempted by looking at the True Positive (TP) and False Negative (FN) values of both algorithms.

Table1. Confusion Matrix.



When calculation is made for Recall(Sensitivity) from these values; From the TP/TP+FN ratio, it was found to be 0.76954 for G4Catchall, while 0.79640 was observed for Enhanced G4Catchall.

Table 2. Recall Calculation Analyses. (TP/TP+FN)



CHAPTER 5: DISCUSSION

The aim of this study is to predict G-quadruplexes that may occur on extreme loops and to develop the G-4Catchall algorithm. The G4Catchall algorithm is a tool that currently incorporates the innovation of considering atypical feature types and applying separate rules for shorter GQs. In this study, we wanted to consider extreme loops in addition to these features. Based on many web servers and algorithms developed on the estimation of G-quadruplexes, an algorithm was written using and changing the parameters of the RNAFold algorithm.

Considering the previous studies, the probability that a hairpin structure will prefer to be G4 when opened may depend on its stability. The stability of the structure can be found by comparing it with the stability of G4. But here it is not easy to find the stability of the G4 while you can find the stability of the structure as DeltaG. Because it is not possible to predict which G4 will be based on. In this proposed solution, in order to compare the two, open G4s in the whole sequence can be detected and compared with the structure that is likely to form G4.

CHAPTER 6: CONCLUSION

As confirmed by the literature, our study showed that it is reasonable to give a temperature range based on the results obtained, even if there is a low increase. The G4Catchall algorithm was already looking at Level 0 values. With this study, by giving the temperature range and looking at the openings that occur at other temperatures and levels other than Level 0, we can say that it is already difficult to find G-quadruplex and therefore there is no other alternative.

More experimentally validated sequences are needed to better assess the accuracy of the algorithms. The study highlights the importance of the stability of extreme loops on G4 formation. We think that ongoing experiments and future studies may play a role in identifying potential G-quadruplex formation sequences and thereby enabling the advancement of many biological processes and therapeutic applications.

REFERENCES

Agarwal, T., Roy, S., Kumar, S., Chakraborty, T. K. and Maiti, S., (2014) '*In the Sense of Transcription Regulation by G-Quadruplexes: Asymmetric Effects in Sense and Antisense Strands*', Biochemistry, Vol. 53, pp 3711-3718.

Arora, A. and Maiti, S. (2009) '*Stability and Molecular Recognition of Quadruplexes with Different Loop Length in the Absence and Presence of Molecular Crowding Agents'*, The Journal of Physical Chemistry B, Vol. 113, pp 25.

Bedrat, A., Lacroix, L. and Mergny, J.-L. (2016) *'Re-evaluation of G-quadruplex propensity with G4Hunter'*, Nucleic Acids Research, Vol. 44, pp 4.

Brázda, V., Kolomaznik, J., Lysek, J., Bartas, M., Johta, M., Stastny, J. and Mergny, J.-L., (2019) *'G4Hunter web application: a web server for G-quadruplex prediction'*, Bioinformatics, Vol. 35, pp 3493-3495

Chambers, V. S., Marsico, G., Boutel, J. M., Antonio, M. D., Smith, G. P. and Balasubramanian, S., (2015) *'High-throughput sequencing of DNA G-quadruplex structures in the human genome',* Nature Biotechnology, Vol. 33, pp 877 -881.

Chen, J., Cheng, M., Salgado, G. F., Stadlbauer, P., Zhang, X., Amrane, S., Guedin, A., He, F., Sponer, J., Ju, H., Mergny, J.-L. and Zhou,J., (2021) *'The beginning and the end: flanking nucleotides induce a parallel G-quadruplex topology'*, Nucleic Acids Research, Vol. 49, pp 9548-9559.

Das, P., Ngo, K. H., Winnerdy, F. R., Maity, A., Bakalar, B., Mechulam, Y., Schmitt, E. and Phan, A. T. (2021) *'Bulges in left-handed G-quadruplexes',* Nucleic Acids Research, Vol. 49, pp 1724-1736.

Garant, J.-M., Perreault, J.-P. and Scott, M. S. (2017) *'Motif independent identification of potential RNA G-quadruplexes by G4RNA screener'*, Bioinformatics, Vol. 33, pp 22.

Grün, J. T. and Schwalbe, H. (2022) *'Folding dynamics of polymorphic G-quadruplex structures'*, Biopolymers, Vol. 113, pp 1.

Huppert, J. L. (2005) *'Prevalence of quadruplexes in the human genome',* Nucleic Acids Research, Vol. 33, pp 9.

Kaplan, O. I., Berber, B., Hekim, N. and Doluca, O., (2016) '*G-quadruplex prediction in E. coli genome reveals a conserved putative G-quadruplex-Hairpin-Duplex switch',* Nucleic Acids Research.

Kikin, O., D'Antonio, L. and Bagga, P. S. (2006) *'QGRS Mapper: a web-based server* *for predicting G-quadruplexes in nucleotide sequences'*, Nucleic Acids Research, Vol. 34.

Kwok, C. K. and Merrick, C. J. (2017) *'G-Quadruplexes: Prediction, Characterization, and Biological Application',* Trends in Biotechnology, 35(10).

Lorenz, R. *et al.* (2011) *'ViennaRNA Package 2.0'*, Algorithms for Molecular Biology, Vol. 6, pp 1.

Mattick, J. S., Amaral, P. P., Carninci, P., Carpenter, S., Chang, H. Y., Chen, L.-L., Chen, R., Dean, C., Dinger, M. E, Fitzgerald, K. A., Gingeras, T. R., Guttman, T., Hirose, T., Huarte, M., Johnson, R., Kanduri, C., Kapranov, P., Lawrance, J. B., Lee, J. T., Mendell, J. T., Mercer, T. R., Moore, K. J. , Nakagawa, S., Rinn, J. L., Spector, D. L., Ulitsky, I., Wilusz, J.E. and Wu, M., (2023) *'Long non-coding RNAs: definitions, functions, challenges and recommendations'*, Nature Reviews Molecular Cell Biology, Vol. 24, pp 430-447.

Monti, P., Brazda, V., Bohalova, N., Porubiakova, O., Menichini, P., Speciale, A., Bocciardi, R., Inga, A. and Fronza, G., (2021) *'Evaluating the Influence of a G-Quadruplex Prone Sequence on the Transactivation Potential by Wild-Type and/or Mutant P53 Family Proteins through a Yeast-Based Functional Assay',* Genes, Vol. 12, pp 2.

Pavlova, A. V., Kubareva, E. A., Monakhova, M.V., Zvereva, M. I. and Dolinnaya, N. G., (2021) *'Impact of G-Quadruplexes on the Regulation of Genome Integrity, DNA Damage and Repair'*, Biomolecules, Vol. 11, pp 9.

Puig Lombardi, E. and Londoño-Vallejo, A. (2020) *'A guide to computational methods for G-quadruplex prediction',* Nucleic Acids Research, Vol. 48, pp 1.

Radecki, P., Uppuluri, R., Deshpande, K. and Aviran, S., (2021) *'Accurate detection of RNA stem-loops in structurome data reveals widespread association with protein binding sites',* RNA Biology, Vol. 18, pp 1.

Ravichandran, S., Razzaq, M., Parveen, N., Ghosh, A. and Kim, K.-K., (2021) *'The effect of hairpin loop on the structure and gene expression activity of the long-loop G-quadruplex',* Nucleic Acids Research, Vol. 49, pp 10689-10706.

Reuter, J. S. and Mathews, D. H. (2010) *'RNAstructure: software for RNA secondary structure prediction and analysis',* BMC Bioinformatics, Vol. 11, pp 1.

Rhodes, D. and Lipps, H. J. (2015) *'G-quadruplexes and their regulatory roles in biology',* Nucleic Acids Research, Vol. 43, pp 18.

Risitano, A. (2004) *'Influence of loop size on the stability of intramolecular DNA* *quadruplexes',* Nucleic Acids Research, Vol. 32, pp 8.

Statello, L., Guo, C.-J., Chen, L.-L. and Huarte, M., (2021) *'Gene regulation by long non-coding RNAs and its biological functions'*, Nature Reviews Molecular Cell Biology, Vol. 22, pp 96-118.

Sun, Z.-Y., Wang, X.-N., Cheng, S.-Q., Su, X.-X. and Ou, T. -M., (2019) *'Developing Novel G-Quadruplex Ligands: from Interaction with Nucleic Acids to Interfering with Nucleic Acid–Protein Interaction'*, Molecules, Vol. 24, pp 3.

Wayment-Steele, H. K., Kladwang, W., Strom, A. I., Lee, J., Treuille, A., Becka, A., Participants, E. and Das, R., (2022) *'RNA secondary structure packages evaluated and improved by high-throughput experiments'*, Nature Methods, Vol. 19, pp 1234-1242.

Weskamp, K. and Barmada, S. J. (2018) *'RNA Degradation in Neurodegenerative Disease'*, in.

Winnerdy, F. R., Bakalar, B., Maity, A., Vandana, J. J., Mechulam, Y., Schmitt, E. and Phan, A. T., (2019) *'NMR solution and X-ray crystal structures of a DNA molecule containing both right- and left-handed parallel-stranded G-quadruplexes'*, Nucleic Acids Research, Vol. 47, pp 8272-8281.

You, J., Li, H., Lu, X., Li, W., Wang, P., Dou, S. and Xi, X., (2017) *'Effects of monovalent cations on folding kinetics of G-quadruplexes',* Bioscience Reports, Vol. 37, pp 4.

Zadeh, J. N., Steenberg, C.D., Bois, J.S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M. and Pierce, N. A., (2011) *'NUPACK: Analysis and design of nucleic acid systems'*, Journal of Computational Chemistry, Vol. 32, pp 1.

Zhang, J., Harvey, S. E. and Cheng, C. (2019) *'A high-throughput screen identifies small molecule modulators of alternative splicing by targeting RNA G-quadruplexes'*, Nucleic Acids Research, Vol. 47, pp 7.

Zhang, R., Shu, H., Wang, Y., Tao, T., Tu, J., Wang, C., Mergny, J. and Sun, X., (2023) *'G-Quadruplex Structures Are Key Modulators of Somatic Structural Variants in Cancers'*, Cancer Research, Vol. 83, pp 8.