Bioinformatics Assignment 1

Kotsampougioukoglou, Ilias (S1736880)

Abstract

The purpose of this assignment is to explore some core molecular biology and bioinformatics concepts. All these will be shown through some manipulations that will be performed on a selected human disease and a gene that is thought to be involved in it.

Figure 1. Schematic presentation of the gene

1. Answer to question 1

Testicular germ cell tumors (testicular cancer) is a very common cancer among young men and particularly it seems that strikes between the ages of 20-35 (1). This is the reason why there is a genetic basis for this disease and many studies have been conducted in order to reveal the secrets of this type of cancer. Furthermore many genes are considered to be involved in this disease, some of them are PDE11A, c-KIT, KIT-ligand(KITLG), SPRY4, and BAK1 (1; 2; 3). However our report will focus on KILTG(KIT ligand) gene which is also known as SF, MGF, SCF, SLF, DCUA, FPH2, FPHH, KL-1, Kitl, SHEP7, DFNA69.

The investigation was mainly conducted through the NCBI database in which there is a vast amount of data for many genes (4). From this database we had access to many human diseases and it was easy to find out which genes are responsible for their occurance. In this way we initialized our searching for cancer and especially for testicular cancer and our investigation led us to the KITLG gene. It is quite interesting that the same gene is conserved in chimpanzee, Rhesus monkey, dog, cow, mouse, rat, chicken, and frog (5). So a homologue exists in a model system such as the mouse and this is probably a true homologue (very low Evalue ~ e-100 means homologue). If we search with BLAST the sequence KITLG of human gene and the sequence Kitl of M.musculus gene (common mouse) then we can see that $Evalue = 1e - 143$ (for similarity) and $ident = 83\%$ (for identity) (5).

2. Answer to question 2

KIT-ligand gene is located on the chromosome 12 at 12q21.32 position. It has 10 exons and 9 introns as it is indicated at Figure 1.

The length of all exons is 5443 nucleotids and the length of introns is 82238 (total lenght of gene is 87681). Thus we find exons in a percentage of 6.21% and introns in the rest 93.79% of the whole gene.

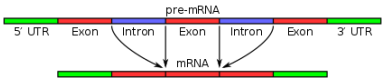


Figure 2. Example of exons and introns

The reference of the sequence that was chosen is NM_000899.4 and all the above information had been taken from NCBI (5). The transcript consists of 5460 nucleotids and the coding sequence is 15.05% of gene. The python code that was used to find how long the sequence is and also what proportion of this sequence the coding part is, can be found in Appendix A.

3. Answer to question 3

The difference between Coding Sequences (CDS) and cDNA is the UnTranslated Regions (UTRs). A CDS or coding sequence is the part of a transcript that is actually translated into protein. Therefore a CDS will (almost) always start with an AUG codon and stop at one of the three STOP codons (UAA,UGA,UAG).

The transcript however will also contain the UTRs which are not actually translated into protein. A cDNA sequence is derived from the transcript by reverse transcription and will, therefore, also contain the 5' and 3' UTRs.

Figure 3. Schematic presentation of unspliced transcript

The CDS of the gene depicted in Figure 3 above will only contain the ATG, the STOP and the two green regions (exons). The cDNA will contain all that and, in addition, the two UTRs. Of course, both the cDNA and the CDS will not contain the introns.

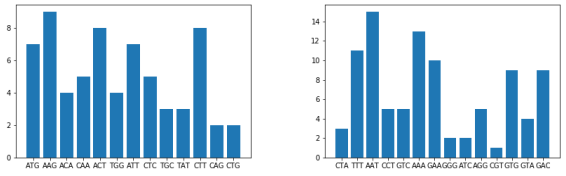
Thus, in our case we have in our possesion both the transcript (cDNA) and the coding sequence. So the translation will be made on the coding sequence. Our amino acid sequence is:

MKKTQTWILTCIYLQLLLFNPLVKTEGICRNRVTN
NVKDVTKLVANLPKDYMITLKYVPGMDVLPSHCWI
SEMVVQLSDSLTDLLDKFSNISEGLSNYSIIDKLV
NIVDDLVECVKENSSKDLKKSFKSPEPRLFTPEEF
FRIFNRSIDAFKDFVVASETSDCVVSSTLSPEKDS
RVSVTKPFMLPPVAASSLRNDSSSSNRKAKNPPGD
SSLHWAAMALPALFSLIIGFAFGALYWKKRQPSLT
RAVENIQINEEDNEISMLQEKEREFQEV*

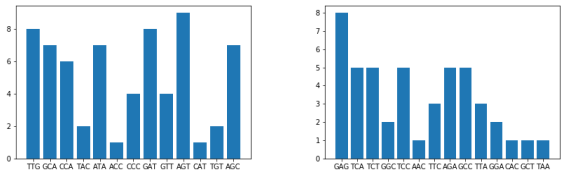
1st base	2nd base T		2nd base C		2nd base A		2nd base G		3rd base
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA[R]	Stop (*Ochre*)	TGA[R]	Stop (*Opal*)	A
	TTG		TCG		TAG[R]	Stop (*Amber*)	TGG	(Trp/W) Tryptophan	G
C	CTT	(Leu/L) Leucine	CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

Figure 4. DNA codon table

Our protein contains 274 amino acids. From the 64 possible codons the 54 are used in our sequence. The most common amino acid in our protein is S (Serine) with 31 appearances. As the Figure 4 indicates there are 6 different codons (TCT, TCC, TCA, TCG, AGT and AGC) for this amino acid and in our protein they appeared 5, 5, 5, 0, 9 and 7 times respectively. The Figure 5 show the appearances of the above mentioned codons. Python code used for all the above information is documented in appendix B.



(a) Figure A (b) Figure B



(c) Figure C (d) Figure D

Figure 5. Usage of codons in KITLG coding sequence. Python code used for these plots is documented in appendix C.

4. Answer to question 4

***Saccharomyces cerevisiae* [gbpln]: 14411 CDS's (6534504 codons)**

fields: [triplet] [frequency: **per thousand**] ([number])

```
UUU 26.1(170666)  UCU 23.5(153557)  UAU 18.8(122728)  UGU  8.1( 52903)
UUC 18.4(120510)  UCC 14.2( 92923)  UAC 14.8( 96596)  UGC  4.8( 31095)
UUA 26.2(170884)  UCA 18.7(122028)  UAA  1.1(  6913)  UGA  0.7(  4447)
UUG 27.2(177573)  UCG  8.6( 55951)  UAG  0.5(  3312)  UGG 10.4( 67789)

CUU 12.3( 80076)  CCU 13.5( 88263)  CAU 13.6( 89007)  CGU  6.4( 41791)
CUC  5.4( 35545)  CCC  6.8( 44309)  CAC  7.8( 50785)  CGC  2.6( 16993)
CUA 13.4( 87619)  CCA 18.3(119641)  CAA 27.3(178251)  CGA  3.0( 19562)
CUG 10.5( 68494)  CCG  5.3( 34597)  CAG 12.1( 79121)  CGG  1.7( 11351)

AUU 30.1(196893)  ACU 20.3(132522)  AAU 35.7(233124)  AGU 14.2( 92466)
AUC 17.2(112176)  ACC 12.7( 83207)  AAC 24.8(162199)  AGC  9.8( 63726)
AUA 17.8(116254)  ACA 17.8(116084)  AAA 41.9(273618)  AGA 21.3(139081)
AUG 20.9(136805)  ACG  8.0( 52045)  AAG 30.8(201361)  AGG  9.2( 60289)

GUU 22.1(144243)  GCU 21.2(138358)  GAU 37.6(245641)  GGU 23.9(156109)
GUC 11.8( 76947)  GCC 12.6( 82357)  GAC 20.2(132048)  GGC  9.8( 63903)
GUA 11.8( 76927)  GCA 16.2(105910)  GAA 45.6(297944)  GGA 10.9( 71216)
GUG 10.8( 70337)  GCG  6.2( 40358)  GAG 19.2(125717)  GGG  6.0( 39359)
```

Figure 6. Codon usage for saccharomyces cerevisiae (yeast)

The usage of codons for yeast are shown in Figure 6(6). From this figure it can be extracted the below table (Table 1) with the most rare codons (usage<=1%) for this species. Thus, if we were to try and express our human cDNA sequence in yeast (saccharomyces cerevisiae), some of the codons in our sequence might cause problems for expression. These are CTC 5, CCC 4, TAA 1, CAC 1, TGT 2, TGC 3, CGT 1, AGC 7, AGG 5, GGC 2, GGG 2 (the number after each codon is the appearances of this codon in our cDNA sequence).

Codon	usage(%)	Codon	usage(%)
CUC	0.54	CCC	0.68
UCG	0.86	CCG	0.53
ACG	0.80	GCG	0.62
UAA	0.11	UAG	0.05
CAC	0.78	UGU	0.81
UGC	0.48	UGA	0.70
CGU	0.64	CGC	0.26
CGA	0.30	CGG	0.17
AGC	0.98	AGG	0.92
GGC	0.98	GGG	0.60

Table 1. Table for rare codons for yeast

5. Answer to question 5

We have the four below coding sequence fragments which encode a homologous protein in different species.
1. CTGAAGCGGGAGGCTGAGACGCTGCGGGAGC GGGAGGGC
2. CTCAAGCGTGAGGCCGAGACCCTACGGGAGC GGGAAGGC
3. GAAGAGCTGAAGAGAGAGGCTGACAATTTAA AGGACAGA
4. AACGAGGAGCTCAAGCGAGAAGCTGATACGC TGAAGGAC

We have run a BLAST search in order to find the possible gene from these coding sequence fragments. The Python code for finding the possible gene is documented in Appendix D.

Some characteristic outputs from this piece of code are:
****
sequence: gi|366039935|ref|NM_008420.4| Mus musculus potassium voltage gated channel, Shab-related subfamily, member 1 (Kcnb1), mRNA
length: 11153
e value: 1.06561e-10
****
sequence: gi|1034137980|ref|XM_016938115.1| PREDICTED: Pan troglodytes potassium voltage-gated channel subfamily B member 1 (KCNB1), transcript variant X1, mRNA
length: 11926
e value: 1.06561e-10
****
sequence: gi|1034625101|ref|XM_006723784.3| PREDICTED: Homo sapiens potassium voltage-gated channel subfamily B member 1 (KCNB1), transcript variant X1, mRNA length: 11938
e value: 1.06561e-10
****
sequence: gi|1207179461|ref|XM_021477270.1| PREDICTED: Danio rerio potassium voltage-gated channel, Shab-related subfamily, member 1 (kcnb1), mRNA
length: 4395
e value: 1.06561e-10
****
sequence: gi|1199363037|ref|XM_021321541.1| PREDICTED: Fundulus heteroclitus potassium voltage-gated channel subfamily B member 1 (kcnb1), transcript variant X1, mRNA
length: 3937
e value: 1.06561e-10

From the above results can be concluded that the likely gene name is kcnb1. Furthermore, the first sequence is encountered mostly in mus masculus, the second one in pan troglodytes, homo sapiens, papio anubis, callithrix jacchus and in some other species, the third one in fundulus heteroclitus and the last one in danio rerio. Also it can be noticed that sequences 1 and 2 differ slightly. Because the sequences are aligned to the correct reading frame these two sequences are translated with the same amino acids. So the resulting protein does not differ and as a conclusion there are no functional implications.

For the above four sequences (1,2,3,4) we use the Needleman−Wunsch algorithm to compare sequence 1 to the other sequences with scoring: match +2, mismatch -3, indel -4.
Hence, the results are:
1-2: score 48
1-3: score -27
1-4: score -17
(7)

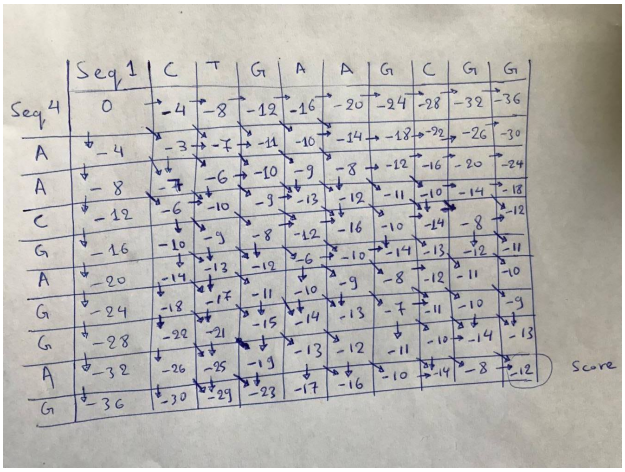


Figure 7. Needleman-Wunsch algorithm (with scoring: match +2, mismatch -3, indel -4) for CTGAAGCGG and AACGAGGAG sequences.

The alignment score is the sum of substitution scores and gap penalties and reflects how good the sequences can be aligned (less gaps). With this in our mind the comparison between sequence 1 and sequence 2 has the highest score. So they are well aligned comparing to the rest of the results. Phylogenetic relationship refers to the relative times in the past that species shared common ancestors. So the species from the first coding sequence fragment probably have closest common ancestor with the species from the second fragment rather than from the last two sequences

fragments. Hence, this result is consistent with the phylogenetic relatedness.

Appendix A. Loading transcript and coding sequence

```
# import Biopython functions
from Bio import SeqIO

# load the whole transcipt
transcript = next(SeqIO.parse("sequence.fasta",
    "fasta"))

# load the coding sequence
coding = next(SeqIO.parse("sequence.txt",
    "fasta"))

print("Lenth of transcript:
    {}.".format(len(transcript.seq)))
print("Coding sequence is {:.4g}\% proportion of
    gene.".format(100 * len(coding.seq) /
    len(transcript.seq)))
```

Appendix B. Manipulating the protein

```
#translation of the coding sequence to
    protein/amino acid sequence
protein = coding.seq.translate()
print(protein)

#number of amino acids
print(len(protein))

protein_codons = list()
different_codons = dict()

for nucleotid in range(0, len(coding.seq), 3):
    codon =
        str(coding.seq[nucleotid:nucleotid+3])
    protein_codons.append(codon)

    if different_codons.get(codon) != None:
        different_codons[codon] += 1
    else:
        different_codons[codon] = 1

print(len(different_codons))

amino_acids = dict()

for amino_acid in protein:
    am_ac = str(amino_acid)
    print(am_ac)
    if amino_acids.get(am_ac) != None:
        amino_acids[am_ac] += 1
    else:
        amino_acids[am_ac] = 1

max_appearance, max_amino_acid = 0, ''

for key, value in amino_acids.items():
    if value > max_appearance:
        max_appearance = value
        max_amino_acid = key

print("The amino acid {0} had the maximum
    appearnaces of {1}.".format(max_amino_acid,
    max_appearance))
```

Appendix C. Plots for frequency of each codon

```
import matplotlib.pyplot as plt

dict1 = dict(list(different_codons.items())[
    :len(different_codons)//4])
dict2 = dict(list(different_codons.items())[
    len(different_codons)//4:
    len(different_codons)//2])
dict3 = dict(list(different_codons.items())[
    len(different_codons)//2:
    3*len(different_codons)//4])
dict4 = dict(list(different_codons.items())[
    3*len(different_codons)//4:])

def plotBars(d, i):
    plt.clf()
    fig = plt.figure()
    plt.bar(range(len(d)), d.values(),
        align='center')
    plt.xticks(range(len(d)), d.keys())
    plt.show()
    fig.savefig('fig{}.png'.format(i))
    plt.close()

plotBars(dict1, 1)
plotBars(dict2, 2)
plotBars(dict3, 3)
plotBars(dict4, 4)
```

Appendix D. BLAST Sequences

```
from Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML

def findGene(sequence):
    database = 'refseq_rna'
    result_handle = NCBIWWW.qblast("blastn",
        database, sequence)

    # Parse the retuned structure
    blast_records = NCBIXML.parse(result_handle)

    # take the first record (we only did one
        search, so there is only one)
    item = next(blast_records)

    E_VALUE_THRESH = 1e-09

    for alignment in item.alignments:
        for hsp in alignment.hsps:
            if hsp.expect < E_VALUE_THRESH:
                print('****************************')
                print('sequence:', alignment.title)
                print('length:', alignment.length)
                print('e value:', hsp.expect)

findGene('CTGAAGCGGGAGGCTGAGACGCTGCGGGAGCGGGAGGGC')
findGene('CTCAAGCGTGAGGCCGAGACCCTACGGGAGCGGGAAGGC')
findGene('GAAGAGCTGAAGAGAGAGGCTGACAATTTAAAGGACAGA')
```

```
findGene('AACGAGGAGCTCAAGCGAGAAGCTGATACGCTGAAGGAC')
```

References

[1] Mark H Greene, Christian P Kratz, Phuong L Mai, Christine Mueller, June A Peters, Gennady Bratslavsky, Alex Ling, Peter M Choyke, Ahalya Premkumar, Janet Bracci, Rissah J Watkins, Mary Lou McMaster, Larissa A Korde *"Familial testicular germ cell tumors in adults: 2010 summary of genetic risk factors and clinical phenotype"*
PMC US National Library of Medicine National Institutes of Health, 2010
doi: 10.1677/ERC-09-0254

[2] Landero-Huerta DA, Vigueras-Villasenor RM, Yokoyama-Rebollar E, Arechaga-Ocampo E, Rojas-Castaneda JC, Jimenez-Trejo F, Chavez-Saldana M *"Epigenetic and risk factors of testicular germ cell tumors: a brief review"*
Frontiers In Bioscience, Landmark, 22, 1073-1098, March 1, 2017

[3] Kevin Litchfield, Max Levy, Robert A. Huddart, Janet Shipley & Clare Turnbull *"The genomic landscape of testicular germ cell tumours: from susceptibility to treatment"*
Nature Reviews Urology
doi:10.1038/nrurol.2016.107

[4] *"www.ncbi.nlm.nih.gov/gene/4254"*
Accessed: October 20, 2017

[5] *"www.ncbi.nlm.nih.gov/homologene/692"*
Accessed: October 20, 2017

[6] *"www.kazusa.or.jp/codon/"*
Accessed: October 21, 2017

[7] *"http://bioinformaticnotes.com/Needle-Water/"*
Accessed: October 22, 2017

[8] *"https://www.ncbi.nlm.nih.gov/homologene/37988"*
Accessed: October 22, 2017