De Montfort University
MSc Bioinformatics

Final Summer Project

Where do proteins come from? Evolve or be lost?: Analysis of an undescribed gene encoding a hybrid proline-rich protein in sugar beet (*Beta vulgaris*)

Image: Adapted from © http://www.museums.org.za

by Stuart David James McHattie

Supervisors: Dr. Nigel Scott and Dr. Yingjie Yang

2003 / 04

Abstract

A novel gene has been discovered in the root region of *Beta vulgaris*; a sugar beet originating from Germany which has spread throughout Europe and across parts of the United States. Harvested for its sugar production qualities, it would be desirable to understand the meaning of the genes in the *Beta vulgaris* genome to improve genetic engineering and selective breeding and therefore produce stronger strains in the future.

The gene has undergone a barrage of analysis techniques in order to identify information about the position and size of the protein, its properties and to understand more about its construction, function and location. It appears to be made up of three main domains including a hydrophobic leader region, a repetitive proline rich region with low conservation and a hydrophobic cysteine rich region with very high conservation.

The hydrophobicity of the cysteine rich region pulses, which appears to indicate a trans-membrane domain. Motif analysis brings about results indicating that the leader and cysteine rich region may be involved in G-protein coupled receptor like communications, and an abundance of residues relating to signal transduction, particularly in the cysteine rich region, may help to support this hypothesis. The proline rich region does not express any motifs describing function but may be adapted to suit the particular species and therefore not be found in other species.

Analysis of the two main domains (proline rich and cysteine rich) was performed using techniques involving repeat pattern analysis on the former by both predesigned software and software of my own design. Multiple alignments were also used to observe the homology of the cysteine rich region which offers results that suggest the protein is involved in wound defence or tolerance to stressful conditions. The cysteine rich region may span across the cytoplasmic membrane and offer a form of communication across this membrane which in turn can cause the proline rich region to alter the structure of the cell wall and therefore strengthen or weaken it as required.

Very little information can be learnt about the origins of the gene and protein from the proline rich region due to the lack of conservation involved, but information of this nature was extractable from the cysteine rich region, due to its highly conserved nature, which leads to the conclusions mentioned above. Proteins which are most similar in this region, showing to be closer in the evolutionary tree, almost all have an attached proline rich region, some of which were proven to contain similar repeat patterns to those found in the proline rich region of the gene being analysed in this project. This is evidence to show that the evolution of the cysteine rich region is ongoing and it appears to be continually refining itself through generations of mutation. However, since the cysteine rich region only accounts for 14% of the sequence, this evidence cannot be assumed to be true for the whole protein and the proline rich region as a result may not be as favourable to evolution as it first appears. It is suggested that the proline rich region may be using the success of the cysteine rich region which it is bound to in order to continue through to the next generations, despite the fact that it is inefficient and should therefore be eradicated under normal conditions. The proline rich region can never become more efficient under these circumstances because the success of the cysteine rich region will always carry the proline rich region through and therefore natural selection is bypassed.

Acknowledgements

I would like to thank Dr. Nigel Scott and Dr. Yingjie Yang of De Montfort University for their support and expert advice throughout the duration of this project. I would also like to take this opportunity to thank my friends and family for their support and interest in my work; encouraging me to focus my work and helping to stimulate new ideas to improve the outcomes throughout. And finally, I would like to thank my girlfriend who has consistently endured a lack of attention and has still stood strong to encourage my best.

Table of Contents

Introduction

1.1 The Story of *Beta vulgaris*

Eukaryota Viridiplantae Streptophyta Embryophyta Tracheophyta Magnoliophyta Magnoliopsida Caryophyllidae Caryophyllales Chenopodiaceae Beta vulgaris

Beta vulgaris is a species of sugar beet originating from Germany in central Europe. Since its discovery and use to make refined sugars for consumption and to supply animal feeds from pressed leaves, the distribution has spread through most of Europe, the far East and Southern areas of the United States.[i] The exact distribution can be seen in figure 1. The needs of the plant are to have a warm climate and fertile soil, but however, some of the regions in which it is grown do not meet these requirements most of the time and as such, the plant is able to withstand certain stresses places upon it by its climate.

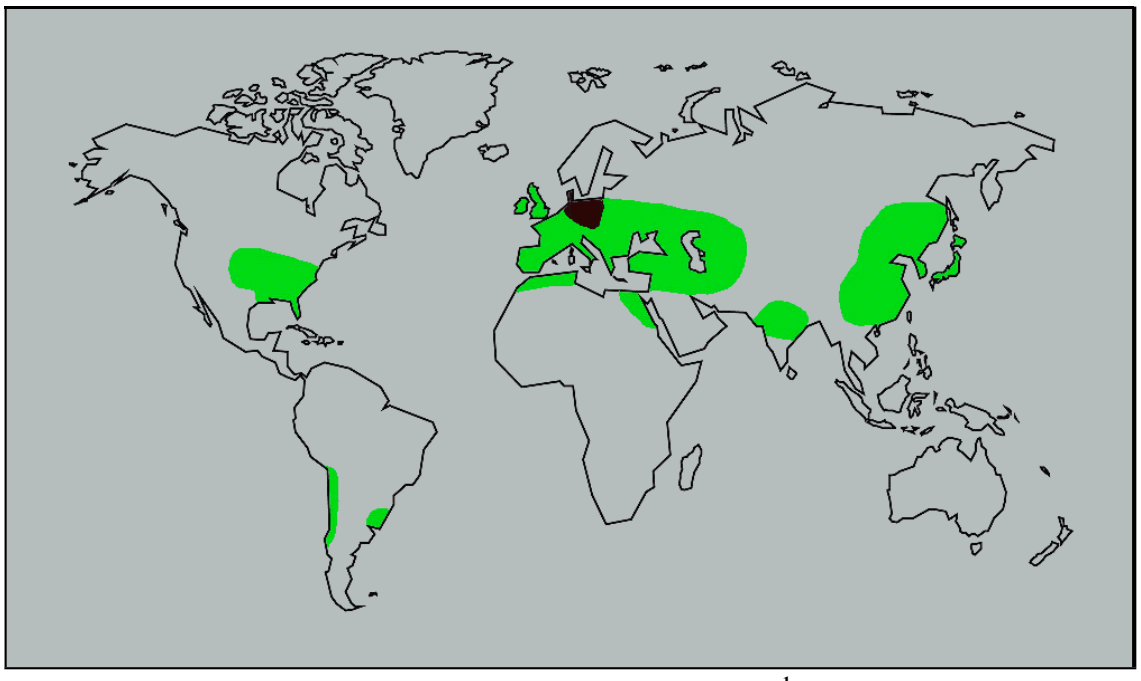


Figure 1 - Distribution of *Beta vulgaris* [1]

Over 270 million tonnes of *Beta vulgaris* was grown world wide in 1997 according to figures from the Food and Agriculture Organisation. The breeding of Beta vulgaris has been underway since 1786, and has led to a number of different varieties, including "White

Slesian Beet" which is the ancestor of today's varieties. By selective breeding, the sugar content can be raised from 8% as high as 20%, but further aims are to also increase resistance to fungal and viral diseases and to reduce the shooting tendancies of the plant.

1.2 The Target of Interest

In *Beta vulgaris*, a novel gene is expressed which appears to show signs of being a Hybrid Proline-Rich Protein. This gene appears to be made up of three domains: a leader sequence, a proline rich region and a cysteine rich sequence. The function of this gene is unknown and very little is known about its expression patterns. Limited analysis of the protein this gene encodes for has been made, and knowledge about it is limited to knowing that the proline rich region is highly unconserved whilst the cysteine rich region is highly conserved. It is thought that the protein may be cell wall binding with the hydrophobic part in the cytoplasmic membrane and the proline rich region buried in the cell wall, but there has been no true proof for this.

1.3 What are Hybrid Proline-Rich Proteins?

Hybrid Proline-Rich Proteins, or HyPRP for short, are a large and widely diverse family of proteins which consist of a proline rich region in which a higher than normal proportion of amino acids in the protein are proline. The reason for the term "hybrid" in the title is that there is a second domain which is also distinctive and is found at the C terminus end of the proline rich domain. In *Beta vulgaris*, this second domain is cysteine rich and is thought to produce a very distinctive shape in order to pass through the cytoplasmic membrane. This domain is very highly conserved and can be seen to vary only very slightly even between species.

Three main forms of structural cell wall proteins have been described in dicotyledonous plants. These include extensins which are hydroxyproline rich glycoproteins (HRGPs) which have roles in strengthening the cell wall when they are expressed or activated. The second type are glycine rich proteins (GRPs) which contain a higher than normal glycine content. Lastly are the hydroxyproline rich proteins (PRPs) which contains a higher than normal proline content.[ii]

A lot of work has been done on HyPRPs in the past, particularly in finding out the attributes and functions of such proteins. Soybean has been of particular interest, expressing several proteins that are shown to have proline rich regions. The three proteins of interest have been labelled SbPRP1, SbPRP2 and SbPRP3[iii]. They were shown to have sequence homology and yet remarkably different patterns of expression. SbPRP1 is found in the root and mature hypocotyls in the early stages of seed coat development; SbPRP2 is expressed in mature leaves, stem and seed coat as well as the apical regions of hypocotyls and young cultured suspension cells; SbPRP3 is found in only aerial parts of the plant in leaves, stem and pods. Overall, the PRPs were found to be present in all organs in one form or another. It was concluded that PRP gene families may play a significant and important role in the plant development stages due to the pattern of expression.

There is a certain difficulty in the observation of HyPRPs in that due to the nature of them being embedded in to the cell wall, they are difficult to separate from within it, and therefore it becomes impossible to look directly at the protein itself. Instead, information gathered about the protein is based on the presence of mRNA which in turn generates cDNA.[iv] The main flaw in this process appears to be that whilst a protein is not being expressed there will be little or no mRNA available in the specimen from which the cDNA can be created. This leads to the reasons why many of the proteins which should have been included in the analysis of repeat components within proline rich proteins of the study were absent. Whilst a variety of environmental and developmental stimulations were applied, it may be that some of the genes were not stimulated to become expressed.

Further problems in the characterisation of PRPs also exist. Allan M. Showalter attempts to summarise some of these in his paper entitled "Structure and Function of Plant Cell Wall Proteins"[v]:

> *'For what function does one look? Is one looking at the right place and time? Is one looking under the correct conditions to see an altered phenotype? Will other members of the gene family or other wall proteins compensate for the loss of one wall protein?'*

Maize has also had HyPRPs observed in its genome[vi] which have been shown to be linked to embryo growth. The expression of the HyPRPs is such that they cannot be found in the

adult plant apart from in the ovaries and it is suggested that it could be a modifier for the structure of the cell wall to protect the cells during later embryogenic development.

1.4 The Significance of Proline

Proline is considered fairly unique in its properties as an amino acid. This is mainly due to the fact that proline is technically an imino acid due to the side chain of the molecule being doubled back in a five member ring via the second nitrogen in its structure.[vii] This causes its isolated form to have an NH_2^+ group instead of the usual NH_3^+. Due to this structural difference, proline is unable to occupy the same structural shapes as mostly any of the other amino acid and is often seen to create bends within the peptide chain structure. It is also the main cause of 'kinks' within structures such as alpha helices since proline exhibits specific shapes when placed in succession; not the standard shape of an alpha helix. Despite the aliphatic and hydrophobic properties of proline, its structural tendencies cause it to usually be on the surface of a protein.

The main functions that proline is involved in are molecular recognition, and in the case of intracellular signalling, binding to domains such as SH3 (Src-homology 3) via structures conforming to the PXXP motif[viii]. Proline is also known to bind to aromatic surfaces of other amino acids, but the process by which this is done is not yet fully understood. However, other than these two main functions, it is highly unreactive due to its complete and enclosed ring structure and for these reasons is rarely involved in active sites or binding.

The binding of motifs involving high quantities of proline to domains such as SH3 and WW means that these domains have become very popular in proline recognition[ix]. This knowledge offers an alternative method of PRP detection which may yield results which offer a more accurate representation of their presence rather than their transcription elements.

The polyproline type II (PPII) helix is the structure formed when multiple proline residues are found consecutively in the peptide chain. This structure is unique to proline, hence why proline does not substitute with other amino acids very favourably. The best description that can be given to this shape is that of a triangular prism similar to the

packaging of a Toblerone. This can be seen in the illustrations of figure 2. Note the interesting aspect of this structure in that the orientation of the helix could change the function due to the asymmetry. It has been suggested that this property may allow the helix to bind in one direction to cause one result and yet bind in the other direction to cause the opposite[x].

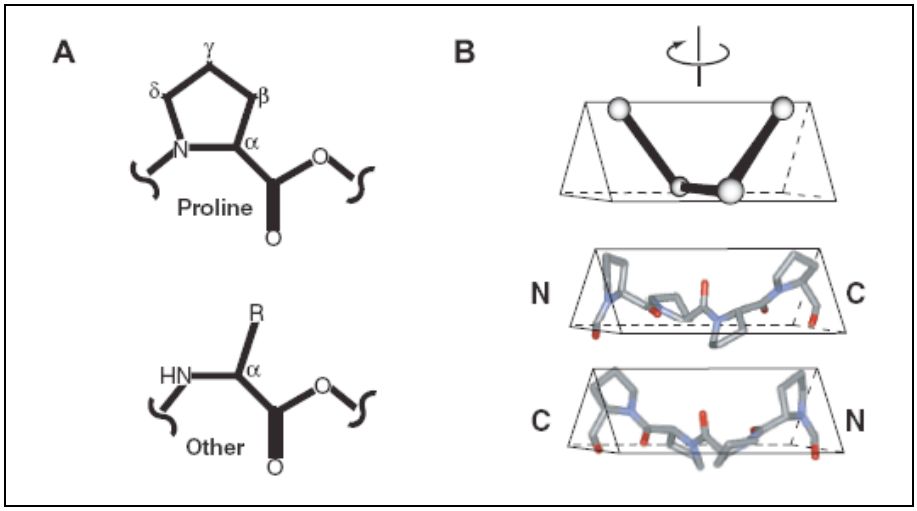


Figure 2 - Polyproline type II helix structure[10]

1.5 The Significance of Cysteine

Substitutions in cysteine are very limited, and only other very small amino acids can be substituted with it, although even this is a neutral event since cysteine has no preferential substitutions.[xi] In extracellular proteins, the cysteines within the structure serve almost entirely to shape the protein, creating disulphide bonds between cysteines. Within cells, however, the reducing environment makes disulphide bonds very hard to create, and for this reason, the cysteines within the proteins rarely form them. This is also true of membranes, in which the cysteine will tend to form bonds in the regions outside of the membrane in order to allow for the turns the protein must make to pass back through. Intracellularly, cysteine still plays a structural role by having a tendency to bind to metals such as zinc. Other amino acids, including histidine, also have this property, and often lead towards the generation of a zinc finger in which all the metal binding amino acids cluster around the metal to form the shape of the protein. Outside of these forms of binding, cysteine does not show any particular property within the protein and can often be substituted by small amino acids with similar properties.

Outside of creating structural changes to a protein, cysteine is often involved in enzymatic active sites. The metal binding property of this, and other, amino acids may be part of an active site on a metal protease. Alternatively, it can be part of the nucleophile of a protein, being the main body of the function of an enzyme in the middle of the protein.

1.6 Objectives

The function of the HyPRP found in *Beta vulgaris* is unknown and the evolutionary path of the protein is also a mystery. For these reasons, the following objectives will base the methodology of the project and supply the aims which are intended to be answered in the discussion:

1. Discover the basic biology of the gene, including where the encoding open reading frame is and basic assessment of codon usage, protein domains and promoter region.
2. Look at the proline rich region to find out the major and minor repeats within the sequence. Take this information and try to find other proteins with similar repeats within them.
3. Look at the cysteine rich region to find out how much homology it has to other proteins and find out whether it is always found with a proline rich region or only on its own. Perhaps try to construct some form of phylogenetic tree, showing the genetic origins of this protein.
4. Use the other proteins found in objectives 2 and 3 to infer function, structure and maybe location. Supplying information that may be useful to be used *in vitro* to test this protein further.
5. Discuss the evolutionary origins and relationships to other proteins in the family and infer whether the protein has a successful future in the evolutionary path or if it will be lost forever.

1.7 The Data

The genetic code for the RS1 gene was supplied in the form of the nucleotides from the *Beta vulgaris* genome, including the code before and after the coding region for the protein. This code, comprised of 7317 nucleotides, is the sole input into the project for analysis and is displayed in appendix B.

Methods

2.1 Analysing the Raw Data for Information

The first steps in the investigation were to identify as much information as possible about the gene and the protein it encodes for using *in silico* techniques. This is done using a variety of pieces of software available online, mostly embedded in to web pages, but also some downloadable elements. This information forms a basis for the main analysis of the gene and avoids wasting time later by analysing elements that turn out to be irrelevant.

2.1.1 Using GenScan to Find the Open Reading Frame

GenScan[xii] is an online gene analysis tool. Its main job is to locate introns and exons within genomic pieces of DNA using Markov models to predict the likelihood that a particular combination of nucleotides will indicate a splicing point. It also identifies open reading frames in genomic sequences. This leads to the output of the probable open reading frame within the raw data it is supplied with.

By pasting the genomic sequence from Appendix B in to the web server's form, GenScan can be made to identify the splice points in the RS1 gene. The organism choice should be *Arabidopsis* since this organism is closest to *Beta vulgaris* of the three that can be chosen. This is an important step as different kingdoms and even species can have differences in the functioning of their DNA, and splice sites are an item that are particularly sensitive to these differences. The final output after changing these settings shows a table of possible splice sites and the probable remaining gene coding region. This is also accompanied by the predicted peptide chain that this encodes for, both without numbering or annotation.

2.1.2 Labelling the Peptides and Nucloetides

In order to continue the analysis, it is often a requirement to go back to the original open reading frame and the peptide chain sequence to find where the output from further analysis fits in to the whole picture of the gene. To do this, it is very handy to have numbering of nucleotides and to be able to see which nucleotides are encoding for which amino acids. For this reason, the information supplied by GenScan is run through a

program that can format it in to a layout useful for these purposes. One such piece of software that will do this job is the Translation tool on ExPASy[xiii]. The output from this particular piece of software does not place numbers in the sequence, but these can be easily added afterwards if necessary.

2.1.3 Finding the Promoter Sequence

The promoter region of the gene is the part in the genomic DNA that is related to identification when the gene is due to be expressed; the promoter provides a binding point from which the RNA polymerase can begin unravelling the DNA. This usually includes a TATAA box which is a form of signal for the RNA polymerase to know it is approaching the coding region. One such piece of software for predicting the location of the promoter is NNPP version 2.2[xiv]. The output from this can be assessed to see how useful it is, with the help of the scores for each identified site; 1 being that of a perfect match and everything less than 1 being an indication of how likely the match is to be accurate.

2.1.4 Locating the Domains

In order to perform analysis on the domains and differentiate them from one another, they must first be located in the coding region of the gene. This is basically a search for motifs within the gene that show a region is from a particular family of genes. The proline rich region is known to be highly unconserved and so is harder to identify, although it is usually identified by a high concentration of proline which is above the usual and expected level for the particular organism. The software used for identifying and then describing domains within the protein is SMART[xv]. By pasting the peptide sequence into the search query box this software identifies the domains and then shows URL links to pages describing them.

2.1.5 Looking at Codon Usage

In order to understand more about the increased levels of certain amino acids used within the gene, a codon counter can be used. This software looks at the codons used within the gene and summarises their abundance in a table. The chosen software in this project also gives access to information concerning the entire genome of *Beta vulgaris* which gives some kind of reference for the output.

CountCodon[xvi] displays information about how many times a particular codon appears in the sequence and the frequency of it is also expressed in thousandths of the entire

sequence. Using this information, and that of the genome of *Beta vulgaris*, the usage of the codons will give an idea of which amino acids are over expressed and which are under expressed. It will also give information regarding whether there is a preferred codon for each amino acid.

The results from CountCodon were transcribed in to Excel where they could be compared. The frequency of each particular residue was calculated by using the SUM command which could add together the percentage usage of each codon. These results were brought together to make a simple subtraction operation to see how much more expressed each residue is. The results will prove useful in identifying the proline rich region and in finding out if any codons have a preference over the normal usage found in the entire *Beta vulgaris* genome.

2.1.6 Using BLAST to Find Similar Sequences

BLAST[xvii] is a very popular local alignment tool which has access to all the previous categorised and annotated genes and proteins in the main gene databases such as the DNA Databank of Japan and SWISS-Prot. By finding annotated proteins that have similarity to the RS1 protein, it not only shows the nature of the domains of the protein, but also allows inference of other properties about the proteins if homology can be found.

Whilst performing BLAST searches on the RS1 protein, the settings in BLAST must be tweaked to get good results. Firstly, BLASTp is the search type needed; this is used for comparing protein sequences. BLAST supports a number of filters to help improve search results and in this instance, the low complexity filter must be disabled because the proline rich region, being composed mainly of repeats, is considered to be composed of low complexity sequence which may make the results become based entirely on the leader domain and the cysteine rich region. By removing this filter, the quality of the results is reduced since the low complexity region will often produce apparent matches which are not at all related. But this is a risk that must be taken if the proline rich region is to take part in the algorithm.

The results from BLAST will be useful in the next stages which include the comparison of RS1 to other proteins that are similar.

2.1.7 Using Dotplot to Identify Similarities

Dotplot is a method of taking two proteins and placing their peptide chains along the axis of a graph. With one protein on the x axis and one on the y axis, a plot point is made wherever the amino acid lines up with the same amino acid on the other axis. So when there is a P for proline on the x axis, at every point in that vertical plane where P is found on the y axis, a plot point is made. Using the output from this can indicate where homology occurs within the proteins, and can identify insertions and deletions by diagonal lines that are broken by horizontal and vertical gaps. Dotplot can only be done between two proteins at a time and so the choice of plot must be selective to only a few proteins of interest.

BioEdit[xviii] is able to perform Dotplot analyses using short sequences that are input into it. It uses slightly more complicated methods of placing the plots, because it also creates lighter dots in grey when the match is not perfect; for example, S and T are preferential substitutions for each other due to their similar properties and function and would receive a dark grey plot instead of a black or white one.

The results from BLAST in the previous section had a selection of 3 sequences chosen for Dotplot analysis to see where the similarity lies, and this resulted in the images found within the results section.

2.1.8 Performing Multiple Alignments with Bioedit

BioEdit also has many other functions, as well as dotplots. This includes being a full alignment editor. Whilst it is unable to perform multiple alignments itself, it calls upon the functions of ClustalW[xix] and gives the results from this with colours. This aids in the alignment of the sequences by eye to correct for errors in the alignment made by ClustalW. After this has been done, BioEdit can be made to produce a consensus sequence to show where the alignments between the sequences are very strong (a consensus residue is formed) and where the alignments are weak (no consensus is formed). An example of the consensus sequences can be seen in the results.

2.1.9 Looking for Motifs Using MOTIF

Motifs are similarity in structure between family members. These similarities act as a form of signature which indicates that a protein is from a particular family. The complication is that structure cannot be obtained very easily, particularly from novel proteins, and so the inference that similar sequence is equal to similar structure must be used. This inference is not perfectly accurate, but it is the best way for motifs to be identified in a short period of time.

In order to help narrow down the functions of the protein, MOTIF[xx] will scan the sequence for known motifs which may identify which famil(y/ies) this protein belongs to. This can be done at the nucleic and protein level which will yield results to identify which aspects of different families are shown in the gene and protein combination. By selecting more than one database to search in, more results can be obtained, but they are often duplicated results and so results were limited to the PRINTS database for the protein level.

2.1.10 Plotting Hydrophobicity

Hydrophobicity plots are useful for two reasons; firstly, any domains which are predicted to be crossing a membrane layer are likely to have high hydrophobicity, and if they cross the membrane more than once, i.e. they double back on themselves; they will more than likely have areas which lack hydrophobicity on the turning points.

The second reason for this analysis is that when repeats exist in the protein, they will all present exactly the same hydrophobicity pattern. This can be helpful in identifying larger repeats which are not always seen using other repeat analysis. If a repeat goes unnoticed during the analysis but the hydrophobicity plot shows a repetitive shape then the residues can be rechecked by eye to see if a repeat in the sequence exists.

Colorado State University has a site which allows the user to input a sequence and receive a hydrophobicity plot based on the Kyte-Doolittle scale[xxi]. This Java applet has no features that you can adjust and so presents a simple interface which forms the basis of the hydrophobicity plots in this project.

2.2 Analysing the Proline Rich Region for Repeats

Having performed the basic analysis of the sequence for the RS1 gene it becomes apparent that the protein would be impossible to categorise without breaking it down into the two domains which make up the latter half of the protein. For this reason, the repeats found in the proline rich region were analysed separately in order to find other proteins with not only a proline rich region but also strong similarities in the repeat patterns to the RS1 gene.

2.2.1 Using Tandem Repeats Finder

Tandem Repeats Finder[xxii] is a program for finding repeats within a DNA sequence and takes account for insertions, deletions and substitutions. Each repeat found is fully annotated and illustrated in order to make it very obvious where the differences lie between repeats. As well as this, a summary table details the repeats that have been found. This program has been put to use to try to identify if repeat patterns exist within the gene as well as the protein. This software does not, however, allow for comparisons of residues within the protein. It is likely that this is due to complications in residue preferential substitution making the software much harder to develop. However, this does not mean that it is impossible.

2.2.2 Using Radar

Radar[xxiii] is software designed to perform the task of detecting repeats in the protein residue sequence. Its intentions are the same as tandem repeats finder but it also has the added complications of dealing with the protein instead of the gene. The output is not as detailed as that from tandem repeats finder and it appeared to be unable to identify the repeats correctly in the RS1 protein. For this reason it was unsuitable for the task and an alternative was sought.

2.2.3 Using Specially Designed Software

Due to the lack of software available to identify the repeats in the protein sequence, it became necessary to design some software from scratch which would do the required task. The language of choice for this program was Java since it is both fast to develop in and is object orientated to limit the code required to handle large quantities of data. A further added bonus is that Java is available free of charge and many sites across the internet have examples of different pieces of code to help out in times of difficulty.

The full written code is presented in Appendix C in four sections, each containing the classes that make up the full program. The Green text indicates where comments have been included and should be easily readable so as to show the full flow of the program. However, the following is a short description of the full program in plain English.

- *TandemRepeats.java* - This is the class that contains the main function which initiates the program.
- *Sequence.java* - This class acts as a container for the sequences that the program will be analysing. A separate instance of this will be initiated for each sequence.
- *Repeats.java* - This class acts as a container to record details about the repeats within the sequences as they are found. Repeat patterns that are identified will be stored in separate instances of this class.
- *PatternMangler.java* - This class contains the code required to compare sequences and find similarity even when there are insertions, deletions and substitutions. As it finds these similarities it returns the results to the Sequence class which in turn stores the similarities in the Repeats class.

When the program begins, it initiates an instance of the main body of the program which in turn begins by reading 'Sequences.txt' which contains a list of all the sequences that are to be analysed. The format of this file is that the first line begins with '>' followed by the title of the next sequence and the line following contains the actual peptide sequence. Each sequence is loaded into its own instance of Sequence.java. After displaying prompts for the user the software tells each sequence in turn to perform analysis on the repeat structure.

The analysis of each sequence is as follows. By starting at the first position within the code, the first three letters are stored and the next six letters following that are also recorded. Both these pieces of information are transferred to the PatternMangler which tries to make a comparison between them. It will first of all check that the peptides in position one of both sequences are identical and then begin comparing on a residue for residue basis. If the residues do not match, then it will begin to look at residues farther down the chain to find a match which accounts for insertions and deletions. If there is still no match it begins to look at substitutions. Each amino acid has preferential substitutions

and so when the software compares the residues it will use a function designed to replace these substitutions in the hope that one of them will present a match. If all the residues can be accounted for after insertion, deletions and substitutions and less than 25% of the residues had to be substituted to find the match, PatternMangler will return a positive result.

Upon return to the Sequence class, the newly found matches are stored in a new instance of the Repeats class. The next six letters of the code would then be compared to the previously identified repeat pattern in the same way. Once the program cannot find any more repeats it begins to look at increasingly longer potential matches at the same location until either further repeats are found or the maximum length of twenty five residues is reached. At this point the size of the pattern being clipped from the code is reduced back down to three and the point of observation is increased to the next residue. Once again the process begins searching for repeats until eventually the end of the protein is reached at which point control is returned to the original TandemRepeats class.

Once control is returned to this point, the next sequence is instructed to perform the same analysis and this continues until all the sequences are completely analysed. The next step in the process is to regurgitate the results into a text file named 'Repeats.txt'. This job is given to the Sequence class which starts each insert into the file with a numbered layout for the peptide sequence for easy comparison by the viewer. This is followed by a list of all the repeats that have been found in that sequence. The format for this is a title for the repeat followed by a dashed line and then details about where the repeat can be found in the sequence and clipped sections of the sequence illustrating the repetitions that have been found. Once all the sequences have done this, the end of the program has been reached and the user is left with the completed 'Repeats.txt' file.

On several occasions in the program, particularly around the points where file access occurs, there are routines for error handling to catch any mistakes in the files or difficulties in accessing them. Since this is the only point at which the program could go wrong, due to human errors, this is all the error handling that is required to ensure a smooth flow of the software.

The running time of the software is very impressive compared to what could be expected, partly due to the fact that the software is object orientated and as such only uses the quantity of system resources required for the task. As an indication of its speed, fifty sequences were fully analysed in less than two seconds on a 1.4GHz machine with 512Mb of RAM.

2.2.4 Taking the Results to BLAST

As described before, BLAST is a useful tool for finding sequences in the databanks that exhibit matches to given search query sequences. Having obtained information about the repeat patterns found in the RS1 gene, they could now be searched within the databanks using BLAST. This should yield results that show greater similarity and relevance to the RS1 gene rather than searching using the entire proline rich region as was done before. The reasons behind this are that due to the highly unconserved nature of the proline rich region, many results that show no resemblance to the RS1 gene may be extracted when searching using the whole domain. The theory behind searching for repeat patterns is that they act similarly to motifs. Proteins containing similar patterns should have similar function. There is also theory that to become repeated there must have been some form of mutation from an unrepeated form and perhaps those proteins expressing this unrepeated form will be identified using the BLAST search. In order to get meaningful results from BLAST it is imperative that the predefined settings to find short, nearly exact matches are selected.

2.3 Analysing the Cysteine Rich Region for Homology

To complete the analysis of the RS1 gene the second main domain must be analysed. In some respects this is a simpler task than the analysis of the proline rich region, as this domain does not have repeat patterns but instead shows high levels of conservation. This means that by comparing it to proteins that have the same domain, the history of the RS1 gene and its position in a phylogenetic tree can be established since small changes in a highly conserved region can be considered huge jumps in evolution.

In order to perform this type of analysis, it is essential to gather a large number of proteins using BLAST to compare with the RS1 gene's cysteine rich region. By putting the

cysteine rich region from RS1 into a BLASTp search, a list of fifty proteins is generated, which form the basis of the data set for this analysis.

2.3.1 Gathering Alignment Scores with ClustalW

As the main part of this analysis, it is essential to perform pairwise alignments between each and every protein in the data set. Strictly speaking, the only analysis required is that of comparison between RS1 and every other protein, since our main interest is in finding the function of the cysteine rich region in RS1; but with high aspirations the intent is to form some kind of phylogenetic tree which will only be possible if the evolutionary distance between each protein is known.

ClustalW[xxiv] is a multiple alignment application which effectively identifies alignments between all the sequences submitted to it. However, it will also supply measurements indicating how similar each protein is to every other protein submitted. This information is supplied in the form of a text file which details which alignment scores were given for each cross. In this format the information is not very useful and so it must be transcribed into a spreadsheet.

2.3.2 Bringing the Scores Together in a Table

In order to make best use of the information the data needs to be arranged into a design similar to that of a mileage grid found in an atlas. Along the top and left hand edge of the grid is a list of the sequences. Each cell of the grid contains an indication of the alignments score for the two sequences with which it lines up.

By also adding shades of green to higher scores, using a macro to automate the process, it becomes immediately apparent which proteins have the highest similarities in their cysteine rich region. This is useful for an overall view, but to perform the simple part of the analysis, only the first column needs to be considered since this is the alignment scores for the RS1 gene. By sorting these into order it is possible to find the proteins that are most closely matched which can have their annotations and documents about them carefully inspected to see what the significance of the cysteine rich region is.

Data from the entire table can then be used to supplement this information with the attempt to create a phylogenetic tree.

2.3.3 Using the Table to Create a Phylogenetic Tree

ClustalW is able to produce the equivalent of a phylogenetic tree during the process of comparing all the sequences together. The result is called a phylogram which is a cross between a phylogenetic tree and a hierarchy diagram. It is supplied as a Java applet on the EBI website which has settings about the way it should be displayed in a right click drop down menu. This allows the changing of the colour of fonts and layout options regarding the position of text.

The total distance between two individuals in the diagram illustrates the genetic and evolutionary distance between them. These distances are arbitrary but can be compared to one another as they are all to scale. Therefore, if a distance between two individuals is larger, the evolutionary distance is greater than a shorter distance between two individuals, but the actual distance on the diagram does not have any particular relevance. Therefore, individuals that are on the same branch of the tree are very close to one another and individuals that are displayed closer to the left of the diagram are generally closer to other proteins than those that are farther to the right.

3.1 Whole Sequence Analysis

During the first stage of the analysis of the RS1 gene, identification of the main aspects of the gene were performed, including finding an open reading frame, annotating the encoded protein and finding out how this relates to other known proteins. The results that are produced by this analysis are detailed as follows:

3.1.1 Finding the Open Reading Frame

The output from GenScan is very long in length and so to pick out the items which are useful to the project, the results from this program have been reduced in size and are included in Appendix D section 9.1.

3.1.2 Labelled Peptides and Nucleotides

After formatting the results from GenScan, a tidy display of the nucleotides beside the amino acids they encode for was generated and nucleotide numbering was included. The results of this formatting can be seen in Appendix D section 9.2.

3.1.3 The Promoter Sequence

The output from NNPP version 2.2 to identify promoter sequences can be seen below. Note that there is one possibility that has a score of 1 identifying it as the correct promoter sequence and there is an obvious TATAA box highlighted on this line:

Promoter predictions for Beta vulgaris (RS1):

Start	End	Score	Promoter Sequence
346	396	0.95	AAGGGTCTGGAAAAAATGGACCCTGACCCGGACCCTTAGGGTCTGGAGGG
1828	1878	0.94	TAGCGCACCCTATAAATCCCAAGTCAATTAATAGTGACAAAAGCATAAAA
2167	2217	1.00	TGAGTCTATTTTAAAAAGAAGCGGTGATTTGTAGGTTGTTAGACTATAAT
3155	3205	0.95	TCTCAAATTCTATATAAGAAGTCATACTACTCTTCCATTTACTTATTCAT
3516	3566	0.89	ACATTCTCCCTATAAACCACCTTATCACACCCCACCATATACTCCTAAAC
5308	5358	0.98	TGTTAGGCTTTATATATCTTGCCTAGTTGCCTACATGTATGTAGTATGAT
5469	5519	0.99	ACTCCCACTATATATATGATTCTCTTTAGCGCAGCTAGTCCTTACATGTT
5603	5653	0.98	TCAAACTTTATAAATAAACACTGGTCATTTTTTCGCTAAAAATATCTAAT
6434	6484	0.98	ATGATGTGATTAAAAAATGGTGTGACAGATGCAATTAACGAAACAAAAAC

Figure 3 - Output from NNPP version 2.2

3.1.4 Domain Identification

SMART, when offered the sequence information for the RS1 gene, came up with several matches identifying the domains and properties of the protein. These are as follows:

name	begin	end	E-value
signal peptide	1	27	-
low complexity	228	494	-
Pfam:Tryp_alpha_amyl	498	579	4.80e-22

Figure 4 - Detected Properties and Domains within RS1

This leads to the production of a simple image illustrating the position of these properties and domains, shown below:

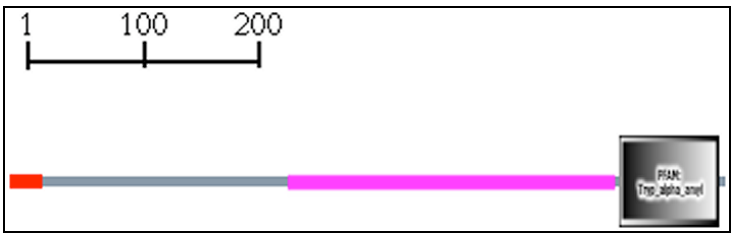


Figure 5 - A Graphical Illustration of the Domains and Properties

Further aspects were detected which are listed in the following table. These were not added to the above image because they either overlap with other detected items, or they are not significant enough to be trusted:

name	begin	end	E-value	reason
low complexity	3	19	-	overlap
transmembrane	7	29	-	overlap
BBOX	22	57	7.72e+03	threshold
low complexity	24	225	-	overlap
AAI	44	579	6.43e+01	threshold
KAZAL	477	528	1.70e+03	threshold
Pfam:Kazal_1	478	528	5.10e+00	threshold
GRAN	487	545	1.36e+03	threshold
HYDRO	487	543	9.58e+02	threshold
low complexity	512	523	-	overlap
CLIP	529	580	4.00e+03	threshold
low complexity	552	559	-	overlap

Figure 6 - Other Detected Domains and Properties not included in the Figure 5

The most significant domain appears to be the cysteine rich region which has been identified with the codename 'Pfam:Tryp_alpha_amyl'. By going to the webpage which describes this family of proteins, the following information is discovered:

Protease inhibitor/seed storage/LTP family

This family is composed of trypsin-alpha amylase inhibitors, seed storage proteins and lipid transfer proteins from plants.

This family forms **structural complexes** with other Pfam families:

Pfam domain	Sequence
Tryp_alpha_amyl	IAAT_ELECO
Alpha-amylase, Alpha-amylase_C	AMY_TENMO

Figure 7 - Domain Description from Pfam

3.1.5 Codon Usage Analysis

Whilst analysing codon usage, several tables were generated in Excel which originate from the data supplied by CountCodon. Tables relating to the codon usage in the RS1 gene and the entire genome of *Beta vulgaris* can be found in Appendix D section 9.3.

3.1.6 Dotplot Graphs for Comparison

After BLAST searching, many proteins were returned showing homology across the entire sequence. The following were selected for dotplot analysis:

- An unnamed protein from *Arabidopsis thaliana*
- A putative proline-rich protein from *Solanum brevidens*
- A protease inhibitor/seed storage/lipid transfer protein (LTP) from *Arabidopsis thaliana*

These were chosen partly due to their presence in the plant kingdom which would make them closer to RS1 than other possibilities, and characteristics from the Pfam description of the cysteine rich region also influenced the choice.

The results from the dotplot analyses against these proteins can be found in Appendix D section 9.4.

3.1.7 Multiple Alignments of Whole Sequences

The multiple alignments of the sequences illustrated in the dotplot give a colourful display showing the amino acids in each sequence and a consensus sequence which shows where all the amino acids are identical. This is shown in Appendix D section 9.5.

3.1.8 Motif Identification

MOTIF gives a very comprehensive list of different possible motifs found in the sequence, the results of which are displayed in Appendix D section 9.6.

3.1.9 Hydrophobicity Analysis

Hydrophobicity plots were performed on the whole sequence and each of the three domains separately. The results of such plots can be found in Appendix D section 9.7.

3.2 Repeats within the Proline Rich Region

The following are results from the analyses of repeats within the proline rich region.

3.2.1 Tandem Repeats Finder

The results from Tandem Repeats Finder can be found in Appendix D section 9.8. There is also a graphical version of this in the same section, showing the repeats as highlighting instead of just numbers.

3.2.2 Radar

The results from Radar were disappointing, but they can nevertheless be found in Appendix D section 9.8 and show why specially designed software was decided upon.

3.2.3 Specially Designed Software

The specially designed software constructed in Java produces a text file with the results in it. This output can be seen in Appendix D section 9.8 and has been transferred to a graphical format to help compare results to those from Tandem Repeats Finder.

A final repeat pattern was also observed in the sequence which is indicated by the repeating shape of the hydrophobicity plot near the beginning of the sequence. This was identified by eye and can be seen as the largest repeat listed in 9.8.

3.2.4 Other Proteins with the Repeats

When using BLAST to identify other genes with the repeats shown at the nucleotide level, all the results from all the searches were relatively insignificant. When using BLAST to identify other proteins that show the same repeat patterns at the amino acid level, only two of the repeat patterns were identified in other organisms with significant similarity. The results showing most significance were:

Search sequence: SPPYTPKPPVVSPPY

Protein Definition	Access ID	Score (bits)	E-value
Proline-rich cell wall protein [*Medicago sativa*]	3818416	42	0.002
MtN4 [*Medicago truncatula*]	2598599	42	0.002
Proline-rich protein 1 [*Vitis vinifera*]	22074208	41	0.002
Cell wall protein like [*Arabidopsis thaliana*].	7268262	37	0.045
PtxA [*Pisum sativum*]	2578444	37	0.045
Proline-rich protein [*Nicotiana glauca*]	6782440	32	1.5

Search sequence:
TPPPHGFKPPIVTPPYTPPYSPKPPHHHPYKPPHYTPKPPKVSPPYTPKPTPH

Protein Definition	Access ID	Score (bits)	E-value
Proline-rich cell wall protein [*Medicago sativa*]	3818416	72	4×10^{-12}
MtN4 [*Medicago truncatula*]	2598599	71	8×10^{-12}
Unnamed protein product [*Arabidopsis thaliana*]	11994732	64	9×10^{-10}
hydroxyproline-rich glycoprotein [*Sorghum bicolor*]	21627	63	1×10^{-9}
hydroxyproline-rich glycoprotein - sorghum	100753	63	1×10^{-9}

3.3 Sequence Conservation in the Cysteine Rich Region

During the analyses of the conservation of the cysteine rich region, the following results were obtained:

3.3.1 ClustalW

All the appropriate output from ClustalW can be found in Appendix D section 9.9. This is in the form of screen shots illustrating multiple alignments graphically.

3.3.2 Collation of Results from ClustalW

ClustalW gives alignment scores for the crossing of every possible combination of the 51 proteins. Whilst it would be impossible to list the 1275 alignment results in a numerical form, the table shown in 9.9.1 shows the matches as shades of green, with the brightest shades being the closest matches. The scores for the proteins against RS1 have been displayed in another table since these are most significant to the project.

3.3.3 Phylogenetic Tree

The phylogram generated by ClustalW can be seen in Appendix D section 9.9.2 and shows an indication of how each protein relates to every other protein.

Discussion

4.1 First Impressions

Working on a novel protein, the first impressions gathered offer a frame to base the remainder of the preliminary analysis on. Since nothing is really known about the protein other than it is split in to three domains and has a proline rich and cysteine rich region, it was fundamental to find out as much as possible about it.

4.1.1 Finding an Open Reading Frame

The open reading frame, shown in the output from GenScan in section 9.1, shows that there are no introns within the gene and that the encoding region is therefore all contained within one stretch of genetic code. This region is 1752 base pairs long and encodes for a protein constructed of 583 amino acids. Whilst the score for the probability of this match is 91.5%, there are no alternatives detected and so the likelihood of it being something other than this would be very low.

4.1.2 Laying Out the Results

To improve analysis in the latter parts of the project, the output from GenScan was formatted to show where the amino acids of the protein relate to the predicted open reading frame of the gene. The output shown in 9.2 is the outcome of that analysis and will help to describe detections later.

4.1.3 Promoter Sequence Identification

Section 3.1.3 has the output relating to the promoter sequence of the gene. This can be found in the non-coding region of the gene just before the open reading frame. The score of 1 on the highlighted line indicates that the promoter region identified is 100% likely to be correct according to the algorithms of the software that predicted it. Also highlighted is the legendary TATAA box which is used as a transcription factor. This, when bound to enhancers or enzymes, increases the transcription of the gene by helping to unravel the DNA double helix which in turn allows RNA polymerase to stretch open the DNA and mRNA is able to collate alongside the DNA, giving a template for the protein to become encoded from.

4.1.4 Locating the Domain Positions

It is already known that this protein has three domains within it, but to find them, *in silico* techniques can be used to identify known motifs that are found in the particular domains. In section 3.1.4, it can be seen that a number of domains are identified. The most significant of these were plotted in to a diagram shown in figure 5. It appears that the first 27 amino acids make up a signal peptide which is also shown to be trans-membrane from amino acid 7 onwards to the end of the region.

Many low complexity regions were also discovered, but these are due, mainly, to the proline rich region which is not only very heavily smothered by proline residues, but it also exhibits a lot of repeat patterns. These patterns mean that many results from BLAST based on this query will end up appearing similar when they are not.

The final domain, labelled 'Tryp_alpha_amyl', begins at residue number 498 and extends through to almost the end of the protein. The three domains do not cause an exact coverage of the protein and so were extended slightly to cause complete coverage. The signal peptide was extended up to peptide number 33 where the first long chain of proline can be found, and the final domain was extended to the end of the protein, with the beginning being at residue 498. These are illustrated as coloured text in latter results to show the significance of the domains.

The final cysteine rich region has a family name due to its conserved state and so there is information readily available on the likely function of this domain. It is said that this domain has uses in trypsin-alpha amylase inhibitors, seed storage proteins and lipid transfer proteins. However to jump to the conclusion that one of these will be the function of this domain would be to "jump the gun" since that may be the use of this domain when it is isolated in a protein, but this protein is a hybrid and this may have greater than expected effects on the function of any domain within it.

4.1.5 Motif Analysis

It would appear from the analysis of the peptide chain, that three main types of motif are present in the protein. The most prevalent of these is the rhodopsin-like G Protein Coupled Receptor superfamily signature. Four variations of this are shown at various positions

within the protein. The other two motifs are labelled "Type III secretion system inner membrane R protein' and 'Sugar transporter signature' which are both only seen on one occasion in the protein.

Following the links from the results allows the ability to learn more about each motif discovered. Rhodopsin-like GPCRs are part of a very large group of proteins that all share similar function. G-protein coupled receptors are varied in their use, but all contain the telltale domain that passes seven times through a membrane.[xxv] This is the result of six loops, three either side of the membrane. When the receptor on the outside of the membrane is stimulated by a ligand, the tertiary structure of the protein becomes changed. On the other side of the membrane, GDP (guanine di-phosphate) found within Gα (a subunit of GPCRs) becomes replaced by GTP (guanine tri-phosphate) which in turn causes GβGγ (further subunits of GPCRs) to disassociate from Gα. The newly activated Gα then activates an effector molecule which does the required work related to this initial ligand. Effectively, it is a messaging system that elaborately passes a signal across a membrane layer and causes an effect on the other side. These motifs are found mainly within the leader and cysteine rich regions.

The type III secretion system inner membrane R protein motif forms a structure used for transport across cell membrane layers. It is used by some viruses to inject proteins and DNA into the host cells. It is found within RS1 within the first part of the cysteine rich region.

The sugar transporter signature motif is found in proteins that are related to sugar transport in the organism. It is highly conserved between organisms and is thought to originate to a time before the split between eukaryotes and prokaryotes due to the very high similarity between today's proteins from these two groups. It is found in the RS1 gene within the middle of the cysteine rich region.

4.1.6 Studying Hydrophobicity

The hydrophobicity plots of the RS1 gene tell a lot about the structure of the protein, particularly in the repeating patterns of the amino acids, and in the location of different parts of this structure. The picture of the whole sequence gives the impression of

hydrophobic regions in both the leader and cysteine rich regions whilst the proline rich region remains hydrophilic. The leader region of the protein is composed almost entirely of hydrophobic residues, with a peak at around the tenth residue. This sort of hydrophobicity in the leader region is a key trait in proline rich proteins and is mentioned within papers talking about the analysis of the structure of this class of protein.[xxvi]

The proline rich region shows some very obvious repeating patterns through the hydrophobicity. This is expressed as a sequence of near identical peaks and troughs in the graph, based mainly towards the end of the region. Also to be seen is a very large repeat nearer the beginning of the sequence which was initially missed by the software being used to analyse repeat patterns. The only peak showing significant hydrophobicity in this region can be seen at 462nd and 463rd residue positions which lines up with two isoleucine residues next to each other. This amino acid is very hydrophobic and also very unreactive. However, it often takes part in recognition of hydrophobic binding products such as lipids.[xxvii]

The cysteine rich region appears to have three main peaks of hydrophobicity within it. This tends to indicate that the cysteine rich region is likely to be embedded into a membrane layer. If this is true, the points that show less hydrophobicity are likely to be on the turning points of the protein as it doubles back on itself. The first peak ranges from the beginning of the cysteine rich region to the 24th amino acid in the region. There are then three amino acids that show a lack of hydrophobicity followed by the second peak of twenty three amino acids. A further two amino acids show a lack of hydrophobicity and finally, a chain of 20 amino acids form the third peak, followed by a chain of fourteen hydrophilic residues.

By comparing these numbers to the actual chain, it may be possible to see if the cysteine residues are forming their usual disulphide bonds, which is what causes the protein to double back on itself (red is hydrophilic, blue is hydrophobic):

It would seem that whilst it is possible to align some of the cysteine residues, there is likely to be a far more complicated structure forming than simply three straight lines across the membrane. This is something that could only be confirmed by physical crystallography of the protein, which would be very difficult if the protein is embedded in a membrane layer.

4.1.7 Codon Usage

In section 9.3 there are three tables listed which illustrate codon usage within the entire genome of *Beta vulgaris* and from the RS1 gene. The final table shows the comparison of different residue abundance between the two. It can be gathered from these that:

- Whilst it doesn't appear apparent that any codons have preference over others, with particular interest in the proline codons, there is obvious over expression and under expression of certain residues.
- Proline, Tyrosine, Histidine, Threonine and Cysteine are all over expressed in the RS1 gene, although this may in part be due to the gene being considerably smaller than the entire genome and therefore not having such averaged out results. However, with Proline being over seven times greater expression than in the entire genome, the likelihood of this being due to chance alone is very slim. The same applies to Tyrosine and Histidine which are jointly over twice that of the expected percentages.
- Those amino acids expressing a lower than expected abundance are shown at the bottom of the third table. These include Tryptophan, Arginine, Glutamate, Methionine, Aspartate and Asparagine to name but 6 of the low scoring residues. Of these, Tryptophan is not found anywhere in the sequence and Arginine, Glutamate and Methionine are found over 10 times less than could be expected in the genome of *Beta vulgaris*. These figures are in part due to the very excessive abundance of proline since if one residue is found more, then another is forced to be found below the threshold. However, to find no Tryptophan where 3 in 200 residues should be Tryptophan is a significant result.

These results are fairly meaningless without any idea what each amino acid contributes to the protein and so the following is a short summary of the significance of each, taken from the webpage of Amino Acid Properties[xxviii]:

- **Alanine** - arguably the most boring amino acid since it is non-polar and has no preference to water. It does not form structures very easily and is often found in non-essential parts of the protein.
- **Cysteine** - will tend to form disulphide bonds extracellularly with other cysteine residues. Intracellularly the reducing environment can cause cysteine to bond to zinc and other metals to form bonds with other cysteine residues.
- **Aspartate** - frequently involved in active sites and binding sites, the polar charged properties of aspartate make it form hydrogen bonds with either water or positively charged residues. It has a rigid structure because it is small and will also bond to cations like zinc.
- **Glutamate** - almost identical to aspartate other than being slightly larger and therefore less rigid. This makes it less preferential in active sites than aspartate.
- **Phenylalanine** - an aromatic and hydrophobic amino acid that likes to be buried in the core of the protein. Tends to stack with other amino acids due to the aromatic ring. Often seen to bind to poly-proline proteins.
- **Glycine** - a very unique amino acid due to its single hydrogen atom side chain. This amino acid can fit places that other amino acids cannot; such as sharp turns. It is also able to bind to other molecules easily such as the G-x-G-x-x-G motif found in protein kinases to bind to ATP. It is very rarely substituted.
- **Histidine** - has a side chain which easily gains and loses protons, making it variable in terms of pH and charge. This can make it useful as part of a signal relay system where the charge can be altered to suit the requirements of the process.
- **Isoleucine** - has two non-hydrogen substituents attached to its beta-C side chain. This causes bulkiness of the backbone and means that beta sheets are a preferential structure for this residue. Being hydrophobic, it prefers to be buried in the protein core.
- **Lysine** - whilst the main body of lysine is hydrophobic, the side chain is strongly positively charged. This can cause bonds to other polar amino acids such as aspartate. Anions or carboxylate groups will often bond to this side chain, making it popular in active and binding sites within the protein.

- **Leucine** - this amino acid is similarly described to isoleucine, except for the preference to form alpha helices. Leucine is not often involved in active sites.
- **Methionine** - this hydrophobic amino acid is similar to cysteine in structure in that is has a sulphur atom attached to it, but because of the methyl group attached in methionine, the reactivity of it is greatly reduced, making it less involved in active sites.
- **Asparagine** - prefers to be on the surface of proteins due to being polar and often plays a part in the active or binding sites of the protein. It can also form a part of the Asn-His-Cys catalytic triad that is often seen in proteins.
- **Proline** - technically an imino acid, proline is able to adopt shapes that other amino acids cannot, including the triangular prism that a sequence of proline residues forms instead of a helix. Proline often gives kinks to the overall structure and although it is both aliphatic and hydrophobic, it is often seen to be on the surface of proteins.
- **Glutamine** - with a polar structure, glutamine prefers to be on the surface of the protein and can often be seen to be involved in the active and binding sites. Other polar or charged molecules interact with glutamine's polar side chain.
- **Arginine** - whilst the backbone of this amino acid is hydrophobic, the long side chain has a charged end which means that the protein is often found both internally and externally of the protein. It will often create hydrogen bonds to other charged molecules.
- **Serine** - with the ability to mimic proline, serine is often found on the surface of the protein, causing sharp bends in the sequence. It is quite common in the functional regions of the protein, binding with polar molecules.
- **Threonine** - with the bulky backbone caused by c-beta branching, the usual structure for threonine is beta sheets. It is usual for threonine to be found in the functional centre of the protein, binding to polar substrates and causing phosphorylation often as part of a signal transduction process.
- **Valine** - sharing the beta branching property of threonine, valine is usually found in beta sheets instead of alpha helices. It often binds to hydrophobic substrates such as lipids, although it is relatively unreactive and so rarely part of the functional component of the protein.

- **Tryptophan** - a hydrophobic amino acid that prefers to be buried within the core of the protein. The aromatic ring of tryptophan contains a non-carbon (nitrogen) component which allows it to bind, on rare occasions, to non-protein atoms. It, and other aromatic proteins, also have a tendency to bind to poly-proline proteins.
- **Tyrosine** - the hydroxyl group of tyrosine makes it more likely to bind to non-protein atoms than tryptophan or phenylalanine. It too has the tendency to bind to poly-proline proteins and will often stack with other aromatic amino acids.

With a complete summary of the amino acids, it is possible to make hypotheses towards the reasons for higher and lower expressions of amino acids. It would seem likely, with such a large number of proline in the proline rich region that this region of the protein will have a very unusual shape that twists and turns almost irregularly, creating shapes that no other amino acid can create. It would also seem that the threonine abundance may indicate a possible signal transduction function to the protein, as well as the histidine being a possible relay for a signal.

The lack of tryptophan from the protein entirely may be due to its size as a larger amino acid. It has a larger and bulkier structure which may not fit in to the tightly woven arrangement of the proline rich region. It would seem that the four lowest abundance amino acids within the protein are classified as larger amino acids which may be supportive evidence for this hypothesis.

4.2 Comparing the Whole Sequence to Similar Proteins

The results from BLAST gave a selection of proteins that could be considered similar to RS1. Unfortunately, to get these results, the low complexity filter had to be disabled and so they are likely to be less accurate. The sequences obtained from three of these results, chosen due to their apparent significance, were used to perform dotplots and multiple alignments in order to identify the properties of the RS1 gene.

4.2.1 Dotplots Offering A Graphical Interpretation

The dotplots that were performed in bioedit are able to show a significant amount of information regarding the conservation of particular parts of the sequence and can help in the identification of domains.

Section 9.4 shows the dotplot images which are referred to in this discussion. All three show a similar output which reinforces the argument that RS1 is split into three domains, the middle of which is highly repetitive and unconserved whilst the last domain is extremely conserved. The first part of the diagram in each case is the leader region of the protein. This covers an area ranging from the beginning of the RS1 gene through to the 33rd residue. There is a pale area to the left and the top of the graph that is identically gapped in both orientations which shows where sequence similarity is very weak. There is also a denser, vaguely diagonal, area in the top left of the graph where the sequence is similar, but not strongly so. This indicates a conserved region where the sequences of the two proteins are similar. This region appears to be roughly the same length in all three proteins.

There is a very dense box in the middle of all the graphs, where patterns are expressed in the direction of both axes. This would, at first glance, indicate good sequence similarity, but in fact shows quite the opposite. The banding effect creating the patterns is caused by similarity at almost all points within the block that covers both proteins. The length of this block is different in both sequences shown on each diagram which is illustrated by the fact that it is not a square block. What is actually happening here is that the vastly over expressed proline residue is matching up with more proline in the other sequence. Giving a good match as far as dotplot is concerned. The banding patterns are generated by the repetitive proline sequences which cause repetition of the pattern across the block. Effectively, every part of the block is similar to most of the rest of the block, indicating repetition. Low conservation of sequence is also identified by the lack of a direct diagonal line passing across the dotplot.

Finally, there is a very dense, narrow and straight line coming directly from the bottom right hand corner of the dense block. This is the beginning of the cysteine rich region. It continues right up to the end of the protein and is almost perfectly black. This is an indication of a region that is highly conserved. No matter which proteins are chosen and from which species, this line appears in the dotplot because it is so highly conserved, it has hardly ever changed.

4.2.2 Getting More Detail by Multiple Alignments

By doing multiple alignments of the sequences, the more specific information about the proteins being looked at can be observed. The output from ClustalW is displayed in section 9.5 and shows the multiple alignment of RS1 against two other proteins, which is all that is required to see similar information to that already explained in the previous section. When looking at the first 33 amino acids in the diagram, it can be seen that whilst the sequence similarity is fairly low between the three proteins, there are distinctly similar colours (which indicate amino acid property similarity) across the three amino acids. This indicates at least a small amount of conservation of this region.

However, now diverting attention to the proline rich region, which is everything from 34 up to 497 amino acids, it can be seen that the similarity between the sequences lives entirely in the fact that all the sequences have a high proportion of proline residues in this area. The consensus sequence at the bottom illustrates this perfectly by the fact that it is composed almost entirely from 'P' which stands for proline.

Finally, the cysteine region begins at marker 498, at which point, the look of the proteins change entirely. Suddenly there is high sequence similarity, with almost all the consensus sequence residues completed. This is the perfect example of a high conserved region which obviously serves an extremely important function for this protein. The next steps are to find out what this function may be.

4.3 Progress in the Proline Rich Region

It is very obvious, especially when considering the previous results, that this protein is not able to be analysed in its entirety. It is even suggested in some papers that proteins of this hybrid nature exhibit bifunctional properties in which the two halves of the protein offer different functions and that at some time in evolution these two functions became linked in some way which causes the hybridisation.[xxix] For these reasons, the two main domains of the protein were divided up for separate analysis. This section of the discussion deals with the proline rich region.

4.3.1 Understanding the Results

Having performed analysis on the repeat patterns of proline, which appear to be the only components of the region that are conserved at all, the results were placed in section 9.8. The two types of analysis performed were that of the nucleotide sequence and that of the amino acid chain.

During analysis of the nucleotide region, extensive results were returned which indicate a large number of repeats found in the sequence, which even overlap with other repeats on two occasions. When comparing the results of this to those of the amino acid chain repeats, it can be seen that on almost every occasion, where the protein has repeat patterns, they are mimicked in the nucleotide sequence. One of the questions to be answered is why the proline rich region is so repetitive. There are a number of hypotheses for this including that the function of the residues in the repetition is useful to the protein and so the number of active sites and binding sites are being doubled up. It could be that transcription in the past has caused repetition of DNA duplication, which in turn would be reflected in proteins. With this evidence, that the repetition is almost definitely on the nucleic level, it seems that the latter of these hypotheses may be correct.

4.3.2 Similarities from BLAST

In order to put these repetition patterns to use, it is important to take them in to a search query which will bring out results from similar gene/protein combinations. This may give more of a clue as to the use of this proline rich region. Tandem Repeats Finder gives a list of consensus sequences that should be searched for in BLAST, but however, these queries produced no significant results and so they were eventually generated from the results given by the specially designed software. The lack of results at the nucleic level could be expected since this region is of very low conservation and so will often mutate into different codons in different species, even though the amino acid chain is left unchanged. To find proteins that contain the proline rich region isolated from other domains and containing the same repeat patterns could be considered a particularly useful result.

The results shown in 3.2.4 are presented in two tables. These show the top five scoring matches for the only two repeat patterns to yield similarity to other proteins. In the first repeat pattern, SPPYTPKPPVVSPPY, the top five matches all have a copy of the cysteine

rich region attached to the end. The only result to show reasonable similarity that does not have this cysteine rich region is that from *Nicotiana glauca*.

The second repeat pattern, TPPPHGFKPP IVTPPYTPPY SPKPPHHHPY KPPHYTPKPP KVSPPYTPKP TPH, is found in many more proteins than the shorter pattern, and some of the results are duplicated from the previous search. More of the results from this search appear to lack the cysteine rich region, with particular interest in the two hydroxyproline-rich glycoproteins at the bottom of the table.

4.4 Solving the Mysteries of the Cysteine Rich Region

The second domain to be looked at is of course the cysteine rich region which has so far proven to show high similarity to other known cysteine rich regions. The analysis results are found in section 9.9.

4.4.1 Understanding the Results

The first two figures in 9.9 are showing multiple alignments of the cysteine rich regions of 51 proteins, selected for their apparent similarity to the RS1 gene. The top of these figures illustrates similarity between the sequences by highlighting the amino acids with similar properties in the same colour. The histogram beneath the sequences shows how well each residue matches across all the sequences. It appears that whilst the entire cysteine rich region is highly conserved, the second half is more so than the first, with exception to two residues which show no bar in the histogram. When ClustalW is instructed to highlight only those that show the highest conservation, the second figure is produced, and as can be seen, the majority of this highlighting is on the right side of the screen; the end of the cysteine rich region.

ClustalW was also able to supply alignment scores for all the possible crosses within the 51 proteins. This, when transcribed to excel, gives the map shown on the next page of the appendix. Lighter green patches are an indication of good alignment scores. In order to use this information to do more research in to the purpose of the cysteine rich region, the scores for alignment between RS1 and the other 50 proteins are displayed in a table following this. It can be seen that 17 of these proteins show alignment scores of 70 and above, which is again evidence of such high conservation of this domain.

Finally, a phylogram shown in 9.9.2 gives a visual impression of the distance between each of the proteins in terms of their similarity to one another. By tracing the lines of the diagram, the longer the route is between two proteins, the farther apart they are to one another from a sequence point of view.

4.4.2 Most Similar Results

The results from the cysteine rich region similarity analysis show that there are a number of proteins showing high levels of similarity to RS1 in the cysteine rich region. The first ten of these will present a good basis for trying to deduce more information about the purpose of the cysteine rich region.

Here is a list of the first ten proteins showing to have the highest alignment scores to the cysteine rich region of the RS1 gene:

Protein Definition	Access ID	Alignment Score
Probable cell wall protein - garden pea	7446483	80
Proline rich protein - apple tree (fragment)	7522211	80
Proline rich protein [*Lycopersicon esculentum*]	19521	78
36.4 Kd Proline-rich protein [*Lycopersicon esculentum*]	1709767	78
MtN4 [*Medicago truncatula*]	2598599	77
Cytokinin-induced proline rich protein - southern Asian dodder	7446488	77
Cell wall protein - *alfalfa*	1076501	75
Proline-rich protein - *Solanum brevidens* (fragment)	2147979	74
Putative proline-rich protein [*Solanum brevidens*]	15022163	74
Salt-inducible protein RF2 [*Medicago sativa*]	1092083	74

It would seem, at first glance, that none of these proteins have the cysteine rich region existing without the proline rich region. This further strengthens the idea that although the two domains appear to give reason to believe that they have different functions, they work as a team to achieve the same goal.

4.5 Inferring Function and Structure

Function and structure can be inferred from the analogy that:

Similar Sequence ≈ Similar Structure ≈ Similar Function

The fact is that each of these elements is interlinked to some extent. The structure is entirely based on the sequence and the function is entirely based on the structure. Therefore, if there is a minor change to the sequence, this will often only alter the function by insignificant amounts. As a result, information from the two regions analysed above can now be used to help define the function of the domains.

4.5.1 Using Information from the Proline Rich Region

The results from the proline rich region are very vague due to the fact that the proline rich region is conserved so little. It also does not help that most of the results showing similarity in the proline rich region repeats also have the cysteine region attached, offering possible contamination of results since proline is supposed to be isolated for analysis.

The protein from *Nicotiana glauca* shows signs of being related to drought stress tolerance in the stomatal region of the nicotine plant. This relation to stress on the cell is supported in all the results of the first table in section 3.2.4 as all the proteins appear to relate to stress tolerance. The protein from *Medicago sativa* appears to relate to salt tolerance[xxx]; the protein from *Medicago truncatula* is important in cell wall structure during development[xxxi]; the protein in *Vitis vinifera* is wound induced. The other two proteins listed do not appear to have any reference material directly relating to them.

The proteins from *Medicago sativa* and *Medicago truncatula* are shown again in the second table from section 3.2.4. This strengthens their relation to the RS1 gene as they are the top search results in both cases. Little has been written about the function of the other proteins listed.

4.5.2 Using Information from the Cysteine Rich Region

The results from the cysteine rich region, despite being very well honed in upon because of the high conservation of the region, do not tell very much about the cysteine rich region as

an individual domain since all the results that are significant are also joined to the proline rich region. This tends to make the outcomes the same as those from the proline rich region.

It is interesting that some results shown to be similar in the analysis of the proline rich region are also appearing in the results for the cysteine rich region. The one particular protein that catches some attention is that from *Medicago truncatula*, MtN4, since this has been rated highly in both the searches from the proline rich region.

The cell wall protein of *alfalfa* appears to be salt-induced[xxxii] which again demonstrates that the RS1 protein is likely to be related to some form of stress induction. And the final protein of the ten listed states very clearly in the title that it is also salt-induced.

4.6 Making Conclusions

4.6.1 Recapping on Objectives

Before making final conclusions relating to the aims of the project, it is important not to lose grip of the actual objectives. These are, after all, the basis upon which all the analysis has been performed. Therefore, to give the greatest relevance to these objectives, they are re-listed here:

1. Discover the basic biology of the gene, including where the encoding open reading frame is and basic assessment of codon usage, protein domains and promoter region.
2. Look at the proline rich region to find out the major and minor repeats within the sequence. Take this information and try to find other proteins with similar repeats within them.
3. Look at the cysteine rich region to find out how much homology it has to other proteins and find out whether it is always found with a proline rich region or only on its own. Perhaps try to construct some form of phylogenetic tree, showing the genetic origins of this protein.
4. Use the other proteins found in objectives 2 and 3 to infer function, structure and maybe location. Supplying information that may be useful to be used *in vitro* to test this protein further.

5. Discuss the evolutionary origins and relationships to other proteins in the family and infer whether the protein has a successful future in the evolutionary path or if it will be lost forever.

4.6.2 Objective 1

Getting a foot hold and deciding where to start with the gene turned out to be the hardest part in terms of getting to grips with a gene that is novel and unknown. At this point nothing is known about what has been presented for analysis and with no definitive starting point; it was easy to rush in to doing many different types of analysis without a clue why they were being performed. Only once the picture began to form was it obvious where the next steps in the metaphorical mountain could be placed.

After the beginning parts of the analysis, it was obvious that the gene contains an open reading frame of 1752 nucleotides, starting at nucleotide 3253, which encodes for a 583 residue protein. The gene has a promoter region beginning at nucleotide number 2167 and ending at 2217, containing a TATAA box towards the end. The protein is made up of three domains, including a leader region, a proline rich region and a cysteine rich region. These regions appear to have different properties about them, but in particular the proline rich region appears to be very repetitive and lowly conserved, whilst the cysteine rich region appears to be very highly conserved. This is confirmed by both dotplot analyses and multiple alignments of seemingly similar sequences.

Codon usage analysis introduces further information regarding the construction of the protein, including the fact that the protein shows a higher than normal abundance of Proline, Tyrosine, Histidine, Threonine and Cysteine. It also shows a lower than normal abundance of Trytophan (which is missing entirely), Arginine, Glutamate, Methionine, Aspartate and Asparagine. The relevance of these abundances of particular residues is not entirely certain but it is thought that the shape of the proline rich region will be very angular due to the structural organisation of proline. It is also thought that the abundance of histidine and threonine may be an indication that the protein performs signal inductions.

The hydrophobicity plots performed on the protein indicate that there is a strong possibility that the cysteine rich region passes through a membrane layer and that the cysteine within

this may help to maintain a number of folds which allow the protein to pass three times through the layer. Motifs identified as present in the protein, particularly in the cysteine rich region, indicate that the sequence may be a G-protein coupled receptor, even though it only passes the three times through the membrane.

4.6.3 Objective 2

Whilst analysing the proline rich region, many repeats were found. These major and minor repeats could also be seen at the nucleic level as well as the peptide level. Two of the repeats in the peptide chain could also be seen to exist in other proteins that are already documented. These repeats, namely SPPYTPKPPVVSPPY and TPPPHGFKPPIVTPPYTPPYSPKPPHHHPYKPPHYTPKPPKVSPPYTPKPTPH, are found in other proteins with a similar structure involving a proline rich and cysteine rich domain. However, they cannot be found to exist in a protein consisting of only the proline rich region, apart from on one occasion in a protein from *Nicotiana glauca* which shows signs of being related to drought stress tolerance in the stomatal region of the nicotine plant. Other proteins that were found in the same way also show expression during stresses including salt tolerance expressions and wound induced expression. There are also some proteins showing developmental cell wall strengthening properties which score highly in the similarity scores.

4.6.4 Objective 3

Other proteins containing a cysteine rich region were isolated and compared by multiple alignments to the cysteine region of the RS1 gene. These analysis techniques have led to the manufacture of a table which shows how the similarity between proteins vary that contain this cysteine rich region and this in turn has led to finding the proteins with most significant similarity to RS1. These proteins have shown similar results to the proline rich region and has also shown that almost every protein described that has the cysteine rich region also has the proline rich region, perhaps indicating that the two domains work separately, but have an interaction between them, linking their use. Again, there are results indicating that the proteins that have appeared are related to stress induction, but since this is not the isolated function of the cysteine rich region, it is hard to decipher the exact function of this domain of the protein.

A phylogenetic tree was also created from the results from the multiple alignments, and this gives a graphical view showing the similarity, or indeed distance, between different proteins containing the cysteine rich region.

4.6.5 Objective 4

It would seem, following the overall results of the project, that the protein is very likely to be both a cell membrane and cell wall protein. The motif results indicating that the cysteine rich region may be a form of G-protein coupled receptor, backed up by the hydrophobicity plots saying that the cysteine rich region is a membrane spanning protein, would indicate that there is a high possibility that the cysteine rich region resides in the cytoplasmic membrane near the cell wall. The short hydrophilic tail at the end of the protein remains dangling in the cytoplasm of the cell, waiting to receive a signal that the cell wall must be reinforced. The cysteine rich region passes through, back and through the membrane again, moving on to the proline rich region which may be embedded in to the cell wall, according to the results that have been brought about through analysis of the two domains. Most of the proteins mention that they are embedded in or on the cell wall and it would seem that the unusual structures that proline forms would be the perfect candidate for weaving through the cell wall structure. The tail of the cysteine rich region would receive a signal in the form of a hormone or damaged proteins by binding to it with the high abundance of binding residues in that area (note the tyrosine residue on the very tip of the tail). This would in turn change the tertiary structure of this part of the protein and cause a transduction reaction on the opposite side of the membrane. This would stimulate the proline rich region to change the structure of the cell wall, and strengthen or weaken it as is necessary. Meanwhile, the leader region of the protein also flagged up clues that it may be trans-membrane during the domain analysis. Being described as a signal peptide perhaps this leader region passes back out of the cell wall and through the cytoplasmic membrane too. It could be suggested that perhaps stimulation of both the leader region and the cysteine rich region would be required to cause an effect in the proline rich region.

This of course is all just a hypothesis, but could perhaps lay a pathway for the *in vitro* analysis of this protein to see if the hypothesis is correct. Without this stage, it is impossible to decide whether any of the hypotheses produced by bioinformatics techniques

are valid or not. Bioinformatics is not a replacement for lab work, but can save many hours of time spent stabbing in the dark in a laboratory.

One suggestion for the work that could be performed in the laboratory could be to use splicing enzymes to reduce the full length of the RS1 gene. By removing elements of the gene or protein such as the leader region or the cysteine rich region, it may be easier to see what significance they hold on the function of the protein. This sort of work was performed on a protein that lost the ability to bind to DNA when the proline rich region was removed by Eiko Kanaya et al.[xxxiii] But before this, the function of the protein as a whole must of course be known.

4.6.6 Objective 5

As has already been mentioned; the protein is made up of three domains, two of which are thought to serve the main functions of the protein. These two domains are the proline rich and cysteine rich domains. The proline rich region is very highly unconserved, which varies dramatically from species to species and is even known to vary incredibly within the same species. This lack of conservation means that any evolutionary trends present, or evidence of an evolutionary pathway are lost due to the incredible adaptability of the region.

However, the cysteine rich region is one of very high conservation, which in turn makes it vary by only minor amounts between species and even kingdoms. This conservation of sequence leaves a very definitive evidence trail, with which the evolutionary origins of the region could be distinguished. This information leads to the results demonstrated in the analysis of the cysteine rich region and introduces more information as to the functional use of the region as inferred from proteins of similar sequence. The phylogenetic tree illustrated in section 9.9.2 of Appendix D shows the distances that exist between the sequences that were analysed in this project. It can be seen that the distance between these proteins varies by only the smallest amounts and that RS1 does not demonstrate a particularly big jump in evolution due to its close proximity to the left hand side of the diagram.

All this leads to the conclusions that the evolutionary pathway followed by the cysteine rich region is not one of a dead end, and it would appear that this region has been evolving for quite some time. It can only be assumed that each minor change to the region is demonstrating a more refined version of the equivalent region from the generations before. However, these results should be limited entirely to the cysteine rich region since this protein has already been identified as a hybrid. The lack of conservation and similarity, even at the repeat level, that can be found in the proline rich region seems to be an indication that it may be quite the opposite of the cysteine rich region. It would seem that the proline rich region, with its drastically varying lengths and almost complete lack of similarity to any other proline rich regions is the perfect example of a protein domain unable to maintain itself in evolution. Evolution is favourable to good trends and appears to only offer positions for slight variations to these in future generations. It appears that the proline rich region cannot be entirely useless, but it may not be very good at what it does and is only able to gain entry into each new generation by the highly refined cysteine rich region to which it is bound. The cysteine rich region is acting as a passageway for the proline rich region and whilst evolution will favour the cysteine rich portion of the protein, it is unable to separate it from the cysteine rich region and so they progress together, despite the high mutation rate and probable lack of function that the proline rich region provides.

References

[i] http://www.mpiz-koeln.mpg.de/pr/garten/schau/varaltissima/Sugar_beet.html - The Max Planck Institute for Plant Breeding Research

[ii] **Hong, J.C., Nagao, R.T., and Key, J.L.** (1989). "Developmentally regulated expression of soybean proline-rich cell wall protein genes" *Plant Cell* **1**: 93-943.

[iii] **Hong, J.C., Nagao, R.T., and Key, J.L.** (1989). "Developmentally regulated expression of soybean proline-rich cell wall protein genes" *Plant Cell* **1**: 93-943.

[iv] **Robertson D., et al.** (1997) "Differential extraction and protein sequencing revels major differences in patterns of primary cell wall proteins from plants" *The Journal of Biological Chemistry* **272(25)**: 15841-15848

[v] **Showalter A.** (1993) "Structure and Function of Plant Cell Wall Proteins" *The Plant Cell* **5**: 9-23

[vi] **Josè-Estanyol M., et al.** (1992) "A Maize Embryo-Specific Gene Encodes a Proline-Rich and Hydrophobic Protein" *The Plant Cell* **4**: 413-423

[vii] http://www.russell.embl-heidelberg.de/aas/Pro.html - Information about Proline written by Robert B. Russell, Matthew J. Betts and Michael R. Barnes

[viii] **Edwards S., et al.** (2003) "The proline-rich region of mouse p53 influences transactivation and apoptosis but is largely dispensable for these functions" *Oncogene* **22**: 4517-4523

[ix] **Zarrinpar A., et al.** (2003) "The structure and function of proline recognition domains" *Science's stke* www.stke.org/cgi/content/full/sigtrans;2003/179/re8

[x] **Zarrinpar A., et al.** (2003) "The structure and function of proline recognition domains" *Science's stke* www.stke.org/cgi/content/full/sigtrans;2003/179/re8

[xi] http://www.russell.embl-heidelberg.de/aas/Cys.html - Information about Cysteine written by Robert B. Russell, Matthew J. Betts and Michael R. Barnes

[xii] http://genes.mit.edu/GENSCAN.html - The New GENSCAN Web Server at MIT by Burge, C. and Karlin, S.

[xiii] http://us.expasy.org/tools/dna.html - ExPASy translation tool

[xiv] http://www.fruitfly.org/seq_tools/promoter.html - The Neural Network Promoter Prediction version 2.2

[xv] http://smart.embl-heidelberg.de/ - Simple Modular Architectural Research Tool (SMART)

[xvi] http://www.kazusa.or.jp/codon/ - CountCodon Program and Database

[xvii] http://www.ncbi.nlm.nih.gov/BLAST/ - Basic Local Alignment Search Tool

[xviii] http://www.mbio.ncsu.edu/BioEdit/bioedit.html - BioEdit: Biological Sequence Alignment editor for Windows 95/98/NT/2K/XP

[xix] http://www.ebi.ac.uk/clustalw/ - ClustalW: A multiple alignment editor

[xx] http://motif.genome.jp/ - MOTIF

[xxi] http://arbl.cvmbs.colostate.edu/molkit/hydropathy/ - Colorado State University hydrophobicity plot Java applet

[xxii] **Benson G.,** (1999) "Tandem repeats finder: a program to analyze DNA sequences", *Nucleic Acid Research* **27(2)**: 573-580

[xxiii] http://www.ebi.ac.uk/Radar/ - Radar protein repeat analysis hosted by European Bioinformatics Institution.

[xxiv] http://www.ebi.ac.uk/clustalw/ - ClustalW hosted by European Bioinformatics Institution.

[xxv] http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/G/G_Proteins.html - Kimball's Biology Pages by Dr. John W. Kimball of Andova, MA

[xxvi] **Györgyey J., et al.** (1997) "Expression of a novel-type small proline rich protein gene of alfalfa is induced by 2,4-dichlorophenoxiacetic acid in dedifferentiated callus cells" *Plant Molecular Biology* **34**: 593-602

[xxvii] http://www.russell.embl-heidelberg.de/aas/Ile.html - Information about Isoleucine written by Robert B. Russell, Matthew J. Betts and Michael R. Barnes

[xxviii] http://www.russell.embl.de/aas/ - Amino Acid Properties by Robert B. Russell, Matthew J. Betts and Michael R. Barnes

[xxix] **Josè-Estanyol M., et al.** (1992) "A Maize Embryo-Specific Gene Encodes a Proline-Rich and Hydrophobic Protein" *The Plant Cell* **4**: 413-423

[xxx] **Bastola D.R., et al.** (1998) "Alfin1, a novel zinc-finger protein in alfalfa roots that binds to promoter elements in the salt-inducible MsPRP2 gene" Plant Molecular Biology 38(6): 1123-1135

[xxxi] **Wilson R.C., et al.** (1994) "A New Proline-Rich Early Nodulin from Medicago truncatula Is Highly Expressed in Nodule Meristematic Cells" *The Plant Cell* **6**: 1265-1275

[xxxii] **Deutch C.E., et al.** (1995) "Post-transcriptional regulation of a salt-inducible alfalfa gene encoding a putative chimeric proline-rich cell wall protein" *Plant Molecular Biology* **27(2)**: 411-418

[xxxiii] **Kanaya E., et al.** (2001) "Non-sequence-specific DNA Binding by the Filamentous Flower Protein from *Arabidopsis thaliana* Is Reduced by EDTA" *The Journal of Biological Chemistry* **277(14)**: 11957-11964

Unreferenced Materials

Berhardt C. and Tierney L. (2000) "Expression of AtPRP3, a Proline-Rich Structural Cell Wall Protein from *Arabidopsis*, Is Regulated by Cell-Type-Specific Developmental Pathways Involved in Root Hair Formation" *Plant Physiology* **122**: 705-714

Ebener W., et al. (1993) "Expression of DcPRP1 1s Linked to Root Formation and 1s lnduced by Auxin Treatment" *Plant Physiology* **101**: 259-265

Edwards S., et al. (2003) "The proline-rich region of mouse p53 influences transactivation and apoptosis but is largely dispensable for these functions" *Oncogene* **22**: 4517-4523

He C.-Y., et al. (2002) "A soybean gene encoding a proline-rich protein is regulated by salicylic acid, an endogenous circadian rhythm and by various stresses" *Theory of Applied Genetics* **104**: 1125-1131

Josè-Estanyol M. and Puigdomènech (1998) "Developmental and Hormonal Regulation of Genes Coding for Proline-Rich Proteins in Female Inflorescences and Kernels of Maize" *Plant Physiology* **116**: 485-494

Kay B., et al. (2000) "The Importance of Being Proline: the interaction of proline-rich motifs in signalling proteins with their cognate domains" *The FASEB Journal* **14**: 231-241

Menke U., et al. (2000) "StGCPRP, a Potato Gene Strongly Expressed in Stomatal Guard Cells, Defines a Novel Type of Repetitive Proline-Rich Proteins" *Plant Physiology* **122**: 677-686

Orengo C.A., Jones, D.T. and Thornton, J.M. (2003) "Bioinformatics: Genes, Proteins and Computers" *BIOS Scientific Publishers Ltd.*

Appendix A

6.1 Terms of Reference

Name: Stuart David James McHattie
Course: MSc Bioinformatics
Supervisor/Proposed by: Dr. Nigel Scott

Background Information

Where do proteins come from? Evolve or be lost? An interesting gene found within *Beta vulgaris* (sugar beet) exists in which there are three domains: a leader region, a proline rich domain and a hydrophobic cysteine rich region. This is found to be expressed particularly in the storage organs of *Beta vulgaris* but also throughout. There are various versions of this gene containing varying lengths of the proline domain but they all seem to have a highly conserved hydrophobic cysteine rich domain, making it very interesting from the point of view of evolution. It is also known to be part of a large family of genes found to be expressed in many plants and in many forms.

Not very much work beyond the very basic comparison of different versions of the gene and basic statistical analysis of the gene has occurred to date, and this gives the basis of the project a wide scope for use of *in silico* methods to find out more about what this gene does or why it shows such high conservation of sequence in some domains whilst not in others.

Aims

There are several unanswered questions which form the basis of the aims of this project. These are as follows:

1. What are the main characteristics of the gene?
2. Where does it originate from and what are its homologues?
3. Why are there such distinct domains within the gene and what is the function of each?
4. Why are parts of the gene so highly conserved when other parts are not?
5. Is it possible to predict a structure from the sequence and homology to other known structures?

Objectives

In order to answer the above questions, a number of tools on the internet and available to download via the internet must be used. Firstly, it is important to find out what research has been done in the past to identify the answers to the above questions, and to build from that point onwards towards finding the answers to the others. The first objective will be to carry out basic analysis on the genomic sequence containing code for the *Beta vulgaris*

gene. This means to locate homology to other genes and to compare the domains of the gene to identify motifs that can be found in genes of known function. This should make it possible to get a rough idea of where the gene originates from and perhaps a little more information about the use of the gene. It may be necessary to tweak or develop more software than is currently available so as to gain the best information possible about the gene, but this cannot be decided until closer to the time.

Secondly, the gene can be compared on a larger scale, to include genes from other plants, and perhaps those which have had more work done on their analysis. This will include looking specifically at domains that are found in other genes that do not necessarily have the other domains within them. This will give insight into the function of the gene and perhaps answer questions about the highly conserved regions of the gene that cannot be answered in other ways. It may also show how the gene has evolved and show which parts are more likely to change than others.

Lastly, the final objective will be to use all the information gained from the first and second parts of the assessment and to investigate the structure and function of the gene, as well as find out why it is exhibited mainly in the storage organ, but not excluded from elsewhere. By nature of the gene expressing a trans-membrane protein, the 3D structure is very difficult to observe since it is partially entangled in the membrane. If enough information can be gathered from *in silico* approaches, it may be enough to offer more information from *in planta* analyses since it is easier to prove or disprove a theory than it is to generate unknown information from nothing. Once again, if existing approaches do not offer all the information required, it may be important to tweak or manufacture software to be suitable for analysis of this gene.

Required Resources

The only resource required for this project will be the internet since both papers about previous work, and all the tools that may be required for the analyses can be found online. If programming is required at any stage, dependant on the choice of programming language, this may too be available online. Further resources should be easily obtainable if and when required, including high power computing equipment which can be found at De Montfort University.

Summary Schedule

The following is a very rough guideline of a schedule which shows the stages of the the project and when they should be completed by:

Start	**End**	**Details**
9th June		Meeting with supervisor to discuss details of the project
10th June	17th June	Look at available material and previous work done on the gene
16th June		Second meeting with supervisor
18th June		Submission of "Terms of Reference"

21st June	2nd July	Repetition of work already performed on the gene to confirm previously discovered information
5th July	15th July	Assessing new approaches on the discovery of information about the gene
16th July		First PMP meeting and progress assessment
19th July	29th July	Attempting to discover the three dimensional structure of the protein
30th July		Progress assessment
2nd Aug	19th Aug	Looking at comparable aspects of related proteins in other plants by homology
20th Aug		Second PMP meeting and progress assessment
23rd Aug	9th Sept	Analysis of data and compilation of the report
10th Sept		Submission of the written report
17th Sept		Oral presentation of the material

6.2 Progress Reports

6.2.1 Report 1

Programme Title: MSc Bioinformatics

Name: Stuart David James McHattie **Assessment Period :** 07/06/04 - 26/06/04

Project Title: Where do proteins come from? Evolve or be lost? **Report Number:** 1

Objectives for Period (refer to previous report)

Review available material on hybrid cell membrane genes in plants and begin attempting to repeat this analysis for the *Beta vulgaris* gene.

Summary of Progress for Period (identify evidence of progress)

11 papers collected and in the process of being read. Basic BLAST searches and proline repeat alignments performed by hand.

Problem Areas and Suggested Solutions

Very little material found specifically about hybrid cell membrane genes in plants. Refining of search queries to look at domains separately may be required.

Objectives, Deliverables & Plan for Next Period

Analysis of the genomic sequence to identify transcription factor binding sites and transcription translation signals as well as coding regions and domains. Identification of repeats within the proline region. Alignments to known database results. More analysis of literature relating to sequences on structure and function analysis. Analysis of the 2D and 3D structures and trans-membrane sections of the domains.

Date of Next Review: 16/07/04

6.2.2 Report 2

Programme Title: MSc Bioinformatics

Name: Stuart David James McHattie **Assessment Period :** 26/06/04 - 16/07/04

Project Title: Where do proteins come from? Evolve or be lost? **Report Number:** 2

Objectives for Period (refer to previous report)

Analysis of the genomic sequence to identify transcription factor binding sites and transcription translation signals as well as coding regions and domains. Identification of repeats within the proline region. Alignments to known database results. More analysis of literature relating to sequences on structure and function analysis. Analysis of the 2D and 3D structures and trans-membrane sections of the domains.

Summary of Progress for Period (identify evidence of progress)

Raw data acquired from GenScan, RepeatView, SMART (Domain prediction), PFam (Domain database), Tandem Repeats Finder and Promotor sequence predictors. BioEdit used with ClustalW to make multiple alignments of BLAST results and Dot Plots drawn to show repeat patterns and similarity in the Cysteine Rich Region. MOTIF used to identify Motifs in the protein.

Problem Areas and Suggested Solutions

When attempting to get a 3D image of the sequence, there were no close homologues. This is mainly due to the Proline Rich region and the Cysteine Rich region may show better results when isolated due to its conserved nature. There is much raw information to be looked at and some inferences need to be made before it becomes overwhelming.

Objectives, Deliverables & Plan for Next Period

By looking at more detail into the motifs within and surrounding the protein at both the nucleic and peptide level, more information about the function of the protein can be inferred. Taking BLAST search results, it may be possible to construct a phylogenetic tree of sorts to determine the origin of this protein when making alignments between them and the target protein. Hydrophobicity plots of the Cysteine rich region will indicate the folding pattern of the proposed trans membrane part of the protein.

Date of Next Review: 11/08/04

6.2.3 Report 3

Programme Title: MSc Bioinformatics

Name: Stuart David James McHattie **Assessment Period :** 16/07/04 - 11/08/04

Project Title: Where do proteins come from? Evolve or be lost? **Report Number:** 3

Objectives for Period (refer to previous report)

By looking at more detail into the motifs within and surrounding the protein at both the nucleic and peptide level, more information about the function of the protein can be inferred. Taking BLAST search results, it may be possible to construct a phylogenetic tree of sorts to determine the origin of this protein when making alignments between them and the target protein. Hydrophobicity plots of the Cysteine rich region will indicate the folding pattern of the proposed trans membrane part of the protein.

Summary of Progress for Period (identify evidence of progress)

Contrary to the objectives, but reaching an equally useful outcome, the codon usage within the protein has been assessed and compared to the codon usage within the Beta vulgaris genome. A program was designed to find the tandem repeats within the protein sequence. The cysteine rich region underwent multiple alignment to other cysteine rich regions to give alignment scores. These identify which regions are most similar and will hopefully be able to generate some kind of phylogenetic tree to identify where the protein comes from. A hydrophobicity plot was made to help identify folds in the cysteine rich region and spot larger repeats.

Problem Areas and Suggested Solutions

The program does a very good job of handling many large sequences very quickly, but suffers from minor inaccuracies in identification as a result. For this reason, very large repeats are not always identified. This will not be fixed due to time constraints and the fact that it is a minor problem.

Objectives, Deliverables & Plan for Next Period

By taking the major repeat units of the proline rich region and searching in BLAST, it should be possible to identify proteins that have a similar proline rich function. Similarly, by observing the proteins which have shown high similarity in the cysteine rich region, it may be possible to identify the function of this region and to see if it is ever found without the attached proline rich region. By taking a closer look at the shape of the hydrophobicity plot, more information about the exact shape of the folding could be inferred.

Date of Next Review: N/A

Appendix B

7.1 The Genetic Code for RS1

GGATTATTTTTTAGATCCGACTCCAACCCTCCTCGACTCGGAGTCGGACTTTTTAAACGGAGTCGGATTACTT
TTTTGTCGGAGTCAACTCGGATTTTTTCTTCGGTTCAGAGCGCACTAGGGTCGGATTCGGATCGGATTGCCAT
CCCTAGGTAGTATTAAATGTAATTACAAAAATACTGCTACTAAATTTATGAGAAAATAATTTAACTAGCTACA
AAAATACTGCTAGTATAAAATAATTTATGAGAAGATATTGAAGGTAGGGATGGGCAAGGGTAGGGCTCGGGTC
GTCCAGCTGGACCCTAGACCCGGGCCCTAAATATTTTTGCAGACCCGGACCCTAAGGGTCTGGAAAAAATGGA
CCCTGACCCGGACCCTTAGGGTCTGGAGGGTCTAGAGGGTCCAGGCCTAAATTTTTTATTTTGTCAAATTTTC
AGCATTATTAATATCAATAATCATCTGAAATTCACATGAAACAAACACAAAAAATCGCATGAAACAAACATAA
GAATTGGCATGAAACAAACACAAACATATAAATTGAAAAAAACGAAACATACACAAACATAAACTTATAAACT
GAAAAAAAACGAAACAAACACAATTCACAAACATGTAAACTGAAAAAAAGAAACAAAAACACAAACATACAAA
CTGAAAAAAACGAAACAAACACAACTTACATAAGAGTTCAGAATGGGTGTTTTAGTTTAGGTTTTAGTCTTTT
AGAAAATTAGTTTTTTTTTTTAAAAAAAAAGTTAAAATGTATATATTAAATAAGTTTAGGGTCTAAGGGTCGGG
TCTGGGTCCAAAATTTTAAGACCCAGACCCGGACCCTAAAAAATTTACTTGGACCCAGACCCGGACCCTTAGG
GTCTGAAAAAGTTGGACCCAGACCCTTAAACTAGGATCGGGTCCAACAGGGTCCCAGTAGGGTCCTGGACCAT
GCCCATCCCTAATTAAAGGTAGGGGTGGAAGTTCGATCTTCGATTTGGTTTTAGAACAAAACCAAACCAAACC
AAACCGAAATAAATTTGGTTTCAAAATATCCAATCCAAATCCAAACCAAACCAAAATTTCAAAGCCAAAACCA
ACACAAACCAATCTTTTTAGTGGTGACTTACTGACTTAGGAGAGAAAGAAGCAAGAAACCCCAACTTACTGAC
TTACGTCTTAGGGGCTTTGGGTTTACGAGAAAAGGAAATTCGAAGTGGCAAAATCCCAAATTTCGAGAGTAGG
AATGGGGACGGGGATTTACATATCTATATTTTTGTTTGGATTGGTTTTTTCCGGTTTTCAAAAAACAATACCA
AACCAAAAAATTGAAAAAAAAAAAAGAAAATCCAAACCAAATTACTAATTCGATTTGGTATGATTTGATTTTCG
GTTTTTAACCAAATAACTTACACCCTTAATTGAAAGAAGGATGCGCTATATTATGGGACAAAGATATTAAAAT
TTAACAAAATATCTTTTCAATATTATAATTGTGTGCATACTGCATTGTATACTTTATACTGTTTTCATGACTC
GTATGGATAAAAAAAAAAAATTGTGTGCATTGTTGTGCCTGTGGACTATGGTGTGACTTCACGTGGCTTTTGAG
ACATATTGCATAAGTCACGGACCCAATAGTCCTTGTCTTTTAGCCCTTAGGTGCCAGTTAAAGGCACTACATG
TTAAAGTTCATATCTCAATCATTGAACACCTCTTCCTAAGCAACATTTTTTAATTACAATTAATAACATAATT
AAGGGACCTAATTTCTATTAATGACTAATCCATGAACCCCATTTGCAAATATACATCCGAAAATGGTTCAAGT
TGTAGCGCACCCTATAAATCCCAAGTCAATTAATAGTGACAAAAGCATAAAAAATACCAATAATTTTTCTTCA
AATAAATCCAAGGACGACAAGAAAAGGGCTATCAAATAACAATCACAATGCCTAAGAAATGGAGCACAATTTC
TTGAAACTAGGGGCAGATTTTTTATTTGACTTGAGGTGTACGTTCTTTTGCCAAAAGTCTTTAAATTATTAAG
CATGAAAGGAATATATATTAGTTTAAATTATTAACATGTATAATAATATAAATTTTAACAATATGTGCTATAC
AAATAATGACATGAATTTTATGTCATTTTGTCATTTATTTCATGTTATTTGAGTCTATTTTAAAAAGAAGCGG
TGATTTGTAGGTTGTTAGACTATAATTTCATACAAACATCATATATGGTCATTTTCTTTTCAGTTTTTTTTTT
TTGTTTATCTAAACTTATCTGAACATATTAGTCGTAATCTGAACTTACTAGATCATACTCCCTCCCTCATTTA
TTAGTTTACCCCTTTCTTTTGCACAAAGTTGAGTTCACTCATGGTACACCCGGGTGTACCATGAGCATGGTAC
ACCTAGGGGTATTTTTGTAATAAATTAAAACTCCAGGTGCTCATTTGACACCAAATTTGCTCATATGGGTATT
TTGCTTATTTACTCATTTGCTCATTTCGCTCATTTTGTAATAAAAGTGTTTGCTCATTTTCTTAAATGAGTAG
AAATGTTTGCTCATTTGTTCATTTTGCTCATTCGCTCATTCGTTCATTCGCTCATTTTGCTCATTTGGAGAAA
ATAGTCTTGGGTGTACCATGAGCATGGTACACCCAGGTGTACTATAAATATTCTCCACAAAGTTTTAGGAGAA
ATAATGTTGTGGGTAATAGAAAGAAAGATAAACTAGTATTTTATATTAAAATTGAATGGATGTGTGATGAAAA
GTTTGTGGTCCCATTTTAAAAGTAGCAAAAAAAAAGAAAGGGGTAAACTAATAAGAGACACCCTTAAAAGAAA
TACGGGTAAACTAATAAAGGGTAGAGGGAGTACTTTGTAAGTGCCAAAAACCTTATTTTTACTGAACTAAACT
TAAATTTTTTTTTACAACAACTGAAAATAAGATGAATGGAAATTACAGAATAGACGCTTATGTCTCATAATTT
TACTAAATTAGCCAGACAAATAGTACGGTAGACGAATAGACGATAGACCTAACGTTTTTTGTTTTTAACATGC
TGCTTTATACTGTACTTATACTACTTACAATTTACAGCTATCCATCACAGCCATATTCATTCTTGTGACTTAT
TTTCACGTTGGAAATTCTCAAATTCTATATAAGAAGTCATACTACTCTTCCATTTACTTATTCATTGGTCATT
TTAAAGTCCATTTCTTTTCTTTCATACTCGTTACTCCATTATGGGGGCATATGCATATCCATATGTTGTTAGT
CTTTTGGTGGTTTTATACTTTGCTACCTTTTTCACTTCTTTAGCATGCCCTTATTGTCCTTACCTTCCTCCTC
CCCCTAAACCACCTACTTCAGCTTGTCCTCCCCCTGAACACCCTCCACACACCAAGCCACCACCTCATACGCC
GTCCGGAAAACCGCCTTCCTATCCTCATAAACCACCTCATTCTCCTCATAAACCACCTCAACATTCTCCTCAT
AAACCACCTCAACATTCTCCCTATAAACCACCTTATCACACCCCACCATATACTCCTAAACCGTCGCCGCCCT
TTGTTACTCCACCTTACACACCACCACCACATGGTTTTAAACCTCCTATTGTCACTCCACCATACACTCCACC
TTATTCTCCAAAACCACCACATCATCATCCCTACAAACCACCTCATTACACCCCTAAACCTCCTAAAGTAGCA

CCACCATATACTCCTAAACCTACACCACACACCCCACCACCACATGGCTTTAAACCTCCTATTGTCACTCCAC
CATACACTCCACCTTATTCTCCAAAACCACCACATCATCATCCCTACAAACCACCTCATTACACACCTAAACC
TCCTAAAGTATCACCACCATATACTCCTAAACCTACACCACACACCCCACCACCACATGGCTTTAAACCTCCT
ATTGTCACCCCACCATACACTCCACCCTACACACCCAAACCTCCTATTGTTAGCCCACCCTATACACCAAAAC
CTCCTGTTACTCCACCCTACACACCTAAACCTCCTATTGTCAGCCCACCATATAGCCCACCCTACACACCTAA
ACCTCCCATTGTGAGCCCACCTTACACACCAAAACCCCCTATTGTGAGCCCACCCTACACACCTAAACCCCCC
ATTGTGAGCCCACCCTACACACCAAAACCTCCTGTTGTGAGCCCACCATATAGCCCGCCCTACACACCAAAAC
CTCCTGTTGTGAGCCCACCATATACCCCACCCTACACACCAAAACCTCCTGTTGTGAGCCCACCATATAGCCC
ACCCTACACACCAAAACCTCCTGTTGTAAGCCCACCATATAGTCCACCCTACACACCAAAACCTCCTGTTGTG
AGCCCACCATATAGCCCACCCTACACACCAAAACCTCCTGTTGTAAGCCCACCATATAGTCCACCCTACACAC
CAAAACCCCCAGTTGTAAGCCCACCATATAGTCCACCCTACACACCAAAACCCCCAGTTGTAAGCCCACCATA
TAGTCCACCCTACACACCAAAACCTCCTGTTGTGAGCCCACCATACACACCAACACCCCCAATAGTAAGCCCA
CCAGTAGTAAGTCCACCATACACACCAACACCTCCTATTATAACTCCTAGCCCACCAAGTCCCGTACCGAGTC
CCGAAACACCTTGTCCACCACCACCACCACCAGTACCATGCCCACCTCCTTCGACACCGGTTCAACCAACATG
CTCCATTGATACGTTAAAGTTGAATGCATGTGTTGATGTACTAGGAGGGCTCATACATATTGGTATAGGAAGT
GGTGCTAAAGGTGCATGTTGTCCAATCTTAGGTGGTCTTGTAGGGTTAGATGCTGCTGTTTGTCTTTGCACTA
CTATCAGAGCTAAACTTCTAAACATCAACATTATTCTTCCTCTTGCTCTTCAAGTTTTGGCTGATTGTGGCAA
ATCTCCTCCTCCTGGTTTCCAATGTCCTTCTTCTTATTAGTTCTTTTACTCTTAGGTTGTGATATCAATATGT
GTGTAATTGTATGTGTGTCTTTTTAAGTGCTAAAAAAAGTTCACTGAATAAACACAACTTATTTTAGCAACTA
GTATGGTGACTTTGTGAGTACATGTATTTACATTTTTAGAAAGTATTCCTTTGGGTCCATGAACCTAACTAGT
GATGAGTAGTGACTACTGACTAGTGACTATTTCTATTCTATGAGCTTTGTACTTGTATGTTTCACGTGAGAGA
ATCTACCAATTGTATGTCAAGAGTTGTATGATTGACTTGCCAATGTGATATTGTTAGGCTTTATATATCTTGC
CTAGTTGCCTACATGTATGTAGTATGATTGAACTCATCAACATATGAGTAAGTGTGGAAATGTATTGTGGAAG
TTCAAAAACATCACATGAGCTTGCACTCCTCAATCTATCAATTGTCACTCTCTTTCAGTGGATAGTACTCCCA
CTATATATATGATTCTCTTTAGCGCAGCTAGTCCTTACATGTTATGAGGATGTTGAATTTCATGACGTTATTA
TACTTCCTTCATTTTTTATTAATTAGCTATAGGTTACAAAACATTGTTATGAGATCAAACTTTATAAATAAAC
ACTGGTCATTTTTTCGCTAAAAATATCTAATGAAAATTGTAAAAATATACCTTAGAACATAGTACACTTTTAA
TTACTAGAAAAATACTAAAAATGATACTATTCTAACAAAATACCTATTAATTAAGTTGTTTTGTGTATTCATA
TCGATGTCTCTTTTATTTTGTTGTAAAATTATGGTTTATGAATTATGAGCTTATCAACTACATAAATACTCGT
TAGCACTTCCTACTTTATAGTTTCTCTTGAAAGTATTCTCACTACTAGTTATTCTCTAACCTTTTCCCCATCT
TCATAGGCTGAAGGTCGTATTTCGTGGGACATCATATATTTATATTGGTAGATAACTATTGTCATATATTTAG
ATCAAGTAATAAGAGCACTAGATTTAAGACTTGTAGTCAAGAGTCTAGCTAGAATTTGGTTGTTGCATGTCTT
GATACTCAAATTCACCTCATCTATCTTGCTTGAGTATCCCTTGTTATTTAAAGTTGTTACTTTATTATGACAA
ACATGCCAGCTAATTTCGGATAGAGGGAGTGACATTTTATCTGATCTGTTAGTAGATCTCATCTACATTCTAC
ATACTCTCAAGTCACTTCCTTGTGGGGGAACAAAAAACAAGAAAAGAGGAGAGAGGGTGAAAATGAACATAGT
ACATGCCCGGAATTACTAGATATGGGAAGGTTGGATAATTAAATGGGATGAATAAATGGGTAAGTCGCGTAAG
GTAAAAAGGTAAATTTGGTCTAAAAAAGTTAACTATACTAGCAAATCGAGCAATTAAAAGAACACCGACCCAT
AAAAGAAAAATGATGTGATTAAAAAATGGTGTGACAGATGCAATTAACGAAACAAAAACAATGAATAATCAAC
ACAGGAATTAACGTGGTTCACTAACAATATGTTAGTTAAGTCCACATGCAGAGGAGAAACAGTTTTATTAGGT
TTAGGAGAAGAGTATATATTACAGAACAATATACAGAGGCATAGGTTATTAATATGGTTTATATGGTACCCTA
CTAATTAACCTTAACCAATGGACATAAACACAAAACTGAAACGAACCCGATATATTAAACAACATATACTGAA
AAGTTCGGGAACATTGCCTCCTAACCCTCACCATGACGTTTCACCTCTAGACCCCAACTAGCTGACCGGACAG
TGAGATTCCGTGTTAGGTTATTTCTAACGTCAGATAAATAATTCAAAAGTATAATTAAAAAAAAATAGTATGT
ACATCAGTACATACTACATAGCATAAATTAATCGACATGGACGCAGAAGTCAAAAAGTAAGGTAATACTTCCT
CCGTTTCTTTTTACTTGCAACGGTTTGACTTTTACACTATTCACCAACTCTACTTAGTTTATGTATGGTGATT
TATTGTTAAGGAAATACATATTCATGTGGGATCTTGTTAGATTCGTCTGAATGTATATTTTCTGAATATATAC
TTTTTATAATTTTTATTTATCGGTAATTAAAGTTATTGATGGTTGAAATTATGCATTGGCAAGCGTGAAAGTC
AAACTGTTGCAAGTAAAAAGAAACAGAGGAAGTATGAACAACAATAGTAATTCCCATAGTTGAGTTTTGAACA
TGACCAACAATGTTTTGTTTTTTAAAAATATCTACCCTAGAACTCAGAAGGCTAGACTGTTAGCAAATTCAAC
CCATCTCTTTTTTTTTT

Appendix C

8.1 TandemRepeats.java

```
import java.io.*;

public class TandemRepeats
{
    private File file = new File("sequences.txt");                  //List of sequences
    private Sequence[] sequence = new Sequence[100];                //Will contain the sequences
    private int seqNum, totalSeqs;                                  //Current sequence number and
                                                                            Total number of sequences

    public static void main (String args[])                         //main function
    {
        TandemRepeats tr = new TandemRepeats();                     //New instance of the class
        tr.start();                                                         //Begin the program
    }

    public void start()
    {
        if (!readFile())                                                    //If the file has an error post
                                                                            a message and exit
        {
            System.out.println("Error Reading 'sequences.txt' at sequence number "
                + (seqNum + 1) + ".  Exitting.");
            System.exit(0);
        }
        totalSeqs = seqNum;                                                 //Set the value for total number
                                                                            of sequences
        System.out.println("Total number of sequences loaded: "
            + totalSeqs);                                                   //Display the number of
                                                                            sequences loaded successfully
        System.out.println("Performing analysis (each '.'" +
            "represents a protein analysed)");                          //Display description of next
                                                                            events

        for (int i = 0; i < totalSeqs; i++)                             //Cycle through all the
                                                                            sequences
        {
            sequence[i].analyse();                                          //Analyse the sequence's repeats
            System.out.print(".");                                      //Display a '.' to indicate
                                                                            completion
        }

        System.out.print("\n\n");                                       //Two line breaks
```

```
        System.out.println("All sequences analysed.  Putting results to file.");
                                                    //Display information about
                                                                        next stage

        File file = new File("Repeats.txt");                    //Attempt to open the results
                                                                        file
        if (file.exists()) file.delete();                   //If it exists, delete it
        for (int i = 0; i < totalSeqs; i++)                     //Go through the sequences
        {
            sequence[i].writeResults();                             //Write to the file
            System.out.print(".");                                  //Display a '.' to indicate
                                                                        completion
        }

        System.out.print("\n\n");                                   //Two line breaks
        System.out.println("Information output to file 'Repeats.txt'");
                                                    //Display information about the
                                                                        completion
    }

    public boolean readFile ()                                      //Method for reading sequences
                                                                        in to the program
    {
        try
        {
            BufferedReader in = new BufferedReader(new FileReader(file));
            String input = in.readLine();                           //Read a line from the file
            seqNum = 0;                                                 //Set the number of sequences to
                                                                        0
            while (input != null)                                   //While there is more to read
                                                                        from the file
            {
                String seqName;
                String seq;
                if (input.startsWith(">"))                          //Check for a heading line
                                                                        (FASTA format)
                {
                    seqName = input.substring(1);                   //Enter everything on the header
                                                                        line except the ">"
                    input = in.readLine();                              //Get the next line from the
                                                                        file
                    seq = input.trim();                                 //Get the sequence from the file
                                                                        and trim spaces from it
                }
                else                                                    //Sequence doesn't have a
                                                                        heading
                {
                    seqName = "Unnamed sequence number "
                        + seqNum;                                       //Give a generic heading for the
                                                                        sequence
                    seq = input.trim();                                 //Get the sequence from the file
```

```
                                                                    and trim spaces from it
                }

                sequence[seqNum] = new Sequence (seqName, seq);   //Set up a new sequence with the
                                                                    collected information

                input = in.readLine();                            //Open the next line of the file
                seqNum++;                                         //Add one to the number of the
                                                                    sequence
            }
            in.close();                                           //Close the file
            return true;                                          //Return that the read worked
                                                                    fine
        }
        catch (IOException e)
        {
            return false;                                         //Return false if there was any
                                                                    error at all
        }
    }
}
```

8.2 Sequence.java

```
import java.util.*;                                    //include utilities
import java.io.*;                                      //include input/output tools

public class Sequence                                  //Class name
{
    private String name, code;                         //variables for the name of the
                                                           sequence and the code
    Vector repeatStore = new Vector();                 //vector for holding the repeats
                                                           as they are found


    public Sequence (String argName, String argCode)   //constructor
    {
        name = argName;                                    //keep the name within the
                                                               instance
        code = argCode;                                //keep the code within the
                                                               instance
    }

    public void analyse ()                             //method for repeat analysis
    {
        Repeat r = new Repeat();                       //new variable containing a
                                                               Repeat class
        int curPos = 0;                                //initiate the currentPosition
                                                               variable
        int repeats = 0;                               //initiate the number of repeats
                                                               found at this position
        String pattern = "";                           //set the search pattern to an
                                                               empty string
        while (curPos < code.length() - 50)            //Whilst the current position is
                                                               at least 50 from the end of
                                                               the sequence

        {
            int patternSize = 3;                       //set the pattern size to 3
            boolean match = false;                     //set the boolean match to false
            while (patternSize < 25 && !match)         //whilst the pattern size is
                                                               less than 25 and there is no
                                                               found match

            {
                if (repeats == 0) pattern = code.substring
                    (curPos, curPos + patternSize);    //if there are no repeats at
                                                               this location, set the pattern
                                                               to be the next part of the
                                                               sequence

                PatternMangler pm =
                    new PatternMangler(pattern);       //Make a new patternMangler for
                                                               looking at insertions/
                                                               deletions/substitutions

                String newPattern;                     //Declare a new string for the
```

```
                                                    pattern to compare with

int clipLength = patternSize * 2;                   //Set the length of the
                                                    newPattern
int patternStart, patternEnd;                       //Declare new varibles to pick
                                                    out the newPattern
if (repeats == 0)
    patternStart = curPos + patternSize;            //If there are no repeats at
                                                    this position set the pattern
                                                    start accordingly
else patternStart = curPos;                         //otherwise set the current
                                                    position to be the beginning
patternEnd = patternStart + clipLength;             //Set the end of the pattern
                                                    within the code
if (patternEnd > code.length())
    patternEnd = code.length();                     //If the end of the pattern is
                                                    longer than the code sequence,
                                                    move it back

newPattern = pm.findSimilarity(code.substring
    (patternStart, patternEnd));                    //Try to find similarity using
                                                    the patternMangler and return
                                                    it
match = false;                                      //Set the match boolean to false
if (newPattern != "" && newPattern != null)         //If there is similarity
{
    repeats++;                                      //Increase the number of repeats
                                                    found at this location
    match = true;                                   //Set the match to be true
}

if (!match)                                         //If there has been no match
{
    if (repeats > 0)                                //and if the number of repeats
                                                    was more than 0
    {
        r = new Repeat();                           //Create a new repeat container
        repeatStore.add(r);                         //add it to the vector
    }
    repeats = 0;                                    //set the number of repeats to
                                                    zero
}


else if (match && !newPattern.equals(""))           //any match found
{
    if (repeats == 1)                               //first match found
    {
        r.addWord(pattern);                         //add the initial pattern to the
                                                    repeat object
        r.setBRI(curPos);                           //set the Beginning Residue
```

```
                                                                    Index in the repeat object
                    curPos += patternSize;                          //set the new current position
                                                                    to the end of the first repeat
                }
                curPos += newPattern.length();                      //add the length of the
                                                                    newPattern to the current
                                                                    position
                r.addWord(newPattern);                              //add the new repeat to the
                                                                    repeat object
            }

            if (!match)                                             //no match found
            {
                patternSize++;                                      //increase the size of the
                                                                    pattern being searched
            }
        }
        curPos++;                                                   //increase the current position
                                                                    of the search
    }
    if (repeatStore.size() > 0) repeatStore.removeElementAt(repeatStore.size() - 1);
                                                                    //if any repeats have been
                                                                    found, delete the final
                                                                    redundent one
}

public void writeResults()                                          //write the results to a file
{
    try
    {
        PrintWriter out = new PrintWriter(new FileWriter("Repeats.txt", true));
                                                                    //make a new file writer with
                                                                    the Repeats.txt file
        out.println("Sequence ID:    " + name);                  //Write the sequence ID to the
                                                                    file
        out.println();
        out.println("Sequence Code:");                              //Write the title for the
                                                                    Sequence Code
        for (int x = 0; x < code.length(); x += 60)
        {
            for (int y = 0; y < 60; y += 10)
            {
                if (x + y + 10 < code.length())      out.print(code.substring
                    (x + y, x + y + 10));
                else
                {
                    out.print(code.substring(x + y, code.length()));
                    for (int i = 0; i < 60 - (code.length() % 60) + (6 - y / 10); i++)
                        out.print(" ");
                    break;
                }
```

```
                    out.print(" ");
                }
                out.println(x + 60);
            }                                                    //All the above sets out a nice
                                                                   layout for the sequence code
            out.println();
            out.println();
            Enumeration store = repeatStore.elements();          //Starts an enumeration of the
                                                                   repeats
            int repeatCount = 0;
            while (store.hasMoreElements())                      //Whilst there are more repeats
                                                                   to add to the file
            {
                Repeat repeat = (Repeat)store.nextElement();
                if (repeat.getNoOfWords() > 0)
                {
                    repeatCount++;                               //Add one to the number of
                                                                   repeats found
                    out.println("Identified Repeat # " + repeatCount);
                    out.println("----------------------");
                    out.println();
                    int BRI = repeat.getBRI();                   //Get the location of the repeat
                    int noOfWords = repeat.getNoOfWords();       //Get the number of repeats at
                                                                   this location
                    out.println("Starting at residue " + BRI);   //Write the location of the
                                                                   repeats
                    for (int i = 0; i < noOfWords; i++)          //Cycle through the found
                                                                   repeats
                    {
                        out.println("Repetition " + (i + 1) + ":  " + repeat.getWord(i));
                                                                 //Print the sequence of the
                                                                   repeat
                    }
                    out.println();
                    out.println();
                }
            }
            for (int x = 0; x < 5; x++) out.println();           //Leave a five line gap after
                                                                   all entries have been added
            out.close();
        }
        catch (IOException e)
        {
            System.out.println("FAILED TO WRITE TO FILE, UNKNOWN REASONS");
                                                                 //If there is a problem during
                                                                   the process, abort the program
            System.exit(0);
        }
    }
}
```

8.3 Repeat.java

```
import java.util.*;

public class Repeat                                    //Storage class for the repeats
{
    private Vector word = new Vector();                //Vector to hold the patterns
    int beginResidueIndex;                             //Beginning Residue Index

    public void addWord(String argWord)                //add a repeat pattern
    {
        word.add(argWord);                             //add it to the vector
    }

    public void setBRI(int argBRI)                     //set the Beginning Residue
                                                               Index
    {
        beginResidueIndex = argBRI;                    //store it within this instance
    }

    public int getBRI()                                //return the Beginning Residue
                                                               Index
    {
        return beginResidueIndex;                      //return it
    }

    public int getNoOfWords()                          //return the number of patterns
                                                               in the repeat
    {
        return word.size();                            //return the number
    }

    public String getWord(int index)                   //return a specific pattern in
                                                               the repeat
    {
        return (String)word.get(index);                //return it
    }
}
```

8.4 PatternMangler.java

```
public class PatternMangler                                    //PatternMangler class used for
                                                                    finding near matches
{
    String pattern;

    public PatternMangler (String argPattern)                  //Constructor
    {
        pattern = argPattern;                                  //Store the pattern for
                                                                    comparison within the instance
    }

    public String findSimilarity(String argPattern)            //Find similarity to the pattern
                                                                    supplied as an arguement
    {
        int[] result = new int[pattern.length()];              //Set up an array to contain
                                                                    individual characters within
                                                                    the pattern
        result[0] = argPattern.indexOf(pattern.charAt(0));     //Set the first item to the
                                                                    position within argPattern
                                                                    that the first character of
                                                                    Pattern can be found
        if (result[0] != 0) return "";                         //If this is not zero, the
                                                                    patterns are not tandem
        for (int chars = 1; chars < pattern.length();
            chars++)                                           //Cycle for the length of the
                                                                    pattern
        {
            result[chars] = argPattern.indexOf(pattern.charAt(chars), result[chars - 1] + 1);
                                                   //For each, find the position of
                                                                    the exactly equal residue in
                                                                    the pattern
        }
        int replacements = 0;                                  //Specify the number of
                                                                    replacements made = 0
        for (int chars = 1; chars < pattern.length() - 1;
            chars++)                                           //Cycle through the pattern
                                                                    length again
        {

            if (result[chars] == -1 ||
                result[chars] > (result[chars - 1] + 1))       //If there is no matching
                                                                    residue or it isn't
                                                                    immediately after the last
                                                                    match
            {
                char newChar;
                switch (pattern.charAt(chars))                 //switch for the observed
                                                                    character
```

```
            {
                case 'C': case 'G': case 'P':                    //no preferential substitutions
                                                                    available for these residues
                    return null;                                 //return with a failed match
                case 'E': case 'I': case 'K': case 'L': case 'M':
                case 'N': case 'Q': case 'S': case 'V': case 'Y':
                                                                 //three preferential
                                                                    substitutions available
                    newChar = getSubstitute(pattern.charAt(chars), 2);
                                                                 //get the third possible
                                                                    substitution
                    result[chars] = argPattern.indexOf(newChar, result[chars - 1] + 1);
                                                                 //look for it in place of the
                                                                    previous search
                    if (result[chars] != -1 && result[chars] < result[chars + 1] &&
                        result[chars] <= (result[chars - 1] + 1)) break;
                                                                 //if a suitable match is made,
                                                                    exit the switch
                case 'D': case 'F': case 'H': case 'R': case 'W':
                                                                 //two preferential substitutions
                                                                    available
                    newChar = getSubstitute(pattern.charAt(chars), 1);
                                                                 //get the second possible
                                                                    substitution
                    result[chars] = argPattern.indexOf(newChar, result[chars - 1] + 1);
                                                                 //look for it in place of the
                                                                    previous search
                    if (result[chars] != -1 && result[chars] < result[chars + 1] &&
                        result[chars] <= (result[chars - 1] + 1)) break;
                                                                 //if a suitable match is made,
                                                                    exit the switch
                case 'A': case 'T':                              //one preferential substitutions
                                                                    available
                    newChar = getSubstitute(pattern.charAt(chars), 0);
                                                                 //get the first possible
                                                                    substitution
                    result[chars] = argPattern.indexOf(newChar, result[chars - 1] + 1);
                                                                 //look for it in place of the
                                                                    previous search
                    if (result[chars] == -1 || result[chars] >= result[chars + 1] ||
                        result[chars] > (result[chars - 1] + 1)) return "";
                                                                 //if there is still no match,
                                                                    then the repeat isn't real
                    break;
            }
            replacements++;                                      //if a replacement is made, keep
                                                                    track of it
        }
    }
    if (replacements > pattern.length() * 0.25)
        return "";                                               //if there were more than 25%
```

```
                                                                    replacements, the match isn't
                                                                    valid
    return argPattern.substring(0, result[pattern.length() - 1] + 1);
                                                                //return the pattern which
                                                                    expresses a match
}

private char getSubstitute
    (char argOriginal, int argSubNum)                       //routine for getting
                                                                    substitutes above
{
    switch (argOriginal)                                        //switch for the original
                                                                    residue
    {
        case 'A': return 'S';                                   //A returns S
        case 'D':
            if (argSubNum == 1) return 'E';                     //D returns E on first
                                                                    substitution and
            else if (argSubNum == 0) return 'N';            //N on the second substitution
        case 'E':
            if (argSubNum == 2) return 'N';                     //E returns N on the 1st
            else if (argSubNum == 1) return 'Q';            //Q on the 2nd
            else if (argSubNum == 0) return 'K';            //and K on the 3rd substitution
        case 'F':
            if (argSubNum == 1) return 'Y';                     //etc.
            else if (argSubNum == 0) return 'W';
        case 'H':
            if (argSubNum == 1) return 'Y';                     //etc.
            else if (argSubNum == 0) return 'N';
        case 'I':
            if (argSubNum == 2) return 'V';                     //etc.
            else if (argSubNum == 1) return 'L';
            else if (argSubNum == 0) return 'M';
        case 'K':
            if (argSubNum == 2) return 'R';                     //etc.
            else if (argSubNum == 1) return 'E';
            else if (argSubNum == 0) return 'Q';
        case 'L':
            if (argSubNum == 2) return 'I';                     //etc.
            else if (argSubNum == 1) return 'M';
            else if (argSubNum == 0) return 'V';
        case 'M':
            if (argSubNum == 2) return 'L';                     //etc.
            else if (argSubNum == 1) return 'I';
            else if (argSubNum == 0) return 'V';
        case 'N':
            if (argSubNum == 2) return 'S';                     //etc.
            else if (argSubNum == 1) return 'H';
            else if (argSubNum == 0) return 'D';
        case 'Q':
            if (argSubNum == 2) return 'E';                     //etc.
```

```
                else if (argSubNum == 1) return 'R';
                else if (argSubNum == 0) return 'K';
            case 'R':
                if (argSubNum == 1) return 'K';                    //etc.
                else if (argSubNum == 0) return 'Q';
            case 'S':
                if (argSubNum == 2) return 'A';                    //etc.
                else if (argSubNum == 1) return 'N';
                else if (argSubNum == 0) return 'T';
            case 'T':
                return 'S';                                                //etc.
            case 'V':
                if (argSubNum == 2) return 'I';                    //etc.
                else if (argSubNum == 1) return 'M';
                else if (argSubNum == 0) return 'L';
            case 'W':
                if (argSubNum == 1) return 'Y';                    //etc.
                else if (argSubNum == 0) return 'F';
            case 'Y':
                if (argSubNum == 2) return 'F';                    //etc.
                else if (argSubNum == 1) return 'W';
                else if (argSubNum == 0) return 'H';
        }
        return 'Z';                                                        //never happens under normal
                                                                                circumstances but will not
                                                                                compile without this line
    }
}
```

Appendix D

9.1 GenScan Output

```
Gn.Ex Type S .Begin ...End .Len Fr Ph I/Ac Do/T CodRg P.... Tscr..
----- ---- - ------ ------ ---- -- -- ---- ---- ----- ----- ------
 1.01 Sngl +   3253   5004 1752  0  0   66   32  1084 0.915 100.06
 1.02 PlyA +   5084   5089    6                                1.05

Predicted peptide sequence(s):

>09:20:55|GENSCAN_predicted_peptide_1|583_aa
MGAYAYPYVVSLLVVLYFATFFTSLACPYCPYLPPPPKPPTSACPPPEHPPHTKPPPHTP
SGKPPSYPHKPPHSPHKPPQHSPHKPPQHSPYKPPYHTPPYTPKPSPPFVTPPYTPPPHG
FKPPIVTPPYTPPYSPKPPHHHPYKPPHYTPKPPKVAPPYTPKPTPHTPPPHGFKPPIVT
PPYTPPYSPKPPHHHPYKPPHYTPKPPKVSPPYTPKPTPHTPPPHGFKPPIVTPPYTPPY
TPKPPIVSPPYTPKPPVTPPYTPKPPIVSPPYSPPYTPKPPIVSPPYTPKPPIVSPPYTP
KPPIVSPPYTPKPPVVSPPYSPPYTPKPPVVSPPYTPPYTPKPPVVSPPYSPPYTPKPPV
VSPPYSPPYTPKPPVVSPPYSPPYTPKPPVVSPPYSPPYTPKPPVVSPPYSPPYTPKPPV
VSPPYSPPYTPKPPVVSPPYTPTPPIVSPPVVSPPYTPTPPIITPSPPSPVPSPETPCPP
PPPPVPCPPPSTPVQPTCSIDTLKLNACVDVLGGLIHIGIGSGAKGACCPILGGLVGLDA
AVCLCTTIRAKLLNINIILPLALQVLADCGKSPPPGFQCPSSY

>09:20:55|GENSCAN_predicted_CDS_1|1752_bp
atgggggcatatgcatatccatatgttgttagtcttttggtggttttatactttgctacc
tttttcacttctttagcatgcccttattgtccttaccttcctcctccccctaaaccacct
acttcagcttgtcctcccccctgaacaccctccacacaccaagccaccacctcatacgccg
tccggaaaaccgccttcctatcctcataaaccacctcattctcctcataaaccacctcaa
cattctcctcataaaccacctcaacattctccctataaaccaccttatcacaccccacca
tatactcctaaaccgtcgccgccctttgttactccaccttacacaccaccaccacatggt
tttaaacctcctattgtcactccaccatacactccaccttattctccaaaaccaccacat
catcatccctacaaaccacctcattacacccctaaacctcctaaagtagcaccaccatat
actcctaaacctacaccacacaccccaccaccacatggctttaaacctcctattgtcact
ccaccatacactccaccttattctccaaaaccaccacatcatcatccctacaaaccacct
cattacacacctaaacctcctaaagtatcaccaccatatactcctaaacctacaccacac
accccaccaccacatggctttaaacctcctattgtcaccccaccatacactccaccctac
acacccaaacctcctattgttagcccaccctatacaccaaaacctcctgttactccaccc
tacacacctaaacctcctattgtcagcccaccatatagcccaccctacacacctaaacct
cccattgtgagcccaccttacacaccaaaaccccctattgtgagcccaccctacacacct
aaacccccccattgtgagcccaccctacacaccaaaacctcctgttgtgagcccaccatat
agcccgccctacacaccaaaacctcctgttgtgagcccaccatataccccaccctacaca
ccaaaacctcctgttgtgagcccaccatatagcccaccctacacaccaaaacctcctgtt
gtaagcccaccatatagtccaccctacacaccaaaacctcctgttgtgagcccaccatat
agcccaccctacacaccaaaacctcctgttgtaagcccaccatatagtccaccctacaca
ccaaaaccccccagttgtaagcccaccatatagtccaccctacacaccaaaaccccccagtt
gtaagcccaccatatagtccaccctacacaccaaaacctcctgttgtgagcccaccatac
acaccaacacccccaatagtaagcccaccagtagtaagtccaccatacacaccaacacct
cctattataactcctagcccaccaagtcccgtaccgagtcccgaaacaccttgtccacca
ccaccaccaccagtaccatgcccacctccttcgacaccggttcaaccaacatgctccatt
gatacgttaaagttgaatgcatgtgttgatgtactaggagggctcatacatattggtata
ggaagtggtgctaaaggtgcatgttgtccaatcttaggtggtcttgtagggttagatgct
gctgtttgtctttgcactactatcagagctaaacttctaaacatcaacattattcttcct
cttgctcttcaagttttggctgattgtggcaaatctcctcctcctggtttccaatgtcct
tcttcttattag
```

9.2 Formatted Nucleotide and Peptide Sequence

```
       M  G  A  Y  A  Y  P  Y  V  V  S  L  L  V  V  L  Y  F  A  T
1      atgggggcatatgcatatccatatgttgttagtcttttggtggttttatactttgctacc  60

       F  F  T  S  L  A  C  P  Y  C  P  Y  L  P  P  P  P  K  P  P
61     tttttcacttctttagcatgcccttattgtccttaccttcctcctccccctaaaccacct  120

       T  S  A  C  P  P  P  E  H  P  P  H  T  K  P  P  P  H  T  P
121    acttcagcttgtcctccccctgaacaccctccacacaccaagccaccacctcatacgccg  180

       S  G  K  P  P  S  Y  P  H  K  P  P  H  S  P  H  K  P  P  Q
181    tccggaaaaccgccttcctatcctcataaaccacctcattctcctcataaaccacctcaa  240

       H  S  P  H  K  P  P  Q  H  S  P  Y  K  P  P  Y  H  T  P  P
241    cattctcctcataaaccacctcaacattctccctataaaccaccttatcacaccccacca  300

       Y  T  P  K  P  S  P  P  F  V  T  P  P  Y  T  P  P  P  H  G
301    tatactcctaaaccgtcgccgccctttgttactccaccttacacaccaccaccacatggt  360

       F  K  P  P  I  V  T  P  P  Y  T  P  P  Y  S  P  K  P  P  H
361    tttaaacctcctattgtcactccaccatacactccaccttattctccaaaaccaccacat  420

       H  H  P  Y  K  P  P  H  Y  T  P  K  P  P  K  V  A  P  P  Y
421    catcatccctacaaaccacctcattacacccctaaacctcctaaagtagcaccaccatat  480

       T  P  K  P  T  P  H  T  P  P  P  H  G  F  K  P  P  I  V  T
481    actcctaaacctacaccacacaccccaccaccacatggctttaaacctcctattgtcact  540

       P  P  Y  T  P  P  Y  S  P  K  P  P  H  H  H  P  Y  K  P  P
541    ccaccatacactccaccttattctccaaaaccaccacatcatcatccctacaaaccacct  600

       H  Y  T  P  K  P  P  K  V  S  P  P  Y  T  P  K  P  T  P  H
601    cattacacacctaaacctcctaaagtatcaccaccatatactcctaaacctacaccacac  660

       T  P  P  P  H  G  F  K  P  P  I  V  T  P  P  Y  T  P  P  Y
661    accccaccaccacatggctttaaacctcctattgtcaccccaccatacactccaccctac  720

       T  P  K  P  P  I  V  S  P  P  Y  T  P  K  P  P  V  T  P  P
721    acacccaaacctcctattgttagcccaccctatacaccaaaacctcctgttactccaccc  780

       Y  T  P  K  P  P  I  V  S  P  P  Y  S  P  P  Y  T  P  K  P
781    tacacacctaaacctcctattgtcagcccaccatatagcccaccctacacacctaaacct  840

       P  I  V  S  P  P  Y  T  P  K  P  P  I  V  S  P  P  Y  T  P
841    cccattgtgagcccaccttacacaccaaaaccccctattgtgagcccaccctacacacct  900

       K  P  P  I  V  S  P  P  Y  T  P  K  P  P  V  V  S  P  P  Y
901    aaaccccccattgtgagcccaccctacacaccaaaacctcctgttgtgagcccaccatat  960

       S  P  P  Y  T  P  K  P  P  V  V  S  P  P  Y  T  P  P  Y  T
961    agcccgccctacacaccaaaacctcctgttgtgagcccaccatataccccaccctacaca  1020

       P  K  P  P  V  V  S  P  P  Y  S  P  P  Y  T  P  K  P  P  V
1021   ccaaaacctcctgttgtgagcccaccatatagcccaccctacacaccaaaacctcctgtt  1080

       V  S  P  P  Y  S  P  P  Y  T  P  K  P  P  V  V  S  P  P  Y
1081   gtaagcccaccatatagtccaccctacacaccaaaacctcctgttgtgagcccaccatat  1140
```

```
       S  P  P  Y  T  P  K  P  P  V  V  S  P  P  Y  S  P  P  Y  T
1141   agcccaccctacacaccaaaacctcctgttgtaagcccaccatatagtccaccctacaca  1200

       P  K  P  P  V  V  S  P  P  Y  S  P  P  Y  T  P  K  P  P  V
1201   ccaaaacccccagttgtaagcccaccatatagtccaccctacacaccaaaacccccagtt  1260

       V  S  P  P  Y  S  P  P  Y  T  P  K  P  P  V  V  S  P  P  Y
1261   gtaagcccaccatatagtccaccctacacaccaaaacctcctgttgtgagcccaccatac  1320

       T  P  T  P  P  I  V  S  P  P  V  V  S  P  P  Y  T  P  T  P
1321   acaccaacacccccaatagtaagcccaccagtagtaagtccaccatacacaccaacacct  1380

       P  I  I  T  P  S  P  P  S  P  V  P  S  P  E  T  P  C  P  P
1381   cctattataactcctagcccaccaagtcccgtaccgagtcccgaaacaccttgtccacca  1440

       P  P  P  P  V  P  C  P  P  P  S  T  P  V  Q  P  T  C  S  I
1441   ccaccaccaccagtaccatgcccacctccttcgacaccggttcaaccaacatgctccatt  1500

       D  T  L  K  L  N  A  C  V  D  V  L  G  G  L  I  H  I  G  I
1501   gatacgttaaagttgaatgcatgtgttgatgtactaggagggctcatacatattggtata  1560

       G  S  G  A  K  G  A  C  C  P  I  L  G  G  L  V  G  L  D  A
1561   ggaagtggtgctaaaggtgcatgttgtccaatcttaggtggtcttgtagggttagatgct  1620

       A  V  C  L  C  T  T  I  R  A  K  L  L  N  I  N  I  I  L  P
1621   gctgtttgtctttgcactactatcagagctaaacttctaaacatcaacattattcttcct  1680

       L  A  L  Q  V  L  A  D  C  G  K  S  P  P  P  G  F  Q  C  P
1681   cttgctcttcaagttttggctgattgtggcaaatctcctcctcctggtttccaatgtcct  1740

       S  S  Y  *
1741   tcttcttattag  1752
```

9.3 Codon Usage Statistics

9.3.1 Codon Usage within the *Beta vulgaris* Genome

		U			C			A			G		
		1	2	3	1	2	3	1	2	3	1	2	3
U	U	UUU	F	2.57%	UCU	S	2.07%	UAU	Y	1.87%	UGU	C	0.88%
	C	UUC	F	2.13%	UCC	S	0.94%	UAC	Y	1.58%	UGC	C	0.55%
	A	UUA	L	1.19%	UCA	S	1.59%	UAA	*	0.09%	UGA	*	0.12%
	G	UUG	L	1.97%	UCG	S	0.41%	UAG	*	0.04%	UGG	W	1.54%
C	U	CUU	L	2.50%	CCU	P	2.25%	CAU	H	1.25%	CGU	R	0.86%
	C	CUC	L	1.30%	CCC	P	0.61%	CAC	H	0.88%	CGC	R	0.32%
	A	CUA	L	0.99%	CCA	P	1.96%	CAA	Q	1.73%	CGA	R	0.51%
	G	CUG	L	0.79%	CCG	P	0.42%	CAG	Q	1.23%	CGG	R	0.26%
A	U	AUU	I	2.85%	ACU	T	2.30%	AAU	N	2.66%	AGU	S	1.29%
	C	AUC	I	1.98%	ACC	T	1.07%	AAC	N	1.28%	AGC	S	0.86%
	A	AUA	I	1.21%	ACA	T	1.75%	AAA	K	2.46%	AGA	R	1.51%
	G	AUG	M	2.38%	ACG	T	0.28%	AAG	K	3.11%	AGG	R	1.10%
G	U	GUU	V	2.86%	GCU	A	3.61%	GAU	D	3.68%	GGU	G	3.00%
	C	GUC	V	1.12%	GCC	A	1.31%	GAC	D	1.73%	GGC	G	1.15%
	A	GUA	V	1.23%	GCA	A	2.71%	GAA	E	3.01%	GGA	G	2.78%
	G	GUG	V	1.51%	GCG	A	0.49%	GAG	E	2.66%	GGG	G	1.08%

	First base
	Second base
	Third base
1	Codon
2	Residue
3	Frequency

Residue	Frequency
Leu (L)	8.74%
Ala (A)	8.12%
Gly (G)	8.01%
Ser (S)	7.16%
Val (V)	6.72%
Ile (I)	6.04%
Glu (E)	5.67%
Lys (K)	5.57%
Asp (D)	5.41%
Thr (T)	5.40%
Pro (P)	5.24%
Phe (F)	4.70%
Arg (R)	4.56%
Asn (N)	3.94%
Tyr (Y)	3.45%
Gln (Q)	2.96%
Met (M)	2.38%
His (H)	2.13%
Trp (W)	1.54%
Cys (C)	1.43%
*	0.25%

9.3.2 Codon Usage within the RS1 Gene

		U			C			A			G		
		1	2	3	1	2	3	1	2	3	1	2	3
U	U	UUU	F	1.03%	UCU	S	1.54%	UAU	Y	3.94%	UGU	C	1.54%
	C	UUC	F	0.34%	UCC	S	0.51%	UAC	Y	4.45%	UGC	C	0.68%
	A	UUA	L	0.86%	UCA	S	0.34%	UAA	*	0.00%	UGA	*	0.00%
	G	UUG	L	0.51%	UCG	S	0.34%	UAG	*	0.17%	UGG	W	0.00%
C	U	CUU	L	1.37%	CCU	P	13.18%	CAU	H	3.25%	CGU	R	0.00%
	C	CUC	L	0.17%	CCC	P	5.14%	CAC	H	0.86%	CGC	R	0.00%
	A	CUA	L	0.34%	CCA	P	18.66%	CAA	Q	0.86%	CGA	R	0.00%
	G	CUG	L	0.00%	CCG	P	1.20%	CAG	Q	0.00%	CGG	R	0.00%
A	U	AUU	I	2.23%	ACU	T	2.57%	AAU	N	0.17%	AGU	S	1.54%
	C	AUC	I	0.51%	ACC	T	1.37%	AAC	N	0.34%	AGC	S	3.42%
	A	AUA	I	0.68%	ACA	T	4.45%	AAA	K	6.51%	AGA	R	0.17%
	G	AUG	M	0.17%	ACG	T	0.34%	AAG	K	0.34%	AGG	R	0.00%
G	U	GUU	V	3.25%	GCU	A	1.37%	GAU	D	0.68%	GGU	G	1.20%
	C	GUC	V	0.68%	GCC	A	0.00%	GAC	D	0.00%	GGC	G	0.51%
	A	GUA	V	2.23%	GCA	A	1.03%	GAA	E	0.34%	GGA	G	0.51%
	G	GUG	V	1.54%	GCG	A	0.00%	GAG	E	0.00%	GGG	G	0.51%

First base
Second base
Third base
1 Codon
2 Residue
3 Frequency

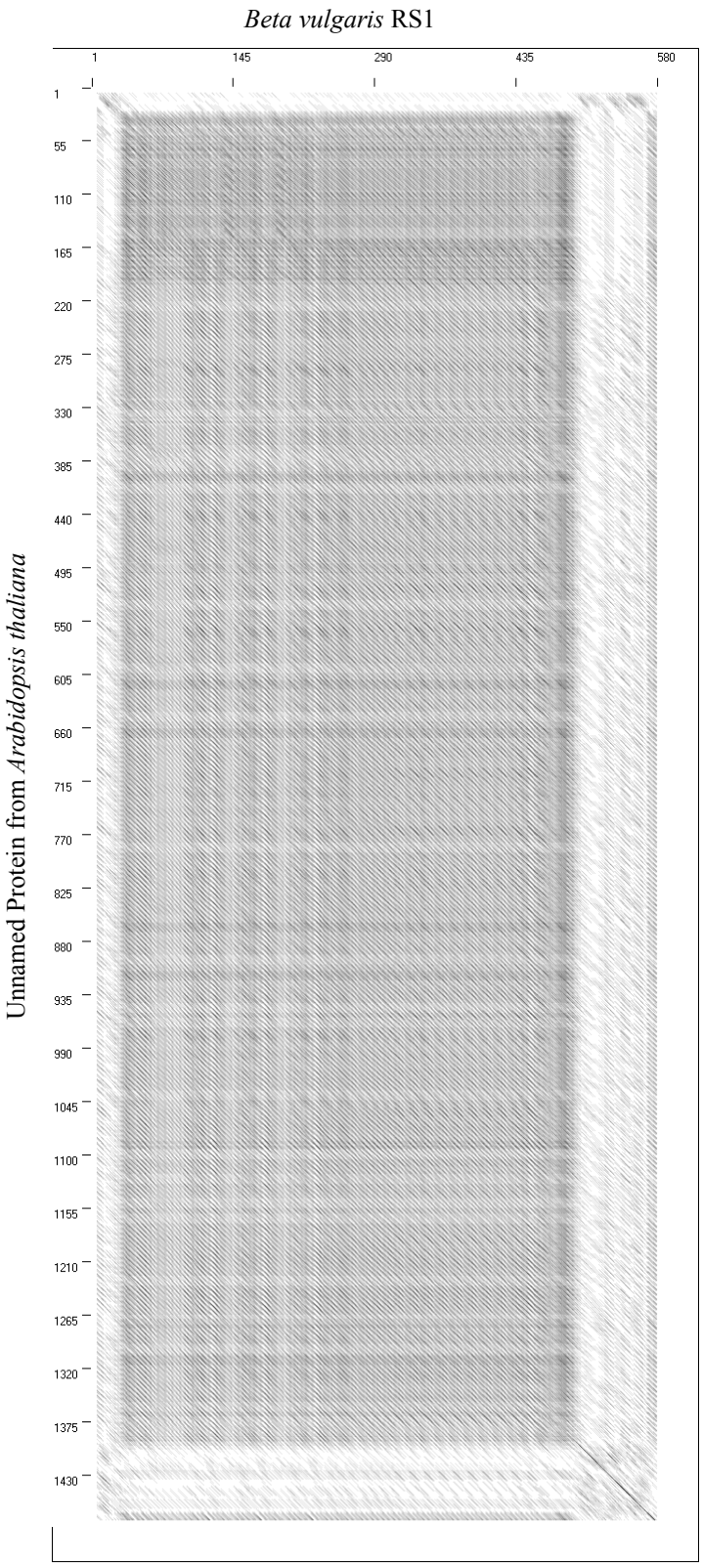
Residue	Frequency
Pro (P)	38.18%
Thr (T)	8.73%
Tyr (Y)	8.39%
Val (V)	7.70%
Ser (S)	7.69%
Lys (K)	6.85%
His (H)	4.11%
Ile (I)	3.42%
Leu (L)	3.25%
Gly (G)	2.73%
Ala (A)	2.40%
Cys (C)	2.22%
Phe (F)	1.37%
Gln (Q)	0.86%
Asp (D)	0.68%
Asn (N)	0.51%
Glu (E)	0.34%
*	0.17%
Arg (R)	0.17%
Met (M)	0.17%
Trp (W)	0.00%

9.3.3 A Comparison of the Abundance of Individual Amino Acids

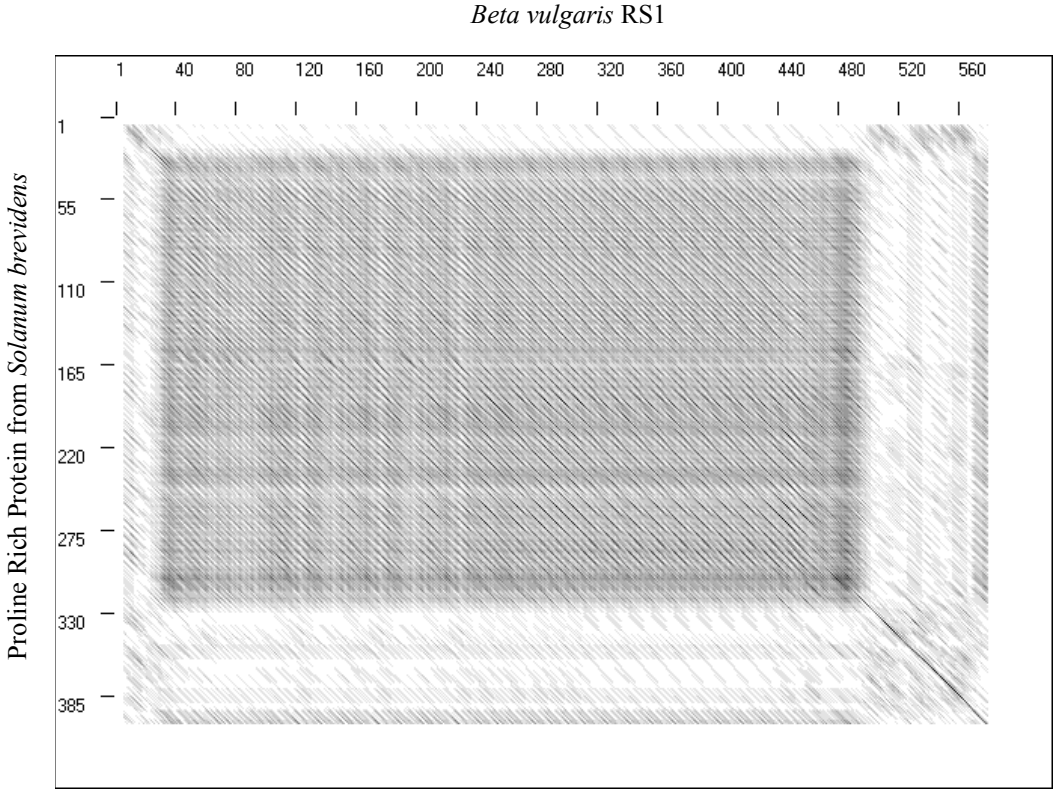
Residue	Beta Vulgaris	RS1	Ratio Difference
Pro (P)	5.24%	38.18%	7.29
Tyr (Y)	3.45%	8.39%	2.43
His (H)	2.13%	4.11%	1.93
Thr (T)	5.40%	8.73%	1.62
Cys (C)	1.43%	2.22%	1.55
Lys (K)	5.57%	6.85%	1.23
Val (V)	6.72%	7.70%	1.15
Ser (S)	7.16%	7.69%	1.07
*	0.25%	0.17%	0.68
Ile (I)	6.04%	3.42%	0.57
Leu (L)	8.74%	3.25%	0.37
Gly (G)	8.01%	2.73%	0.34
Ala (A)	8.12%	2.40%	0.30
Phe (F)	4.70%	1.37%	0.29
Gln (Q)	2.96%	0.86%	0.29
Asn (N)	3.94%	0.51%	0.13
Asp (D)	5.41%	0.68%	0.13
Met (M)	2.38%	0.17%	0.07
Glu (E)	5.67%	0.34%	0.06
Arg (R)	4.56%	0.17%	0.04
Trp (W)	1.54%	0.00%	0.00

9.4 Dotplot Analyses

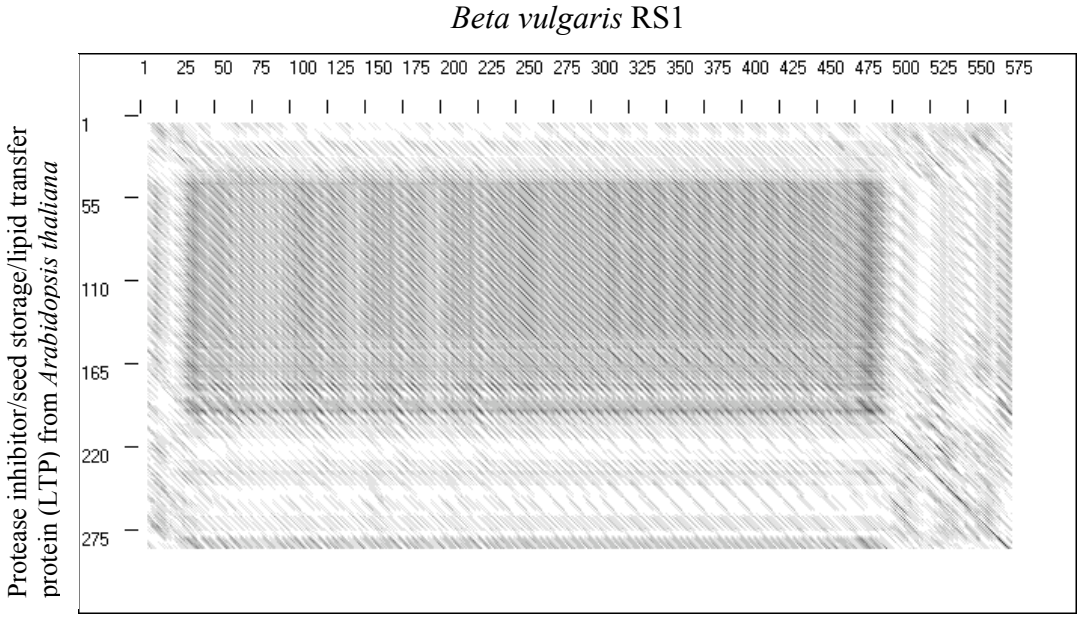
9.4.1 *Beta vulgaris* RS1 versus Unnamed Protein from *Arabidopsis thaliana*



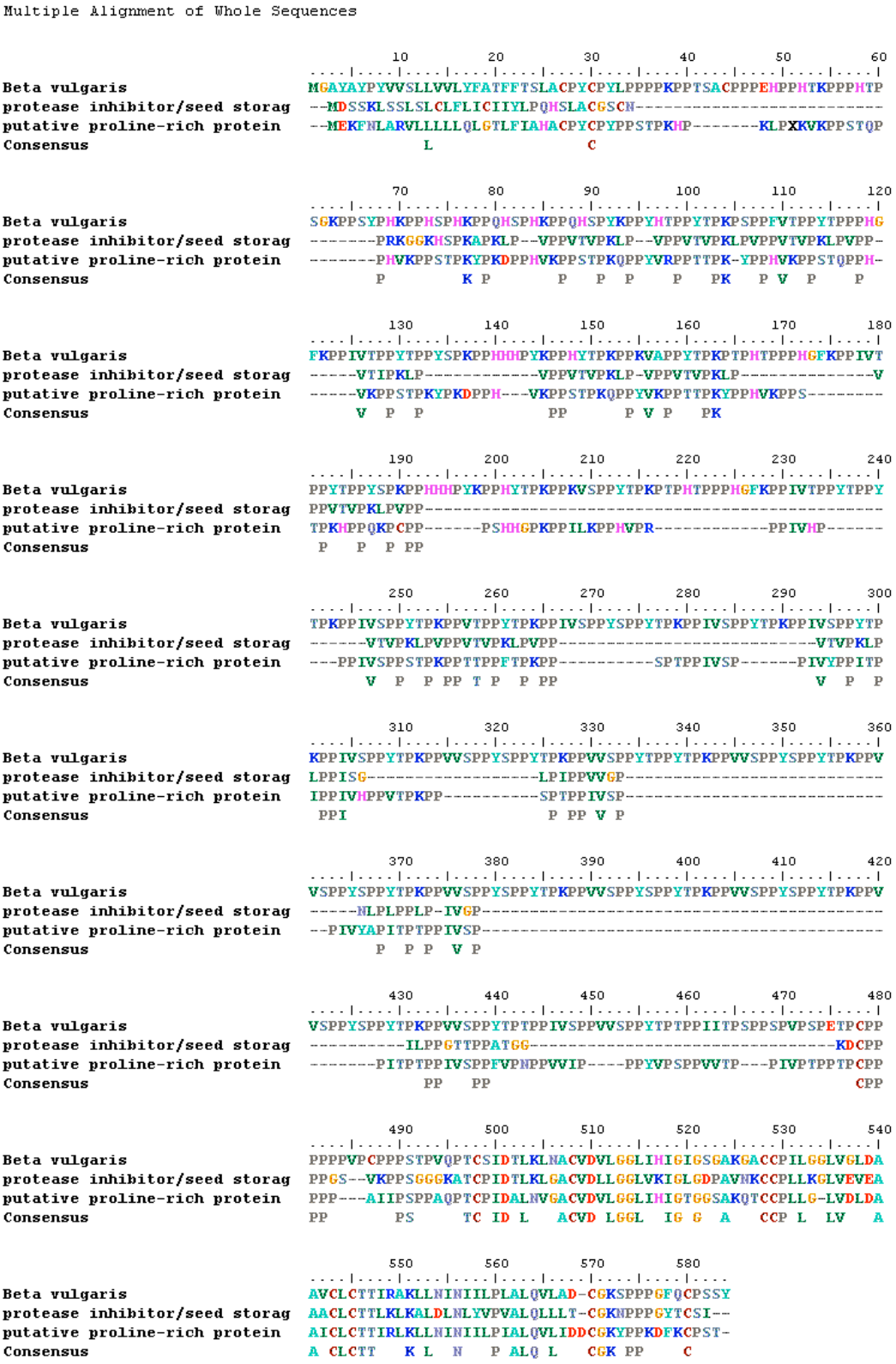
9.4.2 *Beta vulgaris* RS1 versus Proline Rich Protein from *Solanum brevidens*



9.4.3 *Beta vulgaris* RS1 versus LTP from *Arabidopsis thaliana*



9.5 Multiple Alignments of Sequences Similar to RS1



9.6 Results from MOTIF

9.6.1 When Querying Using the Peptide Chain

PRINTS	Position(Score)	Description
GPCRRHODOPSN4	8..29(1076) 538..559(1056) 12..33(1055) 512..533(1045) 527..548(1044) 540..561(1040) 549..570(1034) 530..551(1032) 501..522(1029) 517..538(1029) 521..542(1028) 1..20(1025) 523..544(1025) 556..577(1025) 542..563(1024) 1..22(1021) 11..32(1020) 4..25(1016) 515..536(1016) 2..23(1011) 528..549(1010) 10..31(1009) 555..576(1004) 524..545(1002) 6..27(1001) 7..28(1001)	Rhodopsin-like GPCR superfamily signature
GPCRRHODOPSN1	538..562(1052) 490..514(1031) 5..29(1029) 523..547(1028) 535..559(1028) 528..552(1026) 8..32(1022) 531..555(1018) 540..564(1018) 525..549(1009) 534..558(1002) 2..26(1001)	Rhodopsin-like GPCR superfamily signature
TYPE3IMRPROT3	499..521(1021)	Type III secretion system inner membrane R protein
GPCRRHODOPSN6	7..31(1018) 6..30(1014) 10..34(1012)	Rhodopsin-like GPCR superfamily signature
GPCRRHODOPSN5	9..32(1015)	Rhodopsin-like GPCR superfamily signature
SUGRTRNSPORT2	512..531(1007)	Sugar transporter signature

When these results are highlighted on the peptide chain (overlaps exist), the following results are achieved (text colour is indicative of the three domains in the protein):

9.6.1.1 Rhodopsin-like GPCR Superfamily Signature 4

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420
VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.1.2 Rhodopsin-like GPCR Superfamily Signature 1

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420
VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.1.3 Type III Secretion System Inner Membrane R Protein

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420

VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.1.4 Rhodopsin-like GPCR Superfamily Signature 6

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420
VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.1.5 Rhodopsin-like GPCR Superfamily Signature 5

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420
VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.1.6 Sugar Transporter Signature

MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP PHTKPPPHTP 60
SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP YTPKPSPPFV TPPYTPPPHG 120
FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT PKPPKVAPPY TPKPTPHTPP PHGFKPPIVT 180
PPYTPPYSPK PPHHHPYKPP HYTPKPPKVS PPYTPKPTPH TPPPHGFKPP IVTPPYTPPY 240
TPKPPIVSPP YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY SPPYTPKPPV 360
VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV 420
VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP VVSPPYTPTP PIITPSPPSP VPSPETPCPP 480
PPPPVPCPPP STPVQPTCSI DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA 540
AVCLCTTIRA KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

9.6.2 When Querying Using the Nucleotide Chain

Transfac	Position(Score)	Name	Description
M00345	1483..1490(91) 187..194(89)	GAmyb	GA-regulated myb gene from barley
M00353	526..516(90) 685..675(90) 1505..1515(86) 459..469(85)	Dof2	Dof2 - single zinc finger transcription factor
M00354	526..516(90) 685..675(90)	Dof3	Dof3 - single zinc finger transcription factor
M00352	41..31(87) 526..516(85) 685..675(85)	Dof1	Dof1 / MNB1a - single zinc finger transcription factor
M00355	526..516(85) 685..675(85)	PBF	PBF (MPBF)

When these results are highlighted on the nucleotide chain, the following results are achieved (text colour is indicative of the three domains in the protein):

9.6.2.1 GA-regulated myb Gene from Barley

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC 60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT 120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG 180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA 240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA 300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT 360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT 420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT 480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT 540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT 600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC 660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC 720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC 780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT 840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA ACCTCCTGTT 1080

GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG
CCCACCATAT 1140
AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC
ACCCTACACA 1200
CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA
ACCCCCAGTT 1260
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG
CCCACCATAC 1320
ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC
ACCAACACCT 1380
CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC
TTGTCCACCA 1440
CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC
ATGCTCCATT 1500
GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA
TATTGGTATA 1560
GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG
GTTAGATGCT 1620
GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT
TATTCTTCCT 1680
CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT
CCAATGTCCT 1740
TCTTCTTATT AG 1752

9.6.2.2 Dof2 - Single Zinc Finger Transcription Factor

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC 60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT 120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG 180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA 240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA 300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT 360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT 420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT 480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT 540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT 600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC 660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC 720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC 780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT 840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA ACCTCCTGTT 1080
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 1140

AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC ACCCTACACA 1200

CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA ACCCCCAGTT 1260

GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAC 1320

ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC ACCAACACCT 1380

CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC TTGTCCACCA 1440

CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC ATGCTCCATT 1500

GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA TATTGGTATA 1560

GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG GTTAGATGCT 1620

GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT TATTCTTCCT 1680

CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT CCAATGTCCT 1740

TCTTCTTATT AG 1752

9.6.2.3 Dof3 - Single Zinc Finger Transcription Factor

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC 60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT 120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG 180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA 240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA 300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT 360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT 420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT 480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT 540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT 600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC 660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC 720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC 780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT 840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA ACCTCCTGTT 1080
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 1140

AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC ACCCTACACA 1200
CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA ACCCCCAGTT 1260
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAC 1320
ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC ACCAACACCT 1380
CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC TTGTCCACCA 1440
CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC ATGCTCCATT 1500
GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA TATTGGTATA 1560
GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG GTTAGATGCT 1620
GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT TATTCTTCCT 1680
CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT CCAATGTCCT 1740
TCTTCTTATT AG 1752

9.6.2.4 Dof1 / MNB1a - Single Zinc Finger Transcription Factor

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC 60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT 120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG 180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA 240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA 300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT 360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT 420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT 480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT 540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT 600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC 660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC 720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC 780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT 840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA ACCTCCTGTT 1080
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 1140

AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC ACCCTACACA 1200
CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA ACCCCCAGTT 1260
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAC 1320
ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC ACCAACACCT 1380
CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC TTGTCCACCA 1440
CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC ATGCTCCATT 1500
GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA TATTGGTATA 1560
GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG GTTAGATGCT 1620
GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT TATTCTTCCT 1680
CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT CCAATGTCCT 1740
TCTTCTTATT AG 1752

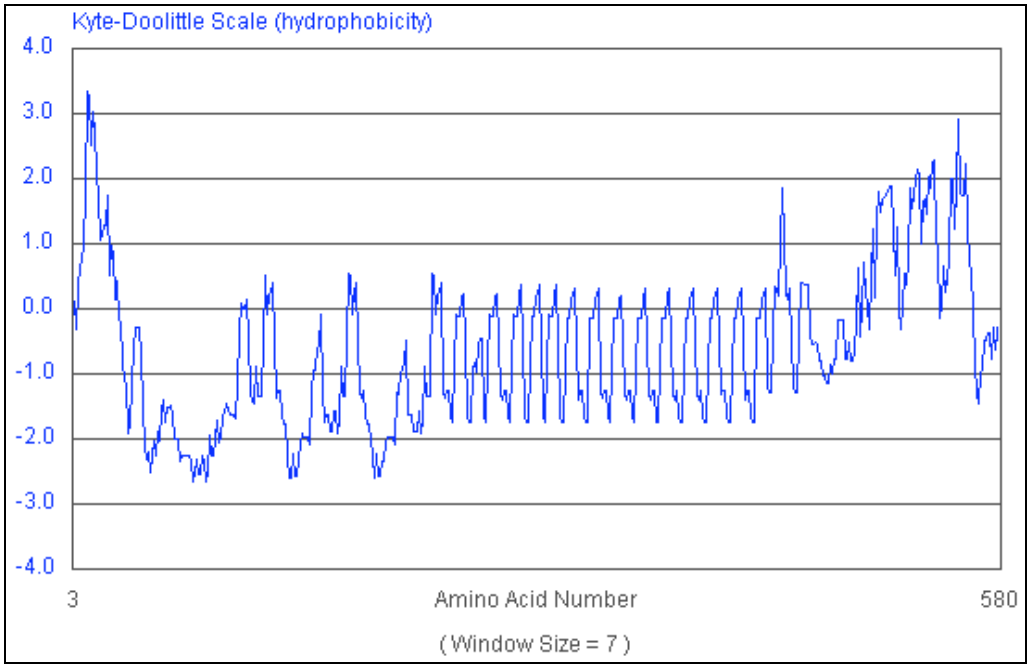
9.6.2.5 PBF (MPBF)

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC 60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT 120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG 180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA 240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA 300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT 360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT 420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT 480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT 540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT 600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC 660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC 720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC 780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT 840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA ACCTCCTGTT 1080
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAT 1140

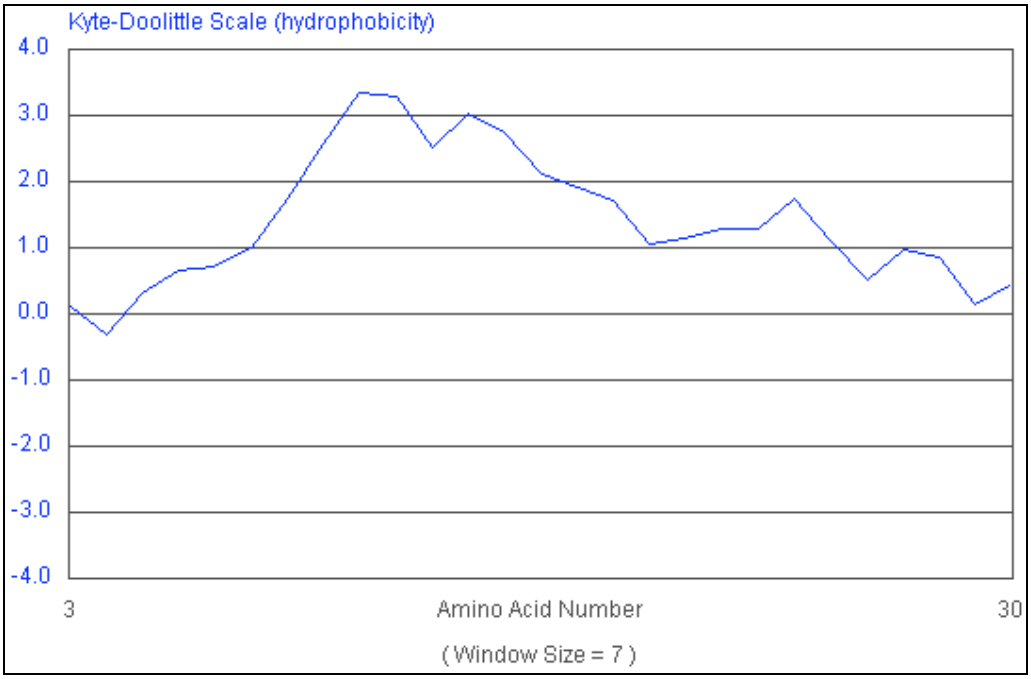
AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC
ACCCTACACA 1200
CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA
ACCCCCAGTT 1260
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG
CCCACCATAC 1320
ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC
ACCAACACCT 1380
CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC
TTGTCCACCA 1440
CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC
ATGCTCCATT 1500
GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA
TATTGGTATA 1560
GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG
GTTAGATGCT 1620
GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT
TATTCTTCCT 1680
CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT
CCAATGTCCT 1740
TCTTCTTATT AG 1752

9.7 Hydrophobicity Plots

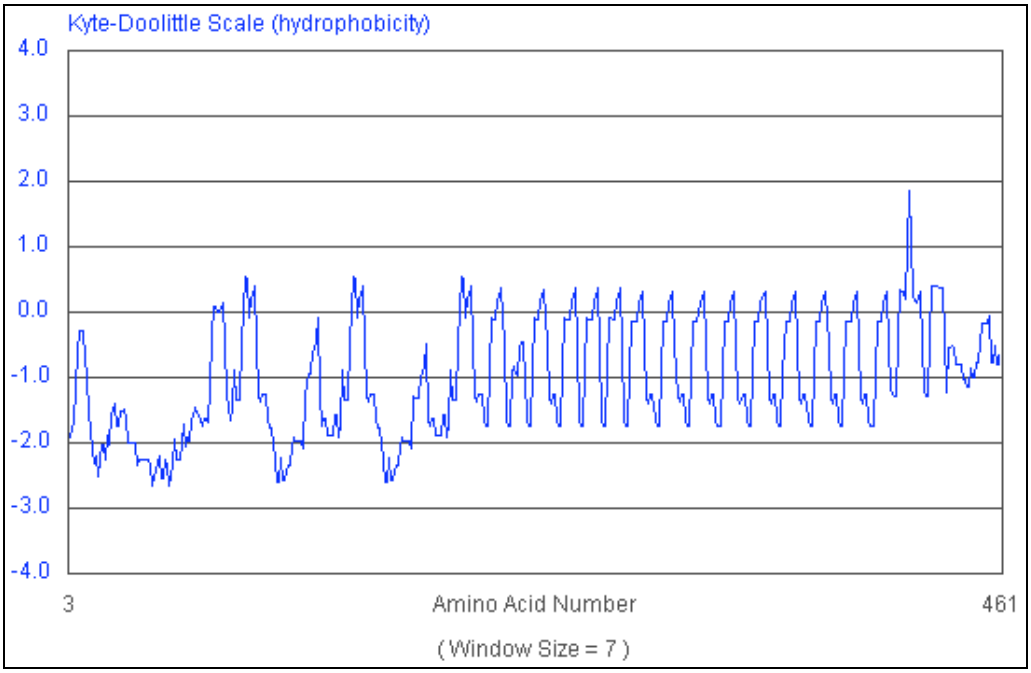
9.7.1 Whole Sequence



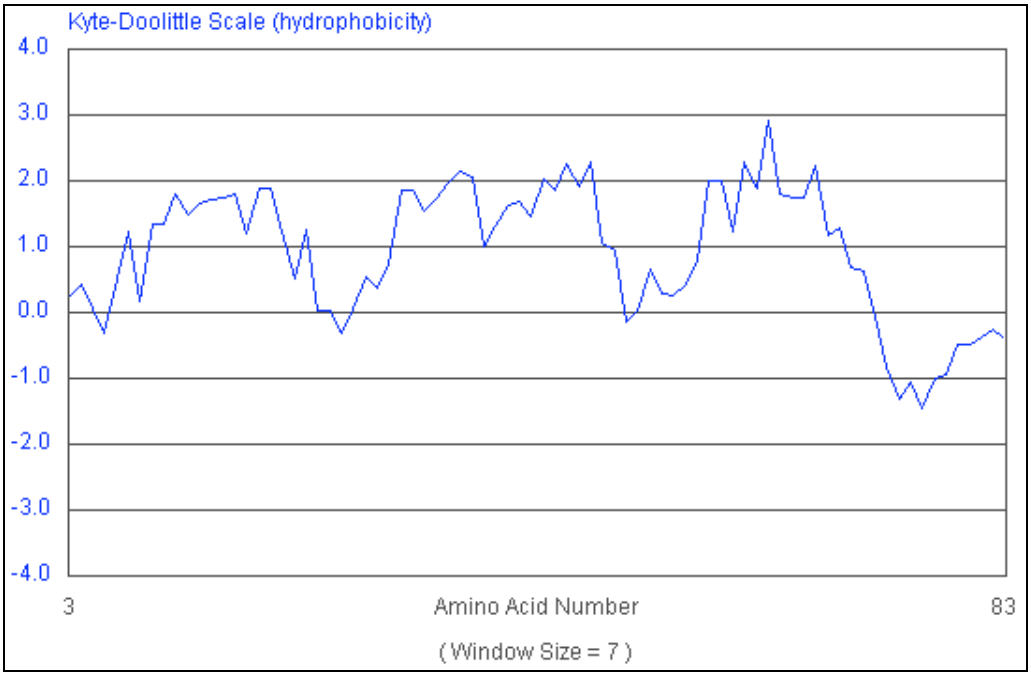
9.7.2 Leader Region



9.7.3 Proline Rich Region



9.7.4 Cysteine Rich Region



9.8 Repeats Analysis in the Proline Rich Region

9.8.1 Results from Tandem Repeats Finder

The output from Tandem Repeats Finder is shown below. Following the table is an illustrative attempt at showing these repeats in the sequence with the domains highlighted by the colour of the text. Highlighting and bold is used to highlight the presence of repeats. Underlining indicates where the repeats exist, yellow and green highlighting indicate changes in repeat pattern whilst darker green indicates an overlap between patterns:

Indices	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	A	C	G	T	Entropy (0-2)
201--285	24	3.7	24	92	7	133	31	42	0	25	1.56
341--716	159	2.4	159	97	0	707	32	43	3	21	1.68
712--956	33	7.2	33	75	14	210	28	45	7	18	1.77
709--798	30	2.9	30	82	11	110	28	46	3	21	1.67
817--956	66	2.1	66	94	0	244	27	45	10	16	1.79
916--1319	45	9.0	45	95	0	682	30	42	10	17	1.82

ATGGGGGCAT ATGCATATCC ATATGTTGTT AGTCTTTTGG TGGTTTTATA CTTTGCTACC
60
TTTTTCACTT CTTTAGCATG CCCTTATTGT CCTTACCTTC CTCCTCCCCC TAAACCACCT
120
ACTTCAGCTT GTCCTCCCCC TGAACACCCT CCACACACCA AGCCACCACC TCATACGCCG
180
TCCGGAAAAC CGCCTTCCTA TCCTCATAAA CCACCTCATT CTCCTCATAA ACCACCTCAA
240
CATTCTCCTC ATAAACCACC TCAACATTCT CCCTATAAAC CACCTTATCA CACCCCACCA
300
TATACTCCTA AACCGTCGCC GCCCTTTGTT ACTCCACCTT ACACACCACC ACCACATGGT
360
TTTAAACCTC CTATTGTCAC TCCACCATAC ACTCCACCTT ATTCTCCAAA ACCACCACAT
420
CATCATCCCT ACAAACCACC TCATTACACC CCTAAACCTC CTAAAGTAGC ACCACCATAT
480
ACTCCTAAAC CTACACCACA CACCCCACCA CCACATGGCT TTAAACCTCC TATTGTCACT
540
CCACCATACA CTCCACCTTA TTCTCCAAAA CCACCACATC ATCATCCCTA CAAACCACCT
600
CATTACACAC CTAAACCTCC TAAAGTATCA CCACCATATA CTCCTAAACC TACACCACAC
660
ACCCCACCAC CACATGGCTT TAAACCTCCT ATTGTCACCC CACCATACAC TCCACCCTAC
720
ACACCCAAAC CTCCTATTGT TAGCCCACCC TATACACCAA AACCTCCTGT TACTCCACCC
780
TACACACCTA AACCTCCTAT TGTCAGCCCA CCATATAGCC CACCCTACAC ACCTAAACCT
840
CCCATTGTGA GCCCACCTTA CACACCAAAA CCCCCTATTG TGAGCCCACC
CTACACACCT 900
AAACCCCCCA TTGTGAGCCC ACCCTACACA CCAAAACCTC CTGTTGTGAG
CCCACCATAT 960
AGCCCGCCCT ACACACCAAA ACCTCCTGTT GTGAGCCCAC CATATACCCC
ACCCTACACA 1020
CCAAAACCTC CTGTTGTGAG CCCACCATAT AGCCCACCCT ACACACCAAA
ACCTCCTGTT 1080
GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG
CCCACCATAT 1140
AGCCCACCCT ACACACCAAA ACCTCCTGTT GTAAGCCCAC CATATAGTCC
ACCCTACACA 1200

CCAAAACCCC CAGTTGTAAG CCCACCATAT AGTCCACCCT ACACACCAAA ACCCCCAGTT 1260

GTAAGCCCAC CATATAGTCC ACCCTACACA CCAAAACCTC CTGTTGTGAG CCCACCATAC 1320

ACACCAACAC CCCCAATAGT AAGCCCACCA GTAGTAAGTC CACCATACAC ACCAACACCT 1380

CCTATTATAA CTCCTAGCCC ACCAAGTCCC GTACCGAGTC CCGAAACACC TTGTCCACCA 1440

CCACCACCAC CAGTACCATG CCCACCTCCT TCGACACCGG TTCAACCAAC ATGCTCCATT 1500

GATACGTTAA AGTTGAATGC ATGTGTTGAT GTACTAGGAG GGCTCATACA TATTGGTATA 1560

GGAAGTGGTG CTAAAGGTGC ATGTTGTCCA ATCTTAGGTG GTCTTGTAGG GTTAGATGCT 1620

GCTGTTTGTC TTTGCACTAC TATCAGAGCT AAACTTCTAA ACATCAACAT TATTCTTCCT 1680

CTTGCTCTTC AAGTTTTGGC TGATTGTGGC AAATCTCCTC CTCCTGGTTT CCAATGTCCT 1740

TCTTCTTATT AG 1752

9.8.2 Results from Radar

The results from Radar are not very useful to the project other than to illustrate the reason for designing specially designed software in the next section. Radar results can be seen below:

```
------------------------------------------------------------------------
No. of Repeats|Total Score|Length   |Diagonal| BW-From|    BW-To|    Level
             8|     790.41|       44|      44|     346|      389|        1
------------------------------------------------------------------------
  112-  157 (104.11/15.77)   PPYTpP...PHGF.....KPPIVTPPYTPPYSPKPP.HHHPYKPPhYTPKPPKVA
  158-  210 (103.57/15.60)   PPYT.PkptPHTPpphgfKPPIVTPPYTPPYSPKPP.HHHPYKPPhYTPKPPKVS
  211-  248 (88.33/11.07)    PPYT.P...KPTP.....HTP.......PPHGFKPPiVTPPYTPP.YTPKPPIVS
  249-  284 (92.70/12.37)    PPY........TP.....KPPV.....TPPYTPKPPiVSPPYSPP.YTPKPPIVS
  285-  317 (84.19/ 9.84)    PPY........TP.....KPPIVSPPY....TPKPPiV....SPP.YTPKPPVVS
  318-  362 (117.62/19.79)   PPYS.P...PYTP.....KPPVVSPPYTPPYTPKPPvVSPPYSPP.YTPKPPVVS
  363-  392 (82.68/ 9.39)    PPYS.P...PYTP.....KPPVVSPPYSPPYTPKPP..............VVS
  393-  437 (117.21/19.67)   PPYS.P...PYTP.....KPPVVSPPYSPPYTPKPPvVSPPYSPP.YTPKPPVVS
------------------------------------------------------------------------
------------------------------------------------------------------------
No. of Repeats|Total Score|Length   |Diagonal| BW-From|    BW-To|    Level
             2|      55.05|       12|      29|      48|       60|        7
------------------------------------------------------------------------
   48-   60 (26.66/ 9.61)    EHPPHtKPPPHTP
   80-   91 (28.39/ 6.56)    QHSPH.KPPQHSP
------------------------------------------------------------------------
------------------------------------------------------------------------
No. of Repeats|Total Score|Length   |Diagonal| BW-From|    BW-To|    Level
             2|      43.77|       12|      18|     507|      524|        8
------------------------------------------------------------------------
  507-  520 (18.50/28.19)    ACVDVLGGLIhiGI
  527-  538 (25.27/12.57)    ACCPILGGLV..GL
------------------------------------------------------------------------
```

9.8.3 Results from Specially Designed Software

Results from the output file of the specially designed software are in the format of:

```
Identified Repeat # 1
----------------------

Starting at residue 72
Repetition 1:  HSPHKPPQ
Repetition 2:  HSPHKPPQ


Identified Repeat # 2
----------------------

Starting at residue 126
Repetition 1:  TPPY
Repetition 2:  TPPY


Identified Repeat # 3
----------------------

Starting at residue 150
Repetition 1:  PKP
Repetition 2:  PKVAP
```

This in turn leads to the generation of the summary:

SPPYTPKPPVVSPPY - Found 9 duplicates
TPPY - Found 3 sets of 2 duplicates
PPYTPKPPV$_{T}$\$^{S}$ - Found 2 duplicates
TPH / TPPPH - Found 2 sets of 2 duplicates
PKP / PKV$_{A}$\$^{S}$P - Found 2 sets of 2 duplicates
HSPHKPPQ - Found 2 duplicates
PPY$_{S}$\$^{T}$ - Found 2 duplicates

The large repeat identified by the repeating of the hydrophobicity plot is:

TPPPHGFKPPIVTPPYTPPYSPKPPHHHPYKPPHYTPKPPKVSPPYTPKPTPH

A summary of the location of these repeats would be:

73-88	2x	HSPHKPPQ
115-243	2.5x	TPPPHGFKPPIVTPPYTPPYSPKPPHHHPYKPPHYTPKPPKVSPPYTPKPTPH
127-134	2x	TPPY
151-158	2x	PK(VA)P
165-172	2x	TP(PP)H
180-187	2x	TPPY
204-211	2x	PK(VS)P
218-225	2x	TP(PP)H
233-240	2x	TPPY
249-269	2x	PPYTPKPPVT
270-277	2x	PPYS
279-300	2x	KPPIVSPPYTP
306-440	9x	SPPYTPKPPVVSPPY

A graphical display of all these repeat patterns is shown below. Each repeat is displayed in alternative highlighting with the pattern underlined to indicate where the repeats are. The larger repeat is highlighted by being italicised with bold showing the location of the repeats:

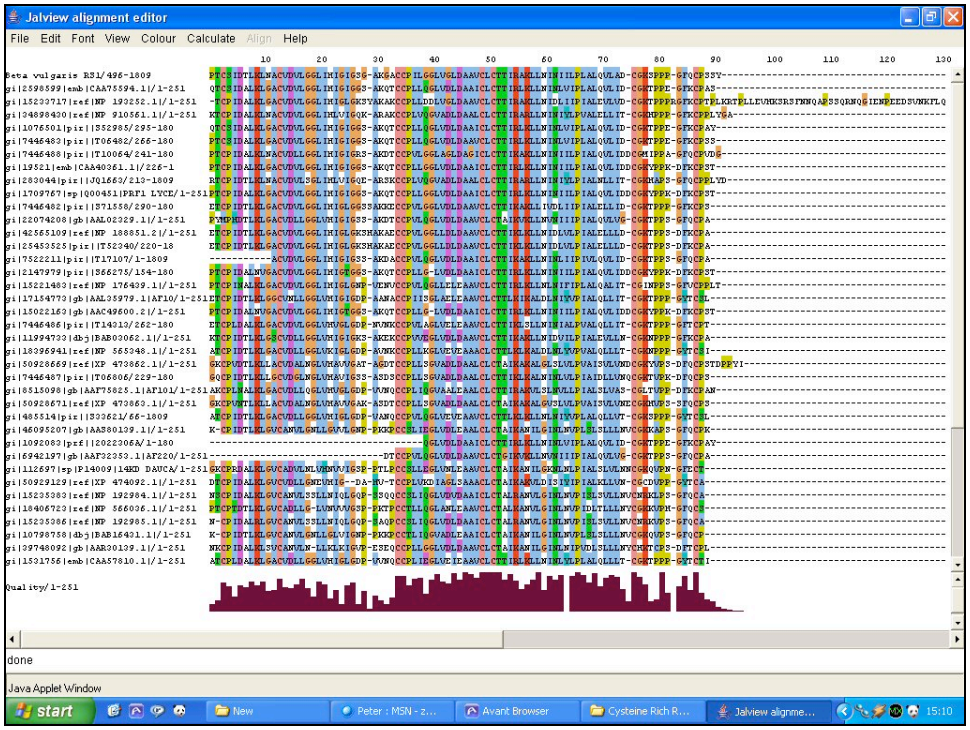
MGAYAYPYVV SLLVVLYFAT FFTSLACPYC PYLPPPPKPP TSACPPPEHP 50
PHTKPPPHTP SGKPPSYPHK PPHSPHKPPQ HSPHKPPQHS PYKPPYHTPP 100
YTPKPSPPFV TPPY***TPPPHG FKPPIVTPPY TPPYSPKPPH HHPYKPPHYT*** 150
PKPPKVAPPY TPKPTPH*TPP PHGFKPPIVT PPYTPPYSPK PPHHHPYKPP* 200
HYTPKPPKVS PPYTPKPTPH ***TPPPHGFKPP IVTPPYTPPY TPK***PPIVSPP 250
YTPKPPVTPP YTPKPPIVSP PYSPPYTPKP PIVSPPYTPK PPIVSPPYTP 300
KPPIVSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYTPPYT PKPPVVSPPY 350
SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT 400
PKPPVVSPPY SPPYTPKPPV VSPPYSPPYT PKPPVVSPPY TPTPPIVSPP 450

VVSPPYTPTP PIITPSPPSP VPSPETPCPP PPPPVPCPPP STPVQPTCSI 500
DTLKLNACVD VLGGLIHIGI GSGAKGACCP ILGGLVGLDA AVCLCTTIRA 550
KLLNINIILP LALQVLADCG KSPPPGFQCP SSY 583

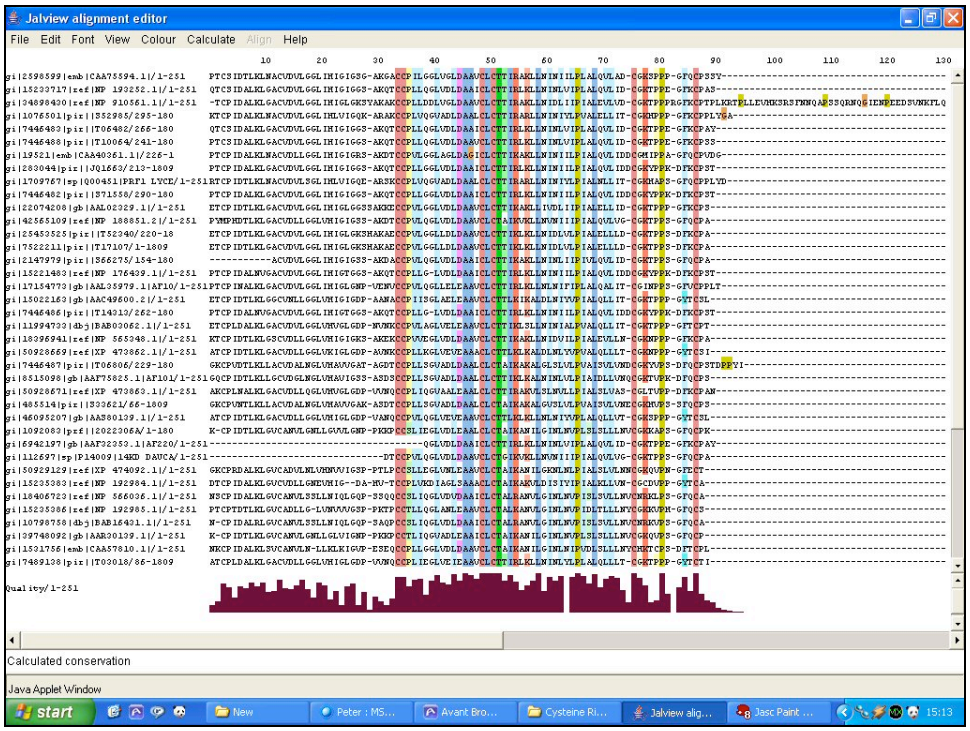
9.9 Sequence Conservation in the Cysteine Rich Region

9.9.1 Results from ClustalW

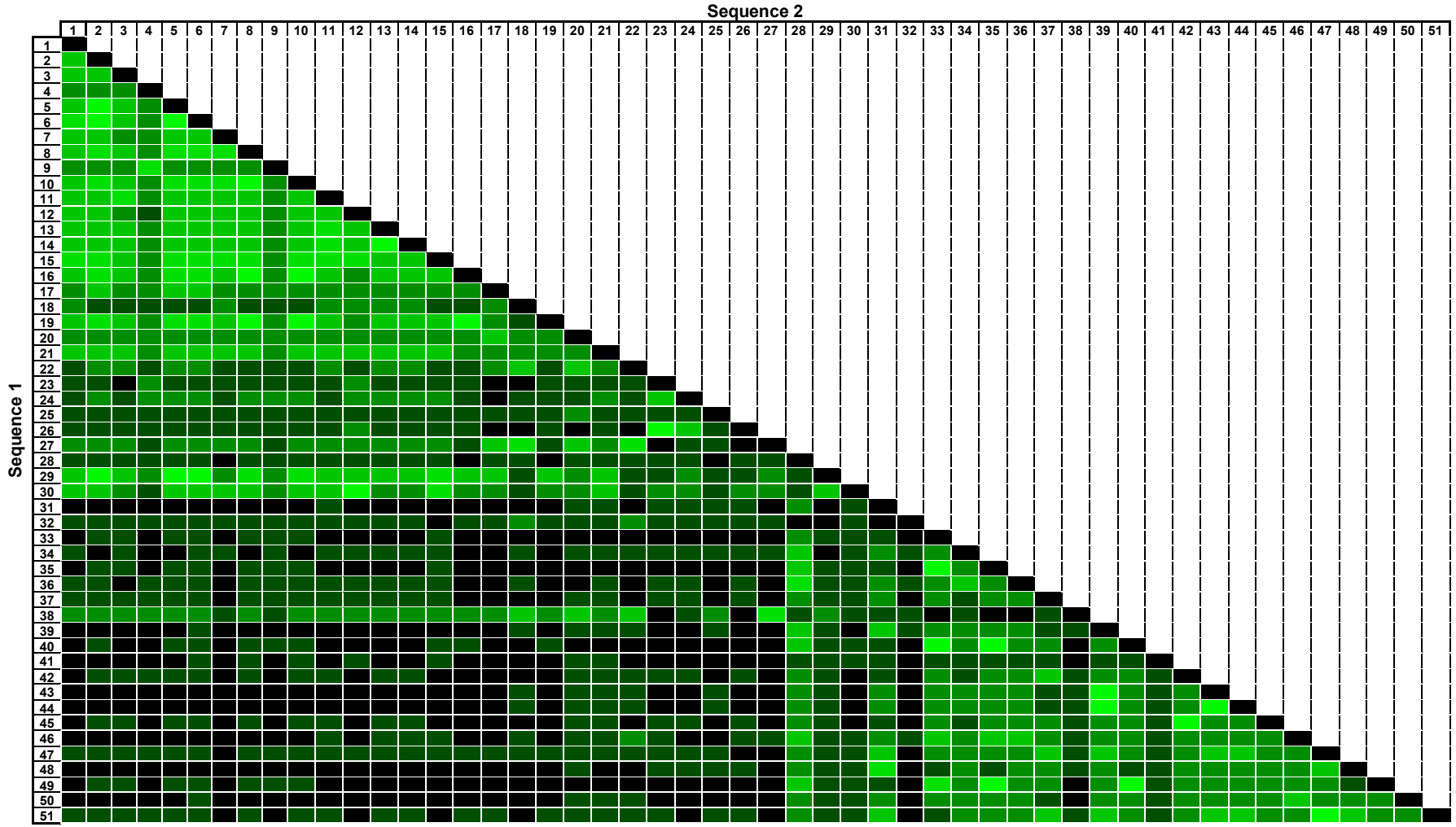
The graphical output from ClustalW shows where residues are similar in properties:



Further analysis using ClustalW shows where conservation of amino acids is highest:



Using the alignment scores of ClustalW creates the following score map:



The following is a list showing the alignment scores of RS1 against the other proteins analysed. These have been sorted into order of alignment score:

Protein NCBI Access Code	Alignment Score against RS1
7446483	80
7522211	80
19521	78
1709767	78
2598599	77
7446488	77
1076501	75
2147979	74
15022163	74
1092083	74
15233717	73
7446482	73
22074208	72
11994733	72
6942197	71
42565109	70
25453525	70
34898430	68
283044	67
7446486	66
15221483	65
485514	65
1531756	63
17154773	61
18396941	58
7446487	58
50928669	53
50928671	52
46095207	51
50929129	51
18406723	51
8515098	50
10798758	50
39748092	50
50911525	50
50927841	50
2226329	48
399204	48
112697	47
7489138	47
15221206	47
18413820	47
15235383	45
15235386	45
7959089	45
688422	45
9211012	45
15235388	45
15237964	45
15235390	44

9.9.2 A Phylogenetic Tree as a Phylogram

The phylogram generated by ClustalW is shown below: