G-Quadruplex Identification Working Group

University of Maryland University College

BTMN 670-9042

Fall 2015

Masrur Alam

Lindsay Cook

Tiffany Younger

Sarah Woodruff

**Table Of Contents**

Background and Purpose………………………………………………………………………….3

* GFinder
* Halobacterium salinarum

Background on Regular Expressions…………………………………………………………..….3

Development of Regular Expression to Identify Candidate G-Quadruplex Sequences……....…..4

Eclipse, Java and JAR Files……………………………………………………………………….9

GFinder: Installation and Use………………………………………………………………..…..14

GFinder: Annotated Output……………………………………………………………………...19

Resources for Independent Study…………………………………………………………….….21

* Regular Expression Resources
* Java Learning Resources
* Java Installation Resources
* BioJava Package Resources

Bioinformatics in the Applied Biological Sciences Degree Program…………………………....25

References…………………………………………………………………………………….….29

Appendix A: Listing of Output Files From GFinder Search…………………………….………33

Appendix B: Listing of JAR Files……………………………………………………………….33

**Background and Purpose**

**GFinder**

The G-quadruplex Identification Group has collaborated with Dr. Linda DeVeaux of the Department of Chemistry and Applied Biological Sciences at the South Dakota School of Mines and Technology (SDSM&T) to develop an analytical tool, GFinder, to identify candidate G-quadruplex sequences specifically in the genome of Halobacterium salinarum sp. NRC-1 with the broader intent of being able to use the tool on other genomes and in explorations of variable loop length. Dr. DeVeaux is a microbiologist by training and her area of focus over the last several years has become extremophiles, specifically now archaeon Halobacterium salinarum and in particular strain NRC-1. At SDSM&T, she also works closely with Dr. Richard Sinden, an expert in G-quadruplex structures whose focus has been more on their role in eukaryotic cells, who provided additional background on these structures for the G-quadruplex Identification Group.

In constructing GFinder, the G-Quadruplex Identification Working Group identified the following tasks:

* Creation of an interface with relevant user inputs (accomplished via Java)
* Flexibly identify candidate G-quadruplex sequences (accomplished via regular expression searching)
* Compare candidate sequences to currently identified genes (accomplished via BioJava)
* Produce structured output to facilitate further exploration with candidate sequences (accomplished via CSV)

**Halobacterium salinarum**

A predominant feature of Halobacterium salinarum is its ability to resist radiation and continue to grow (DeVeaux, et al., 2007); this supports a model of robust DNA repair being possible in its genome. Additionally, these bacteria have a high G-C content in their genome (DeVeaux, et al., 2007). Given that G-quadruplex structures are often found in guanine-rich genomes and play a role in genomic stability as well possibly transcriptional regulation, the goal of GFinder is to explore how these structural features may overlap to allow G-quadruplex sequences to contribute to radioresistance. G-quadruplexes form in areas where the bases associate via Hoogsteen hydrogen bonding forming planes that can then stack (Sen & Gilbert, 1988). The loops between the guanine runs can vary in length from a widely accepted model of up to seven nucleotides to in some cases as many as 30 (Guedin, Gros, Alberti, & Mergny, 2010). Thus the GFinder tool is structured to allow the user to determine what genome will be searched for candidate G-quadruplex sequences, what the maximum length of the loop in those candidates should be and against what genome the subsequence BLAST search should then occur before producing the final report and output.

**Background on Regular Expresssions**

Regular expressions are present in a wide range of scripting computer languages (Perl, Java, Python, etc.) and serve as a powerful programming tool using algebraic notation (Medeiros, Mascarenhas, & Ierusalimschy, 2014) (Campeanu & Santean, 2009). The syntax of regular expressions specify a pattern a user is searching for in a string or a full string against a list of strings (Medeiros, Mascarenhas, & Ierusalimschy, 2014). A modification of regular expressions called the extended regular expressions with back-references (regex), creates a means of creating expressions for patterns (repetitions) by performing a usual pattern match, but then “backtracking when a particular path through the expression makes the match fail”. (Schmid, 2013) (Medeiros, Mascarenhas, & Ierusalimschy, 2014). This backtracking or back-referencing refers to the revisiting of the earlier subexpression when a failure to match occurs, to provide a more accurate pattern search throughout a string (Schmid, 2013). “For example, r:=( 1 (a|b)\*)1c\1 is a regex, where \1 is a backreference to the referenced subexpression in between the parentheses (1 and)1 (Schmid, 2013). In this example r denotes the set within the string, a|b are alterations of the search pattern, and c denotes the frequency (Schmid, 2013). Ad-hoc optimizations are used to reduce the amount of backtracking of regex implementation, which in turn decreases the running time of the pattern search (Medeiros, Mascarenhas, & Ierusalimschy, 2014).

When implementing regex as a programming tool, there are specifics in syntax which are dependent on the programming language and in the case of the GFinder tool, the language selected was Java. When regex is introduced within a class, or step, in Java, the pattern is first specified as a string, then compiled into an instance of the class (Oracle, 2014). The pattern is then “used to create a Matcher object that can match arbitrary character sequences against the regular expression” (Oracle, 2014). Figure 1 is an example of a typical regex invocation sequence.

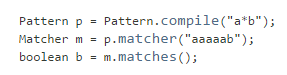


Figure 1: An example of a typical regex invocation sequence. (Oracle, 2014)

**Development of Regular Expression to Identify Candidate G-Quadruplex Sequences**

Development of the GFinder tool required research into what G-quadruplex structures are and what types of sequences are most likely to form them as well as needing to understand how G-quadraplex sequences were typically located within a given genome. This work identified that DNA sequences containing four or more closely spaced guanine-tandems can fold to form intramolecular G-quadruplexes and that these intramolecular G-quadruplexes consist of stacked G-quartets that are linked by three loops between the four G-strands (Balasubramanian, Hurley, & Neidle, 2011). In most of the literature reviewed the predominating sequence motif that can be found on either DNA strand and that carries the most potential for G-quadruplex formation is defined as G3+N1-7G3+N1-7G3+N1-7G3+. This was the key algorithmic element in determining how to structure a regular expression to search the full genome of the NRC-1 strain of archaeon Halobacterium salinarum for potential G-quadraplex sequences even though there were similar algorithm motifs known to form quadruplex sequences such as G3+N1-11G3+N1-11G3+N1-11G3+, which would extend the expected loop size but was also shown to reduce stability (Smith, 2010). In one study using a program based on the (G3+N1-7G3+N1-7G3+N1-7G3+) algorithm it was observed that the frequency or probability of each nucleotide upstream (–ve) or downstream (+ve) of the transcription start site is part of a “putative G-quadruplex-forming sequence (PQS)” (see the density plot in Figure 2) (Balasubramanian, 2011, p. 27). This data was averaged over all human protein coding genes in the genome.

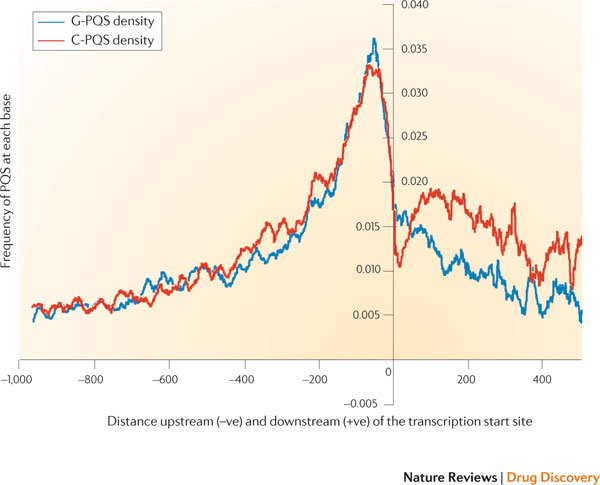


Figure 2: (Balasubramanian, 2011, p. 27)

In a similar study, biological sequences associated with known genes were also observed to contain the G3+N1-7G3+N1-7G3+N1-7G3+ algorithm motif in vitro. (see Figure 3)

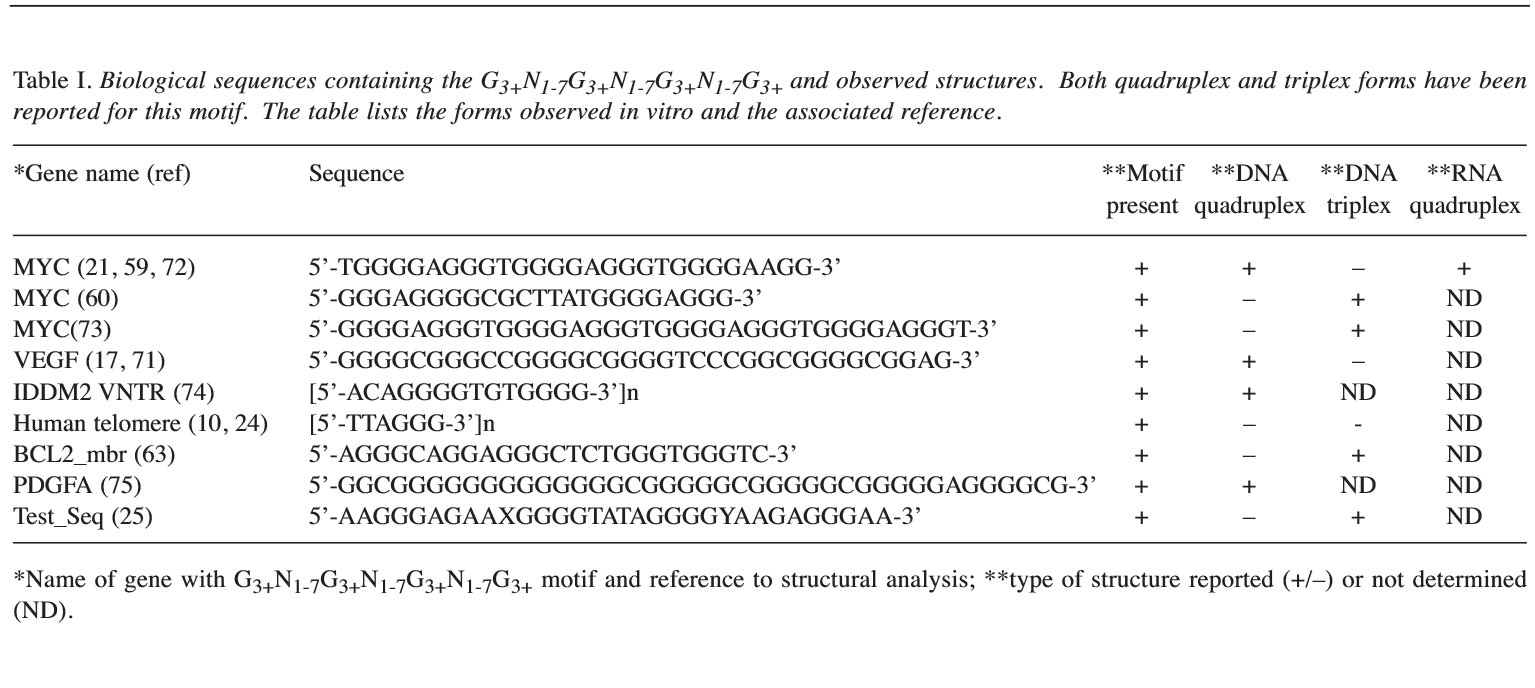


Figure 3: (Smith, 2010, p. 209)

Another study done using the NMR determined structures of gene promoter G-quadruplexes listed the sequences and structures of selected gene promoter G-quadruplexes in Figure 4 that also follow the algorithm G3+N1-7G3+N1-7G3+N1-7G3+ motif.

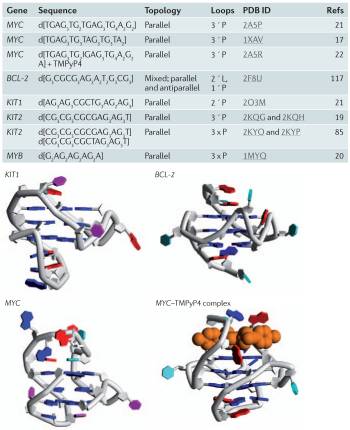


Figure 4: (Balasubramanian, Hurley & Neidle, 2011, p. 265)

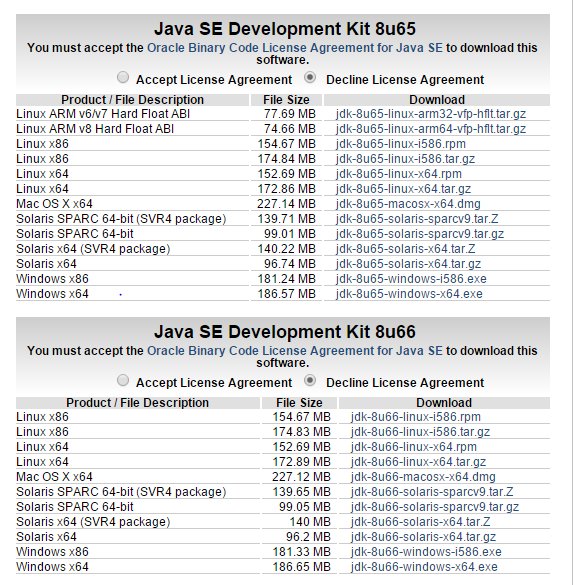
Using the G3+N1-7G3+N1-7G3+N1-7G3+ algorithm, the G-Quadruplex Identification Working Group developed a regular expression (or regex), which is the basis for parsing or searching for patterns in text in many coding languages.  To then program this in Java, the language determined to be the best fit for developing the GFinder tool, the syntax of this regular expression became ([gG]{3,}\\w{1,7}){3,}[gG]{3,} to match the G3+N1-7G3+N1-7G3+N1-7G3+ pattern for any sequence length. Within the first section of the expression, indicated by the parenthesis, the first set of hard brackets dictates that the program should search for G regardless of case and the braces indicate that it should look for a string of at least three of that character. Then the “\\w” indicates that we want the program to find a string of any letter up to a total of seven as indicated by the set of numbers in the next set of braces. After the parenthesis we are looking for the first section to repeat at least three times as indicated by the three in the next set of braces. The final set of brackets indicates that we are once again looking for a run of three or more Gs regardless of case to conclude the regex.

Various studies done on G-quadruplexes have provided a lot of useful information that has been used to computationally predict where significant G-quadruplex forming sequences may be located within a particular genome. There are several algorithms that have been developed using different criteria to identify sequences that are most likely to fold into G-quadruplex in vitro (Di Antonio, 2012). One such algorithmic approach is based on the density of guanine runs or tandem guanine regions within a sequence window and has been associated with the likelihood of G-quadruplex formation (Di Antonio, 2012). Programs that are based on such algorithms are widely used, and can provide valuable information about how predicted G-quadruplex motifs are distributed throughout the genomes of different organisms (Di Antonio, 2012). It has also been observed that the analysis of the data can become more complicated when one takes into account that guanine-rich sequences often display multiple folding possibilities which affects the apparent loop sizes which in turn affects the algorithm (Di Antonio, 2012).

**Eclipse, Java and JAR Files**

To be able to enhance, troubleshoot, and modify GFinder in any way, the use of an integrated development environment (IDE) is needed. Eclipse is the IDE our group used in the development of the GFinder tool. In order to manipulate and test the algorithm or associated methods, proper installations of this IDE, along with a Java Development Kit (JDK) is essential. All instructions provided are based on Windows operating systems.

Before installing Eclipse, a user must first download a JDK. The Java SE download link is http://www.oracle.com/technetwork/java/javase/downloads/index.html and its commercial installation guide can be found here: http://docs.oracle.com/javase/8/docs/technotes/guides/install/install\_overview.html. When on the Java SE downloads home page click on the download button under JDK in the section labeled Java SE 8u65 / 8u66 (or the most up-to-date version) (Figure 5). The next page takes the user to a variety of downloads. The user should choose the proper download, downloading the specified version of Windows, under the Java SE Development Kit 8u65 section (or the most up-to-date version)( Figure 5). If a user is unsure about a Windows computer’s operating system version, it can be checked by right-clicking on “This PC” or “My Computer” from the hard drive application and choosing Properties. Use the Java platform, standard edition installation guide found for a step by step process for proper downloading. Also, save the download file in a specific folder to find to for future use. The designation of a folder for the GFinder tool is discussed in the next section, GFinder: Installation and Use, and saving the guide to that location would be recommended in order to keep all resources in the same location.



c

c

c

Figure 5: Java SE download instructions.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Once the JDK is downloaded and installed, Eclipse can then be downloaded and installed from http://www.eclipse.org/downloads/. There are many IDEs found on the Eclipse downloads home page. Scroll to the Eclipse package labeled, “Eclipse IDE for Java Developers” and click on the appropriate download for type of Windows operating system as mentioned previously (Figure 6). Choose the default mirror download found in the gray and orange box on the left side of the page (Figure 6). This download should be saved to a familiar folder for installation.

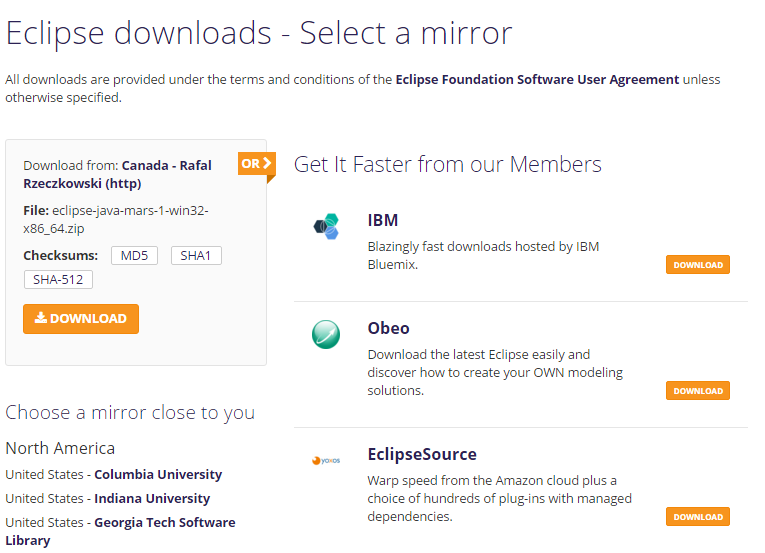
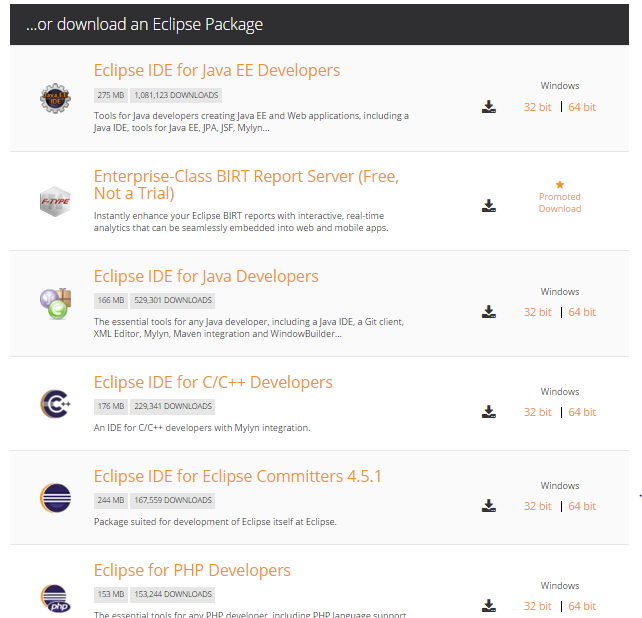


Figure 6: Eclipse IDE for Java Developers download instructions.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

To install Eclipse, the ZIP file of the Eclipse download should be unzipped first. To unzip a file, right-click the file, select the WinZip command, then click “Extract to Here”. This creates a folder named “Eclipse”. Create a shortcut of the Eclipse.exe file onto your desktop for convenience of setup.

To start the setup process of Eclipse, double click the shortcut. A workspace launcher window should populate in which the pathway where all Java projects will be saved is designated. Accepting the default location would be beneficial if you are a beginning Java user; after setup, the file pathway can be found by selecting properties under the file tab in this application. Make sure the box labeled “use this as the default and do not ask again” is checked before clicking OK. After installing Eclipse, restart your computer. Once your system is restarted, open Eclipse and exit out of the welcome message (Figure 7).

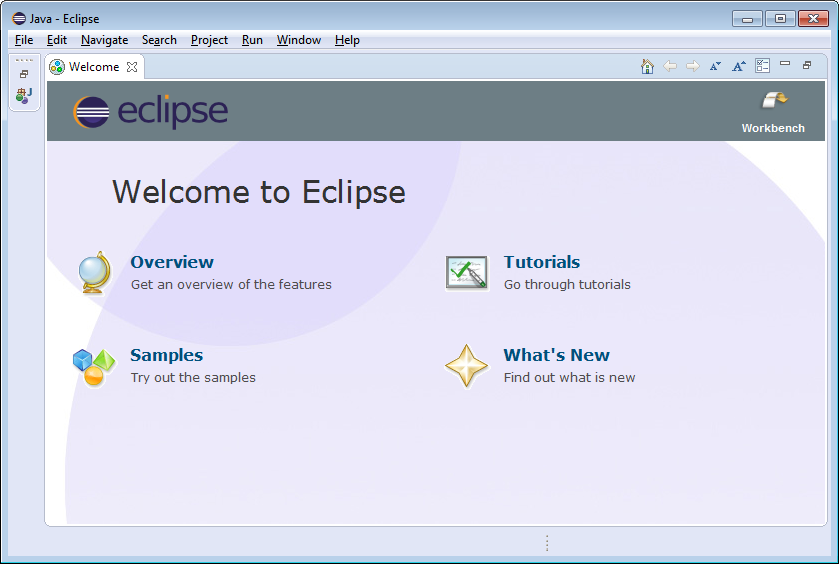


Figure 7: Eclipse IDE for Java Developers welcome screen.

http://pages.cs.wisc.edu/~cs302/labs/EclipseTutorial/images/Welcome.png

The GFinder tool has been provided via an executable JAR file. To export the code in GFinder.java for modifications, first the JAR file would need to be decompiled in Eclipse following the directions found here: http://www.avajava.com/tutorials/lessons/how-do-i-decompile-java-classes-in-eclipse.html?page=2. Then a new Java package would have to be created in Eclipse, essentially creating a new area in which to create modifications to the code. To create a Java package, launch Eclipse, click on the drop down arrow of the new icon and select Java project (Figure 8). The user would then name the project by any name and keep the default settings. After this step is complete, the provided GFinder.java file could be uploaded for modification. To import this file, a user should right-click the package name and click on import. After selecting the import function a window will populate with multiple folders. The user should choose the general folder, then double-click archive file. After the pathway of the GFinder.java file is selected, click the Finish button. After this file is uploaded, under the src file within your created java package, double-click the GFinder file and to the right the Java code will populate. For this code to run the BioJava.jar files provided, all .JAR files will have to be imported (Figure 9). To import the provided .JAR files, first right-click the created package, and select “add external archives” from the build path tab. Click on the browse button and select the .JAR files that were provided, then click open. The imported .jar files will then be located under the “Referenced Libraries” folder within the created package.

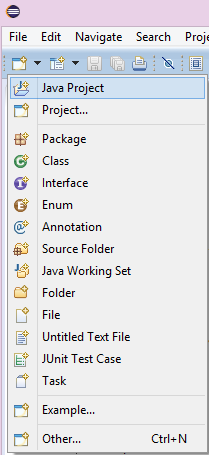
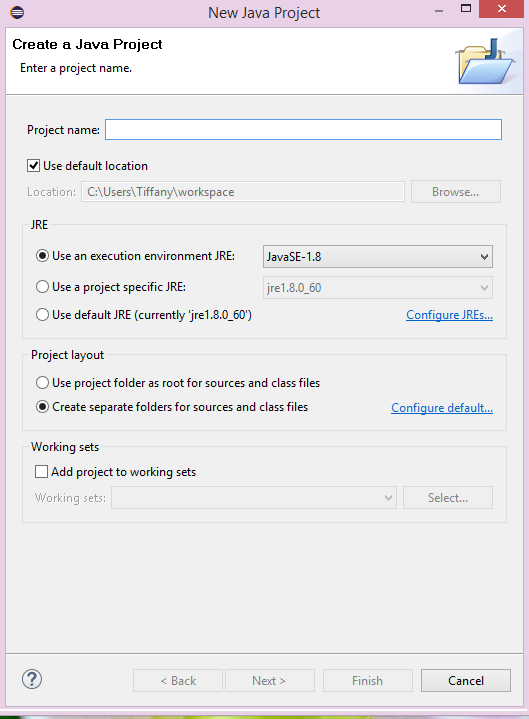
 

Figure 8: Creating a Java project in Eclipse.

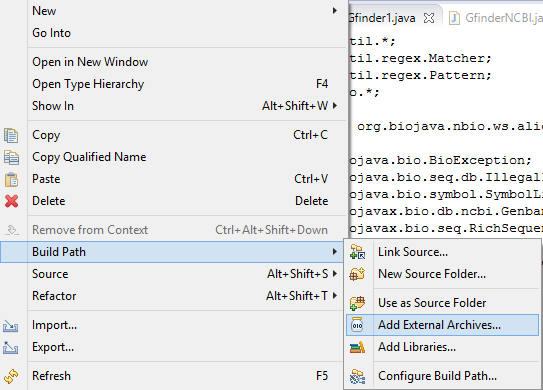


Figure 9: Adding provided .jar files.

**GFinder: Installation and Use**

For better investigation of the radioresistant genes in Halobacterium sp. NRC-1, the G-Quadruplex Identification Working Group designed a platform to isolate candidate G-quadruplex sequences in the three components of its genome. Key papers regarding G-quadruplex sequences, radiation resistant bacteria, and correlative algorithms served as supporting resources for a strategic blueprint. The GFinder tool is essentially focused on a collection of functions within Java, an object-oriented language that does not involve complicated installation of unique libraries. This project involved screening a genomic footprint of approximately 2.6 million nucleotide bases on a platform providing a complete high-throughput screen using a flexible G-quadruplex pattern.

In order for GFinder to run in the user’s computer, there are some basic pre-installation steps that need to be performed. A recent Java SE Development Kit (e.g. JDK 8) must be installed. In addition, a folder for GFinder should be created (e.g. GFinder\_Folder; see the example in Figure 10 below). A high-speed internet connection with consistent bandwidth must be ensured as the inner workings of the GFinder tool are internet intensive. Also, the user is required to have an access to the computer’s command-line interface and be familiar with it (see the example in Figure 10 below).

Upon completion of the Java class files, which contain the steps of the GFinder tool, methods from each file were assigned accordingly into the end program. Next, the end program was compiled into a “.jar” file. The JAR (Java Archive), an executable, compressed file, is a format that serves as a single user-interface while tying together a collection of associated class files. Once it is downloaded, place the GFinder.jar file into the designated folder (e.g. GFinder\_Folder). Doing this helps to reduce the chance of class path and invalid file errors when the GFinder tool is running. After GFinder is correctly installed, the user can operate the command-line interface as follows:

1. Setting the terminal environment to the directory containing the executable .jar, using the following command:

UNIX 🡪 cd /Users/username/Documents/GFinder\_Folder

Windows 🡪 cd C:\Users\*username*\Documents\GFinder\_Folder

*Note: the exact folder path can be found by right-clicking into the properties of the folder*

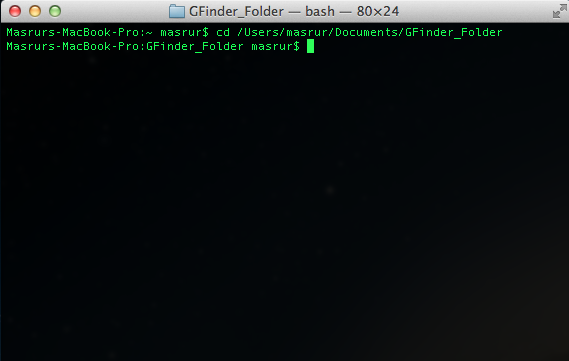


Figure 10: Changing directory for GFinder via Command-line Terminal

2. Executing the GFinder, using the following command:

UNIX 🡪 java -jar GFinder.jar

Windows 🡪 java -jar GFinder.jar

OR

Enable a double-click execution using the following instructions:

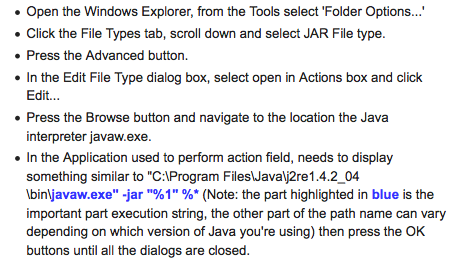


Figure 11: Launching a \*.JAR program by double-clicking it. (Savitt, 2005)

Once GFinder is called, the following welcome screen appears:

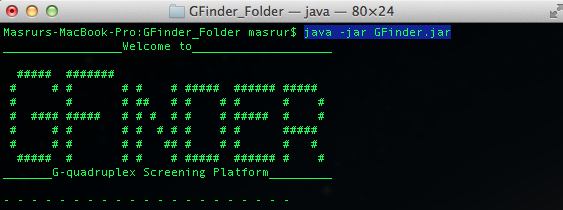
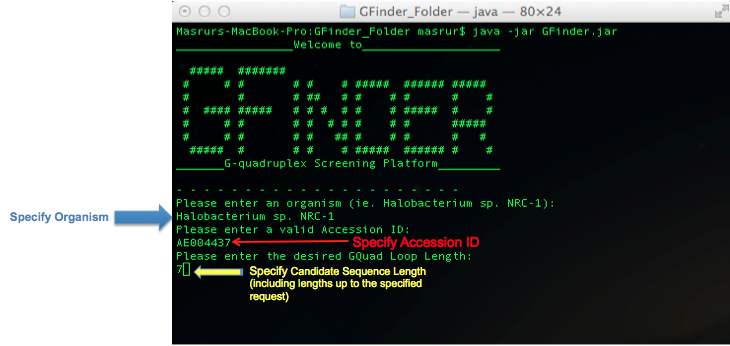


Figure 12: Welcome Screen to GFinder via Command-line Terminal

Certain search parameters are input by the user to determine relevant candidate G-quadruplex sequences: the accession ID of the genome to be searched, the length of a projected G-quadruplex loop, and the organism name to undergo the BLAST search. (see Figure 13 as an example of the display)

 Figure 13: User-inputs for GFinder via Command-line Terminal

Since G-quadruplex structures are loops that are unique from one another as well as differing in sequence and lenth (Harris et. al., 2015), these parameters offer the user a flexible search and GFinder then produces a well-structured report on data structures of potential interest. Once candidate G-quadruplex sequences are identified, the probe hits are order by numeric value while noting the location and GC content of each sequence. Figure 14 (below) demonstrates how these results appear in the interface prior to the production of the final report.

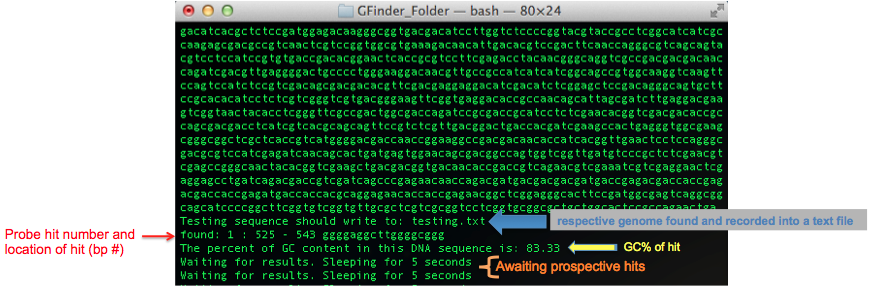


Figure 14: Results in GFinder via Command-line Terminal

Ultimately the completed results are be placed into a CSV file, structured by accession number, sequence description provided by NCBI, as well as the start and stop position of sequence hit. The file is saved to the designated folder (e.g. Gfinder\_Folder ) and a sample can be seen in Figure 15. More complete information on the output can be found in the next section, GFinder: Annotated Output.

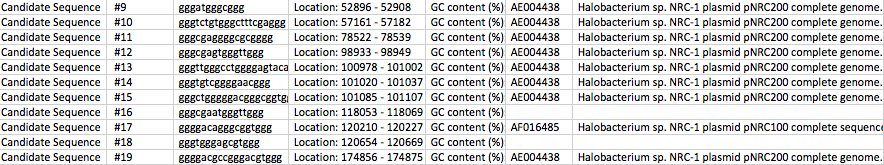


Figure 15: Tab-delimited export file into Gfinder\_Folder

**GFinder: Annotated Output**

The GFinder tool produces a comma-delimited file (CSV output) with a structured format to the results. The CSV output can be opened with Microsoft Excel and also can be saved as an Excel document to facilitate the searching, sorting and exploration to be done with the results. The output file is named based on the genome sequence being processed through the regular expression to identify candidate G-quadruplex sequences as is the page within the file.

*Columns in the CSV output contain information as follows:*

Columns A and B: Together these represent the enumeration of the candidate G-quadruplex sequences found in the genome searched using the regular expression in the GFinder tool. These are listed in the order in which they were found, thus the progression is Candidate Sequence #1, Candidate Sequence #2, etc. (see Figure 16 below for an example)

Column C: This column contains the sequence found in the searched genome by the regular expression that was identified as the candidate G-quadruplex sequence. (see Figure 16 below for an example)

Column D: This column contains the location by nucleotide number of the candidate G-quadruplex sequence. (see Figure 16 below for an example)

Column E: This column contains the calculated G-C content of the candidate G-quadruplex sequence as that metric would tend to be of interest in this circumstance. (see Figure 16 below for an example)

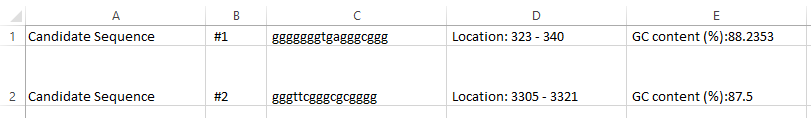


Figure 16: Columns A-E in the GFinder output

Column F (and potentially alternating columns following F): Column F contains the accession ID of the gene that the BLAST search first identified as being a hit. If additional hits are found by the BLAST search, they then appear in columns H, J, L, etc. so as to leave columns available for the description information on any available hits. (see Figure 17 below for an example)

Column G (and potentially alternating columns following G): Column G contains the available description matching the accession ID located in Column F and again corresponds to the first hit found by the BLAST search. If additional hits are found, those descriptions then appear in alternating columns such as I, K, M, etc. and correspond to the accession ID located in the prior column. (see Figure 17 below for an example)

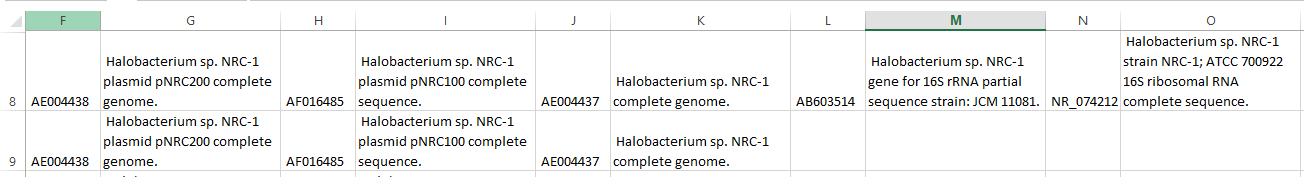


Figure 17: Columns F and beyond in the GFinder output

In working with Halobacterium sp.NRC-1, the G-Quadruplex Identification Working Group utilized the GFinder tool to perform searches on the main genome as well as plasmids P100 and P200. A candidate G-quadruplex sequence found in one of these three locations could produce a BLAST search hit on one or both of the other two if that sequence existed elsewhere; thus sequences may be distinct candidates within one portion of the genomic material but are duplicates when considering G-quadruplex candidates across all three.

**Resources for Independent Study**

The GFinder tool has its conceptual basis in regular expressions, characters within a programming language that define a particular search pattern (Oracle, 2014). The customized G-quadruplex-identifying regular expression, or regex, is then implemented utilizing Java. To better automate this process, BioJava packages are included at the beginning of the process to draw down the genome to be searched and at the end of the process to perform BLAST searching on the candidate sequences as well as to produce the output. Each of these tools is not only available for download without cost, but all are extensively documented online. Though the G-Quadruplex Identification Working Group considered the use of more proprietary tools that were available with a cost for licensing, the goal was to create a tool where flexibility and long term use were prioritized.

In order to take advantage of the flexibility within GFinder, it may be necessary for others utilizing the tool to understand the components and processes that went into its construction. GFinder could also be considering a starting point for other tools that could be constructed for similar or parallel purposes, depending on the direction of research in the future. To help facilitate modifications or the creation of entirely new tools, resources to do independent learning about regex, Java and BioJava are included here.

**Regular Expression Resources**

* Though Wikipedia has its limitations when pursuing rigorous academic research, its strength comes in sourcing technical information due to the overlap between the community interested in maintaining it as a resource and this type of content. Thus, the main Wikipedia page on regex provides an excellent introduction to the purpose, structure and syntax.  
  https://en.wikipedia.org/wiki/Regular\_expression
* Many general resources exist to learn how to implement regular expressions. The focus here is not on the speed at which regex might be mastered, but rather trying to provide multiple venues through which to learn them.
* http://regexone.com/ (The Interactive Tutorial link in the top right gives a full listing of available lessons.)
* http://tutorialzine.com/2014/12/learn-regular-expressions-in-20-minutes/ (This site provides a solid general lesson with straightforward code examples to follow.)
* http://www.rexegg.com/ (This site is particularly effective when researching a particular topic or task.)
* Since GFinder implements a regex specifically in Java, here is the official documentation directly from Oracle about using regex in that context.

https://docs.oracle.com/javase/tutorial/essential/regex/

**Java Learning Resources**

* Oracle provides extensive documentation on the use and implementation of Java. This site serves as the starting point to follow a sequence of lessons to build skills or to research a particular task.

https://docs.oracle.com/javase/tutorial/

* Much like Wikipedia, the use of resources on YouTube needs to be done via careful selection, but a particular standout for learning Java would be the material on the channel, TheNewBoston. Its videos begin with the installation of Java and walk the user through skills that build one on top of another. Each lesson is a few to several minutes so learning can be done in a modular fashion or in larger segments as time allows.

https://www.youtube.com/playlist?list=PLFE2CE09D83EE3E28

* Step-by-step tutorials without videos are also available in order to learn Java. The selections vary in style and quality so a particular user may prefer a different approach, but included here are two straightforward sites
* http://beginnersbook.com/java-tutorial-for-beginners-with-examples/ (This site nicely implements the idea of showing code directly with its results to facilitate learning by example.)
* http://www.tutorialspoint.com/java/ (A nice advantage of this site is the ability to “run” Java code within the browser to see what its implementation looks like and to see results produced interactively.)

**Java Installation Resources**

* Running a Java program requires the code to be compiled, to have the directions converted into ones that the computer can use. It is helpful to have some background on what a compiler is and does when looking at Java installation.

http://whatis.techtarget.com/definition/compiler

* Eclipse is a solid, well-established platform from which to work in Java. Though not a requirement when building in Java, the G-Quadruplex Identification Working Group recommends it for its ease of use once installed.

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/javaeclipseinstallation.html

* Since proper installation of Eclipse is essential, here is a solid resource to use as a guide through that process.

https://www.cs.umd.edu/class/fall2004/cmsc131/EclipseTutorial/install.html

* Java can be downloaded and installed from this site:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

* Eclipse IDE can be downloaded and installed from this site:

https://eclipse.org/downloads/

* Since GFinder also uses packages from BioJava, the Java installation also needs to include the ability to connect with BioJava. Instructions on adding that ability to work with BioJava packages (adding BioJava.JAR) can be found here:

https://www.cs.duke.edu/courses/fall07/cps004g/assign/final/shotgun/addlibrary.html

**BioJava Package Resources**

* All of the official material about BioJava is housed in a single online repository and can be searched for information on the use and implementation of any of its packages.  
  http://biojava.org/wiki/Main\_Page
* The GFinder tools uses two BioJava packages, one to bring in the genome to be searched for candidate G-quadruplex sequences and the other to perform the BLAST search on those sequences to see if they exist in currently identified genes
* NCBIFetch retrieves a genome to be searched for candidate G-quadruplex sequences. In GFinder that retrieval is based on accession ID.

http://biojava.org/wiki/BioJava:CookBook:ExternalSources:NCBIFetch

* NCBIQBlastService performs the BLAST search for each candidate sequence based on the organism genome also specified in the GFinder interface.

http://biojava.org/wiki/BioJava:CookBook3:NCBIQBlastService

**Bioinformatics in the Applied Biological Sciences Degree Program**

SDSM&T’s applied biological science degree is still a young program in the life of the school, but the department clearly understands the importance of bioinformatics and has demonstrated this by already including required coursework on the content. In ongoing discussions with Dr. DeVeaux, the G-Quadruplex Identification Working Group came to understand how the program is looking to expand this knowledge base for their students and potentially integrate it more fully into the curriculum. As part of the within-class discussions which occurred throughout the semester the G-Quadruplex Identification Working Group also posed this question to their fellow classmates. Here is a summary of ideas for the program to consider based on the synthesis from these various sources.

* SDSM&T already has a robust internship and co-operative learning program with a wide variety of companies. Many students enter these programs looking for “real world” working experience and come equipped with the content knowledge for such tasks, but lacking in some of the technical skills that companies are often seeking. While these opportunities are in part a time to build those technical skills, students would have a more robust experience if they had the basics of those skills before starting that work experience.

Suggestion: When connecting with companies to establish these learning opportunities, discuss with them what top two technical skills are that they value in the research positions being held by students. Create a combined lists of skills prioritized by these organizations which are likely to include certain programming languages or techniques of data analysis. Allow students to incorporate an independent research project on one of those technical skills as part of a course that they take in the semester or year prior to that working experience.

* For students pursuing bench or analytical research, understanding the tools that are at their disposal to interpret results from experiments performed is critical. This capability could be expanded within existing laboratory courses to include an increased amount of direct work with these tools.

Suggestion: Make working with the full suite of tools through NCBI a component of existing genetics or cell biology course work. Have students analyze data produced by projects at SDSM&T that utilize ELISA, radioimmunoassays, PCR, arrays of any kind, any of the types of chromatography or other protocols that produce large amounts of image or sequence data. Exercises could be designed around data that is masked but on which publications have been made to compare results that the students derive to those of the actual researchers. Once such exercises were complete, students could then have interaction with those who did the research to discuss the process of reaching those conclusions.

* Bioinformaticians often prioritize the use of the Linux OS, but the ability to use Linux is one that students often pick up independently or on the job rather than in school. Producing students with a working knowledge of Linux would assist them in a broad range of bioinformatics pursuits.

Suggestion: Collaborate with the Computer Science department to have a professor or teaching assistant oversee the execution of an existing project requirement such that it is done in Linux specifically. This could be a project that already exists in the bioinformatics course or another class so additional content does not need to be created and if a current faculty member working in the applied biological science degree was already familiar enough with Linux to fulfill this role, then the involvement of another department might also not be necessary.

* The importance of having a working familiarity with at least some of the multiple programming platforms often used in bioinformatics was something that arose repeatedly during discussions with the entire class. Many of these are open source or are freely available and often come with a wealth of online resources for the motivated, independent student. Areas of particular interest included being able to program in Java or Python, to work with Perl and/or BioPerl, and to understand the basics of SAS and/or R. In particular, R has open-source development and free SAS licenses are now available to active students.

Suggestion: In courses where any type of data analysis or calculating are being done as part of learning or project work, incorporate components that could utilize one or more of these tools. This might start as an extra credit offering while faculty also gains familiarity with the tools, but over time, it can be incorporated as a required part of the coursework. Utilizing a teaching assistant from the Computer Science department to aid the current faculty in their own understanding could be a good use of a cross-department collaboration.

* In keeping with a theme of collaboration, the offerings at SDSM&T could be strengthened by networking with other universities to allow students to take selected offerings, possibly as part of their upper level electives.

Suggestion: Within the state, South Dakota State University has classes where the content could help to round out the bioinformatics components of student. A program to allow online participation in one or two of their courses could facilitate the overall goal. In a more national sense, universities with strong offerings include the University of North Carolina – Chapel Hill, Boston University, George Washington University, New York University, Johns Hopkins University and many others. Collaborations with any of them could provide additional access to students.

**References**

Ajoge, H. O. (2015, July 1). *Pacakage ‘gquad’*. Retrieved from https://cran.r-

project.org/web/packages/gquad/gquad.pdf

Balasubramanian, S., Hurley, L. H., & Neidle, S. (2011). Targeting G-quadruplexes in gene

promoters: a novel anticancer strategy? *Nature Reviews: Drug Discovery*, *10*(4), 261–275. http://doi.org/10.1038/nrd3428

Beaume, N., Pathak, R., Yadav, V. K., Kota, S., Misra, H. S., & … Chowdhury, S. (2012).

Genome-wide study predicts promoter-G4 DNA motifs regulate selective functions in bacteria: radioresistance of *D. radiodurans* involves G4 DNA-mediated regulation. *Nucleic Acids Research*, *41*(1), 76-89. doi: 10.1093/nar/gkm711

Bochman, M. L., Paeschke, K., & Zakian, V. A. (2012). DNA secondary structures: stability and

function of G-quadruplex structures. *Nature Reviews: Genetics*, *1*3(11), 770-780. doi:10.1038/nrg3296

Campeaneau C., & Santean, N. (2009). On the intersection of regex languages with regular

expression. *Theoretical Computer Science*, *410*, 2336-2344. http://dx.doi.org/10.1016/j.tcs.2009.02.022

Capes, M. D., Coker, J. A., Gessler, R., Grinblat-Huse, V., DasSarma, S. L., Jacob, C. G., & ...

DasSarma, S. (2011). The information transfer system of halophilic archaea. *Plasmid*, *65*(2), 77-101. doi:10.1016/j.plasmid.2010.11.005

Chen, Y., & Yang, D. (2012). Sequence, stability, and structure of G-quadruplexes and their

interactions with drugs. *Current Protocols in Nucleic Acid Chemistry*, *17*(5), 1-17. doi: 10.1002/0471142700.ncl1705s50

DeVeaux, L. C., Muller, J. A., Smith, J., Petrisko, J., Wells, D. P., & DasSarma, S. (2007).

Extremely radiation-resistant mutants of a Halophilic archaeon with increased single-stranded DNA-binding protein (RPA) gene expression. *Radiation Research*, *168*(4), 507-514.

Di Antonio, M., Rodriguez, R., & Balasubramanian, S. (2012). Experimental approaches to

identify cellular G-quadruplex structures and functions. *Methods*, *57*(1), 84-92. doi:10.1016/j.ymeth.2012.01.008

Guedin, A., Gros, J., Alberti, P., & Mergny, J. (2010). How long is too long? Effects of loop size

on G-quadruplex stability. *Nucleic Acids Research*, *38*(21), 7858-7868.

Harris, L. M., & Merrick, C. J. (2015). G-Quadruplexes in pathogens: A common route to

virulence control?. *PLoS Pathogens*, *11*(2), e1004562. doi: 10.1371/journal.ppat.1004562

Leo, A., Walker, A. M., Lebo, M. S., Hendrickson, B., Scholl, T., & Akmaev, V. R. (2012).

Technical advance: a GC-Wave correction algorithm that improves the analytical

performance of aCGH. *The Journal Of Molecular Diagnostics*, *14*(6), 550-559. doi:10.1016/j.jmoldx.2012.06.002

Medeiros, S., Mascarenhas, F., & Ierusalimschy, R. (2014). From regexes to parsing expression

grammars. *Science of Computer Programming*, *93*, 3-18. doi: 10.1016/j.scico.2012.11.006

Métifiot, M., Amrane, S., Litvak, S., & Andreola, M. (2014). G-quadruplexes in viruses:

function and potential therapeutic applications. *Nucleic Acids Research*, *42*(20), 12352-12366. doi:10.1093/nar/gku999

Ng, W. V., Kennedy, S. P., Mahairas, G. G., Berquist, B., Pan, M., Shukla, H. D., & ...

DasSarma, S. (2000). Genome sequence of Halobacterium species NRC-1. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(22). 12176-12181.

Oracle (2014). *Oracle: Class pattern*. Retrieved from

http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

Patel, D. J., Phan, A. T., Kuryavyi, V. (2007). Human telomere, oncogenic promoter and 5’-UTR

G-quadruplexes: diverse higher order DNA and RNA targets for cancer therapeutics. *Nucleic Acids Research*, *35*(22), 7429-7455. doi:10.1093/nar/gkm711

Phan, A. T. (2010). Human telomeric G-quadruplex: structures of DNA and RNA sequences.

*The FEBS Journal*, *277*(5), 1107-1117. doi:10.1111/j.1742-4658.2009.07464.x

Prlic, A., Yates, A., Bliven, S.E., Rose, P.W., Jacobsen, J., Troshin, P.V., …Willis, S. (2012).

BioJava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, *28*(20), 2693–2695. doi:10.1093/bioinformatics/bts494

Qin, Y., & Hurley, L. H. (2008). Structures, folding patterns, and functions of intramolecular

DNA G-quadruplexes found in eukaryotic promoter regions. *Biochimie*, *90*(8), 1149-1171. doi:10.1016/j.biochi.2008.02.020

Savitt, J. (2005, October 25). *Setting .JAR file association*. Retrieved from

http://windowstipoftheday.blogspot.com/2005/10/setting-jar-file-association.html

Schmid, M.L. (2013). Inside the class of regex languages. *International Journal of Foundations*

*of Computer Science*, *24*(7), 1117-1134. doi: 10.1142/S0129054113400340

Sen, D., & Gilbert, W. (1988). Formation of parallel four-stranded complexes by guanine-rich

motifs in DNA and its implications for meiosis. Nature, 334(6180), 364-366.

Smith, S. (2010). Evolutionary expansion of structurally complex DNA sequences. *Cancer*

*Genomics & Proteomics*, *7*, 207-216.

*The BioJava Project*. (2013, January 24). Retrieved from http://biojava.org/wiki/Main\_Page

Ujaoney, A. K., Potnis, A. A., Kane, P., Mukhopadhyaya, R., & Apte, S. K. (2010). Radiation

desiccation response motif-like sequences are involved in transcriptional activation of the Deinococcal ssb gene by ionizing radiation but not by desiccation. *Journal Of Bacteriology*, *192*(21), 5637-5644. doi:10.1128/JB.00752-10

**Appendix A: Listing of Output Files From GFinder Search**

1. Halobacterium\_NRC-1\_MainGenome\_OutputFile
2. Halobacterium\_NRC-1\_pNRC100Plasmid\_OutputFile
3. Halobacterium\_NRC-1\_pNRC200Plasmid\_OutputFile

**Appendix B: Listing of JAR Files**

1. GFinder
2. biojava3-alignment-3.0.3
3. biojava3-alignment-3.0.3-javadoc
4. biojava3-alignment-3.0.3-sources
5. biojava3-core-3.0.3
6. biojava3-core-3.0.3-javadoc
7. biojava3-core-3.0.3-sources
8. biojava3-ws-3.0.3
9. biojava3-ws-3.0.3-javadoc
10. biojava3-ws-3.0.3-sources
11. biojava-core-4.1.0
12. biojava-ws-4.1.0
13. bytecode