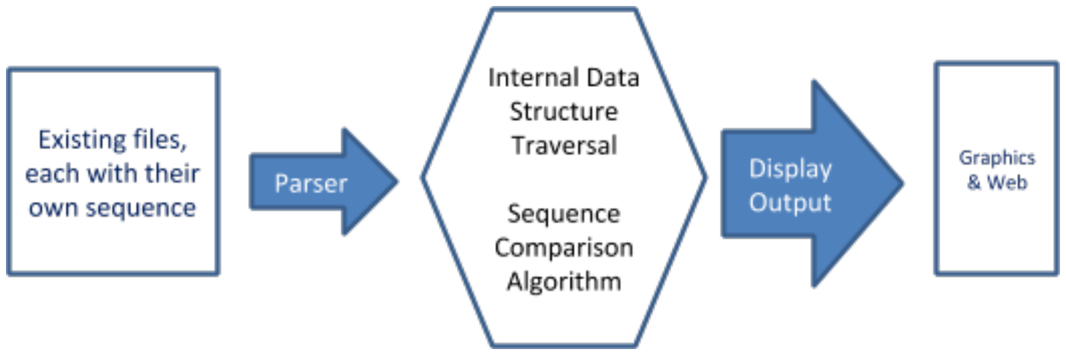
CS51 – Bioinformatics Draft Specification

N. Katz & R. Burgess
April 13, 2014

Introduction: This is a proposal for a bioinformatics tool that can be used to compare genetic sequences and search for similar sequences based upon an entered sequence (via an industry standard files for storing sequences). We may focus specifically on sequences found in microbes that point to evolutionary relatedness.

Project will consist of 4 areas of development:
1. Developing a parser that takes existing files, each with their own DNA sequence, and uses them to build an internal data structure.
2. A data structure and a set of functions for traversing it.
3. A comparison algorithm for comparing an entered sequence with another sequence - either one within the data structure, or a second entered sequence.
4. A GUI for displaying an input field and the output results, ideally existing in a browser.


Existing files, each with their own sequence
Parser
Internal Data Structure Traversal
Sequence Comparison Algorithm
Display Output
Graphics & Web

Overview:

Currently the plan is to use biocaml (see https://www.youtube.com/watch?v=rzrqcTWc2V8 for a biocaml overview and biocaml.org for detailed overview) as our main bioinformatics tool and build upon it in the four areas above.

1. **Parser**: Currently biocaml supports multiple input formats (BAM, BAR, BED, FASTA, FASTQ, etc). This project will include writing a parser for Avro, adding support to output OCaml code. Avro is being used by the Adam project to specify new high-performance bioinformatics data formats. A tool to generate OCaml code from Avro files would thus automatically provide OCaml code to work with these new formats.

2. **Internal Data Structure**: Determine how Biocaml is storing the sequences that are parsed from these files and suggest and implement an alternative strategy (2-3 trees, Red-Black trees, etc)

3. **Comparison Algorithms**: Look at BEDtools (http://bedtools.readthedocs.org/en/latest/) as a guide for genome arthimetic and develop and implement a comparison algorithm.

4. **Display Output:** Write OCaml code to generate visualizations of bioinformatics data (a very crude version of the UCSC Genome Browser. This will allows the end-user to input sequence and potential matches.

Feature List

Core Features

Searching with one entered sequence: Using the Needleman-Wunsch algorithm (or something similar) the program will take an entered sequence and run through each DNA/RNA sequence in a set of known sequences, each of which occur in their own files. Each of these files is indexed at runtime and stored in a data structure. The program will compare the entered sequence to each known sequence.

What is displayed: The program will only display sequences within a certain threshold. If the threshold is set to sequences that are 80% similar, it will only display those known sequences that are 80% or more similar. For each sequence that meets the required threshold, the program will return the following:

1) Meta information - potentially the following:
    a) name of species to which the sequence belongs
    b) Chromosome number (if applicable)
    c) Location on chromosome or larger sequence (if applicable)
2) sequence name (if it exists)
3) a value indicating the degree of similarity to other sequence

Comparing two sequences: Using the same algorithm as above, the program will take two entered sequences and return a value indicating degree of similarity.

Cool Extensions

Visual Representations of Output

We may Write OCaml code to generate visualizations of bioinformatics data - potentially a very crude version of the UCSC Genome Browser. This will allows the end-user to see potential matches more readily.

To control the output, the user may determine the similarity threshold.

Links: For each resulting sequence that gets returned, we may try to display a link to the actual FASTA file or DNA/RNA sequence.

In addition to returning sequences with a particular threshold, the program might also return a particular number of sequences based on a number the user enters.

Upon clicking on a given sequence, the program may display two sequences showing regions of similarity or dissimilarity.

The program might display the output sequences in a more visually informative way, such as a phylogenetic tree for microbes wherein the nodes are species names. An excerpt below describes the significance of these representations and their pertinence to evolutionary systematics.

In the laboratories of Woese and others, it was shown that phylogenetic relationships of bacteria, and, indeed, all life-forms, could be determined by comparing a stable part of the genetic code... The part of the DNA now most commonly used for taxonomic purposes for bacteria is the 16S rRNA gene. From http://www.ncbi.nlm.nih.gov/pmc/articles/PMC523561/

Identification

The program may also

- -Identify the start and stop codons in a gene sequence and return the location(s) of these codons.
- -return the number of codons in a DNA sequence.
- -return the number of amino acids in an amino acid sequence

Conversion

The program may also:

- -convert a DNA sequence to its RNA counterpart.
- -convert a DNA sequence to its complementary DNA sequence
- -convert a DNA or RNA sequence to its corresponding amino-acid sequence
- -return the number of bases in a DNA sequence.

Technical Specification

Modules Included:

Input_Parser- Ocaml Code that opens a file, potentially at runtime, and converts a set of sequences into our internal data structure for searching algorithms to work on (Sample http://biocaml.org/doc/dev/api/Biocaml_fasta.html

Helper modules may include a functor that can take sequence and map them onto different data structures, such as lists and red-black trees.

Internal_data - Ocaml code that stores baseline sequences for Comparison - Sample (http://biocaml.org/doc/dev/api/Biocaml_table.html)

The key features in this file would be the structure of the red-black data tree itself. This would probably include a data type that includes how a leaves, branches, and nodes would be represented in code.

Algorithm - Ocaml code that perform sequencing comparisons - Sample (http://biocaml.org/doc/dev/api/Biocaml_pwm.html)

Other resources include:

Sequence Alignment Algorithm
http://www.cmb.usc.edu/papers/msw_papers/msw-103.pdf

Needleman-Wunsch Algorithm
http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm

This would include an OCaml implementation of a sequence-comparison algorithm such as Needleman-Wunsch. (insert links to algorithm here.)

Graphics Display - Ocaml code that implements a GUI for output analysis - Sample (http://genome.ucsc.edu/cgi-bin/hgTracks?db=hg19&position=chr21%3A33031597-33041570&hgsid=369943387_shR6NVp6AxaesiHHI844kfF0ftao)

Graphical Elements would include the following:
  Input fields
  Search Button
  Radio buttons (for selecting threshold, number of sequences to return, sequence type, etc.)
  Output (either as text or if time permits text with graphics, e.g. phylogenetic trees)

Each of these would have their own styling and may trigger events that the Graphics Display responds to.

We might implement the Graphics and Web piece with the help of the js_of_ocaml library, which converts OCaml to pure javascript code so that the UI will work in a web browser.

Data Type: Each DNA sequence will be stored in a data type that will contain information such as location in the chromosome, chromosome number, and name of the species. Each sequence will be in its own file - for instance, a FASTA file that contains meta-information (chromosome number, etc.)

Data Structure: Initially, we plan to populate and store our sequence data types in a list since we are still working out how to compare sequences. I do not yet know of an exact way of comparing sequences the way you compare numbers (using < or >), strings (ABC order), or webpages (the presence of keywords). So we are still looking into the ideal data structure for storing our sequence data types.

Questions we will investigate: Is there an advantage of using a tree as opposed to just a list if we are going to iterate through each sequence anyway? Is there a way to sort sequences in such a way so that branches or clades with largely dissimilar sequences do not have to be traversed? Maybe if the tree is ordered phylogenetically, wherein the root node is a common ancestor and each level of the tree corresponds to a subsequent generation. We will have to look into this.

<u>**Project Schedule**</u>

14-Apr	BioCaml Installed Successfully - Both
16-Apr	BioCaml Up and Running - Both
	Parser Code Milestones - Rich
20-Apr	Parser Code 50% Complete
23-Apr	Parser Code 90% Complete
25-Apr	Parser Code Complete/Tested
	Internal Data Structure Milestones - Rich and Nevin
18-Apr	Code 50% Complete
20-Apr	Code 90% Complete
22-Apr	Code Complete/Tested
	Algorithm Milestones - Nevin
24-Apr	Code 50% Complete
27-Apr	Code 90% Complete
30-Apr	Code Complete/Tested
	Graphics Milestones - Nevin
22-Apr	Code 50% Complete
25-Apr	Code 90% Complete
28-Apr	Code Complete/Tested
	Other Project Milestones - Both
25-Apr	Write-Up 50% Complete
30-Apr	Write-Up 100% Complete
28-Apr	Demo 50% Complete
30-Apr	Demo 100% Complete
1-May	Final Integrated Code/ Testing Complete
2-May	Project Submission

RESOURCES

Sequence Alignment Algorithm
http://www.cmb.usc.edu/papers/msw_papers/msw-103.pdf

Needleman-Wunsch Algorithm
http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm

RDP
http://rdp.cme.msu.edu/

BLAST Lesson
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3867762/

Algorithms for Molecular Biology
http://www.almob.org/

16S Ribosomal RNA Algorithms
NAST Algorithm
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1538769/
http://www.ncbi.nlm.nih.gov/pubmed/16845035
http://greengenes.secondgenome.com/downloads

Chimeric Sequences
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3044863/

List of sequence alignment software
http://en.wikipedia.org/wiki/List_of_sequence_alignment_software

http://www.biostars.org/p/14359/

Intro bioinfo projects
http://rosalind.info (in Python)

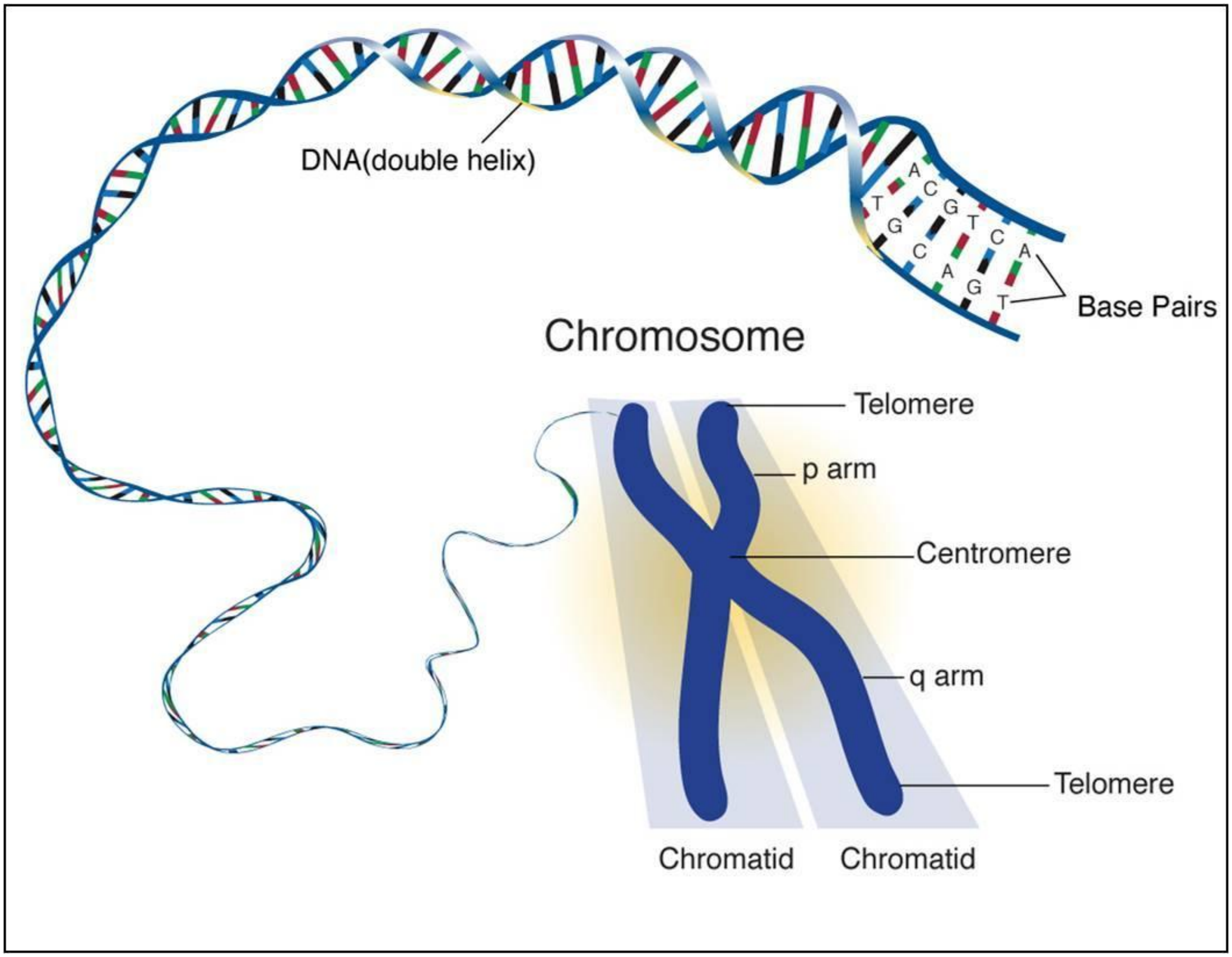
BIGCAT
http://www.bigcat.unimaas.nl/wiki/index.php/BiGCaT_people

Needleman-Wunsch
http://stackoverflow.com/questions/21199263/how-to-handle-multiple-optimal-edit-paths-implementing-needleman-wunsche-algorit

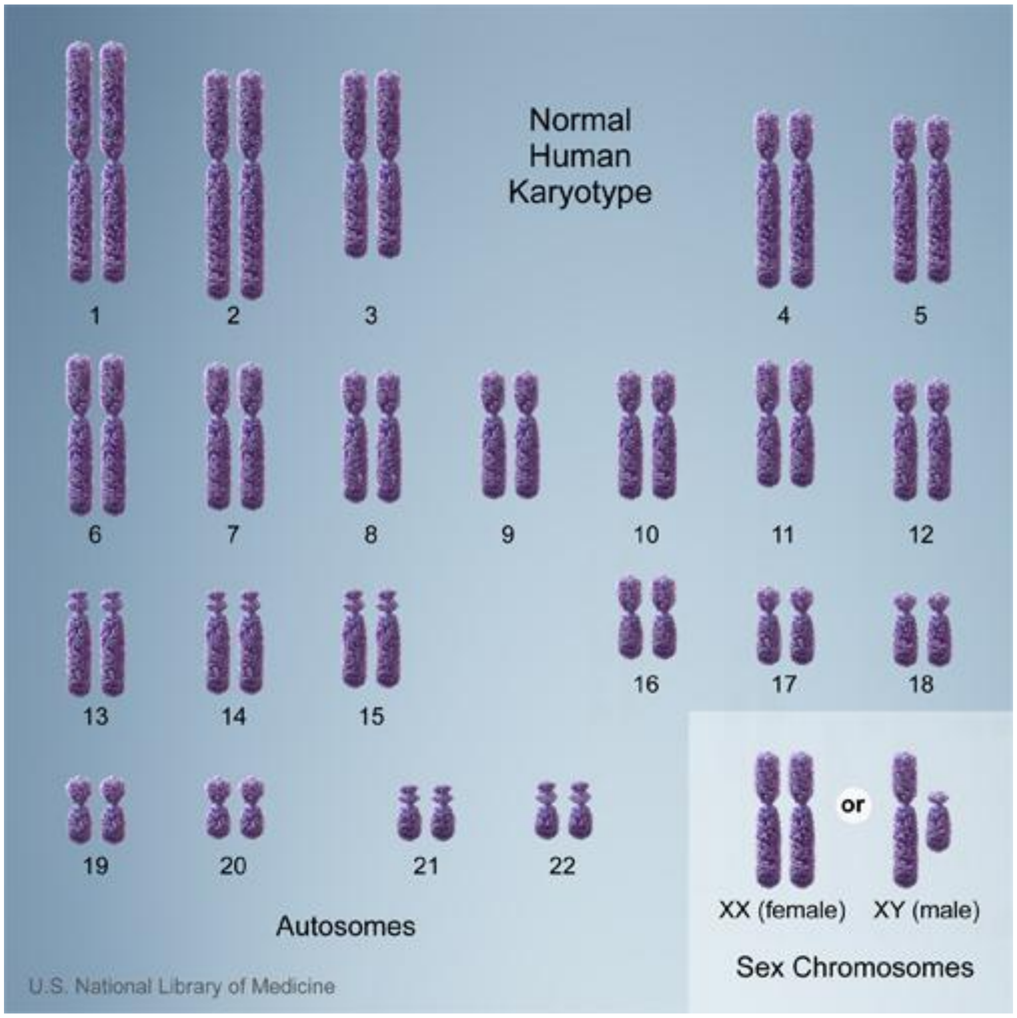
Memorandum
http://www.ueda.info.waseda.ac.jp/~sakai/ocaml/ocamlmemo.html

Background

The entire genetic information in a living thing is known as the genome. A genome is made up of a particular number of chromosomes, each like a "book" of DNA. A functional section of DNA on a chromosome is called a gene.

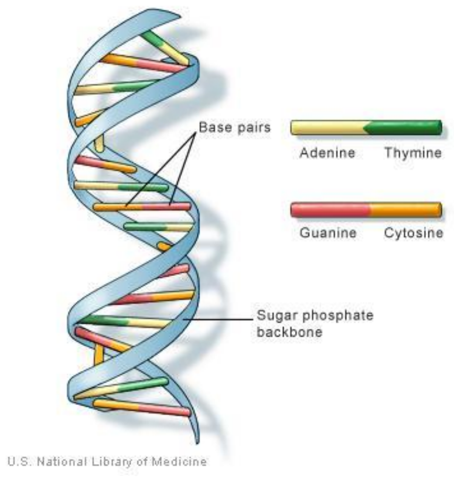


In humans there are 23 pairs of chromosomes. In a chromosome pair, the two chromosomes each have the same genes - though they may have different versions of the same gene.



A chromosome is basically tightly coiled up DNA. The information in DNA is encoded as a base sequence. A "letter" in this sequence can be one of four bases: Adenine, Thymine, Cytosine, and Guanine - or A, T, C, and G.

DNA occurs as a double-helix made of two strands. The bases of one strand bond to the bases of the other. The base "A" pairs with "T", while "G" pairs with "C."



So a 9-base section with two strands of DNA may look like this:

ATCGTGATC
TAGCACTAG

Only one strand has the actual information - the other is just used as a template for replicating the DNA and making temporary copies of genes called messenger RNA.

RNA & Proteins

The central dogma of molecular biology is DNA -> RNA -> Proteins

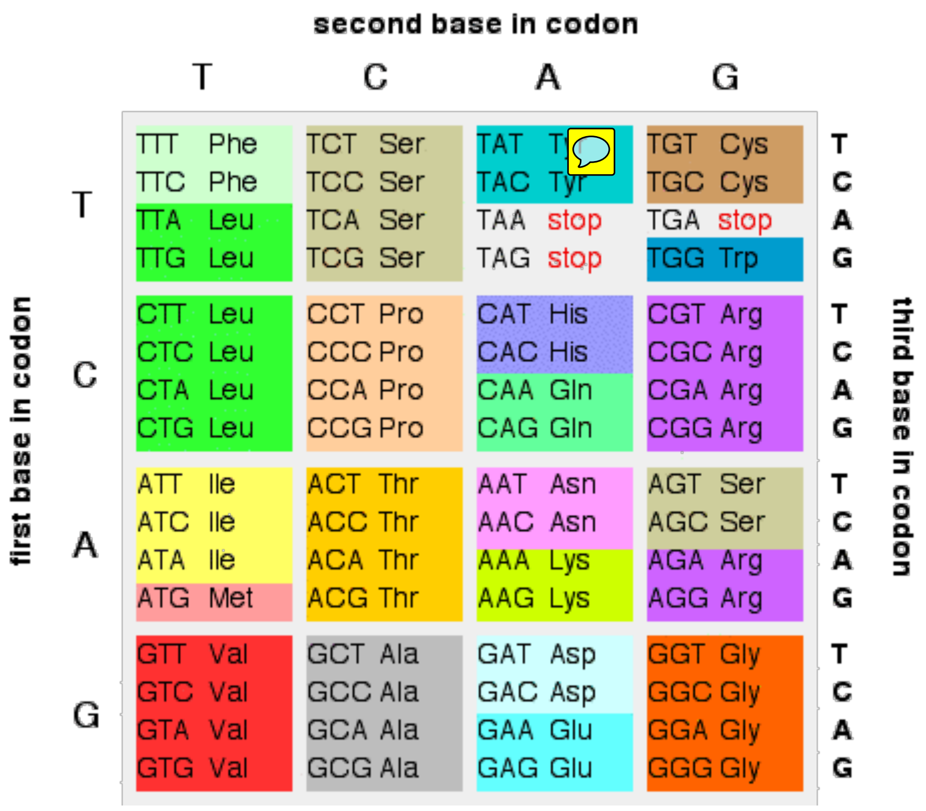
More specifically, DNA is the template for making messenger RNA, which is the blueprint for building proteins. Proteins are made out of chemicals known as amino acids.

RNA comes in several types, but the one being discussed here are mainly messenger RNA and secondarily ribosomal RNA. Messenger RNA is a single stranded copy of DNA.

It also has Adenine, Guanine, and Cytosine for the bases, but instead of Thymine it has uracil. So in the diagram below, the sequence in blue would be the complementary RNA sequence.

ATAGTGATC
UAUCACUAG

A base sequence occurs as 3-base "words" known as codons. Each codon corresponds to a specific amino acid. The codons in a sequence determines the sequence of amino acids that are produced.
Here's how codons map onto bases.



	second base in codon				
first base in codon	T	C	A	G	third base in codon
T	TTT Phe	TCT Ser	TAT T	TGT Cys	T
	TTC Phe	TCC Ser	TAC Tyr	TGC Cys	C
	TTA Leu	TCA Ser	TAA stop	TGA stop	A
	TTG Leu	TCG Ser	TAG stop	TGG Trp	G
C	CTT Leu	CCT Pro	CAT His	CGT Arg	T
	CTC Leu	CCC Pro	CAC His	CGC Arg	C
	CTA Leu	CCA Pro	CAA Gln	CGA Arg	A
	CTG Leu	CCG Pro	CAG Gln	CGG Arg	G
A	ATT Ile	ACT Thr	AAT Asn	AGT Ser	T
	ATC Ile	ACC Thr	AAC Asn	AGC Ser	C
	ATA Ile	ACA Thr	AAA Lys	AGA Arg	A
	ATG Met	ACG Thr	AAG Lys	AGG Arg	G
G	GTT Val	GCT Ala	GAT Asp	GGT Gly	T
	GTC Val	GCC Ala	GAC Asp	GGC Gly	C
	GTA Val	GCA Ala	GAA Glu	GGA Gly	A
	GTG Val	GCG Ala	GAG Glu	GGG Gly	G