# Bioinformatics Lab 1

Simon Jorstedt and Max Björklund

2024-11-18

## Question 1.1

Let the proportion of allele A in the population be $P_0(A) = p$

Let the proportion of allele a in the population be $P_0(a) = q$

The probability of an offspring AA in the first generation should be $P_1(AA) = p \cdot p = p^2$

The probability of an offspring aa in the first generation should be $P_1(aa) = q \cdot q = q^2$

The probability of an offspring Aa in the first generation should be $P_1(Aa) + P_1(aA) = p \cdot q + q \cdot p = 2pq$

The proportion of A in the first generation is therefore $P_1(A) = P_1(AA) + \frac{1}{2}P_1(Aa) = p^2 + pq = p(p+q) = p$

The proportion of A in the first generation is therefore $P_1(a) = P_1(aa) + \frac{1}{2}P_1(Aa) = q^2 + pq = p(p+q) = q$

Since $P_0(A) = P_1(A) = p$ and $P_0(a) = P_1(a) = q$ , the allelle proportion are the same between generation 0 and generation 1. This will also be true for every following generation, hence Hardy-Weinberg equlibrium is reached.

## Question 1.2

```
MM <- 357
MN <- 485
NN <- 158

p <- (MM*2 + MN) / 2000
q <- 1-p

chisq.test(x = c(MM, MN, NN), p = c(p**2, 2*p*q, q**2))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(MM, MN, NN)
## X-squared = 0.099938, df = 2, p-value = 0.9513
```

Hence, it appears as though the population in a state of Hardy-Weinberg equilibrium. Chi-square-goodness-of-fit shows that we cannot reject the null-hypothesis that the population allelle distribution is as expected.

## Question 2.1

The organism in question is a Branchipus schaefferi, more commonly known as a fairy shrimp. The protein product of the CDS is named **cytochrome c oxidase subunit I**

## Question 2.2

```
# Protein Sequences

Transeq <- "LLGDDQLYNVIVTAHAFVMIFFMVMPILIGGFGNWLVPLMLGAPDMAFPRLNNLSFWMLPPSLTLLVASSMVESGVGTGWTVYPPLSAAIAH
GenBank <- "LLGDDQLYNVIVTAHAFVMIFFMVMPILIGGFGNWLVPLMLGAPDMAFPRLNNLSFWMLPPSLTLLVASSMVESGVGTGWTVYPPLSAAIAH

# For printout
# Transeq <- "LLGDDQLYNVIVTAHAFVMIFFMVMPILIGG......PVLAX"
# GenBank <- "LLGDDQLYNVIVTAHAFVMIFFMVMPILIGG......PVLA"


# DNA Sequences

GenBankOrigin <- "TCTCCTAGGAGATGACCAACTTTATAACGTCATTGTTACTGCTCACGCTTTTGTAATGATTTTCTTCATAGTTATACCAATCCTTA

Backtranseq <- "CTGCTGGGCGATGATCAGCTGTACAACGTGATCGTGACCGCCCACGCCTTCGTGATGATCTTCTTCATGGTGATGCCCATCCTGATCG

BacktranseqRevComp <- "NNNGGCCAGCACGGGCAGGCTCAGCAGCAGCAGCACGGCGGTGATCACCACGGCCCAGGCGAACAGGGGCATGCGATCCAG
```

## Question 2.3

As displayed in the code chunks where the translation results are stored, a similar result is obtained when translating the nucleotides into protein the difference is that the translated protein ends with an "X".

When translating back, we get a very different result. Reversing-complementing the sequence does not seem to result in a similar sequence either.

```
Backtranambig <- "YTNYTNGGNGAYGAYCARYTNTAYAAYGTNATYGTNACNGCNCAYGCNTTYGTNATRATYTTYTTYATRGTNATRCCNATYYTNAT

# Codon Table 5
translate <- list()
translate[["A"]] <- "A"
translate[["T"]] <- "T"
translate[["G"]] <- "G"
translate[["C"]] <- "C"
translate[["Y"]] <- c("C","T")
translate[["R"]] <- c("A","G")
translate[["S"]] <- c("G","C")
translate[["W"]] <- c("A","T")
translate[["K"]] <- c("T","G")
translate[["M"]] <- c("A","C")
translate[["B"]] <- c("C","G","T")
translate[["D"]] <- c("A","G","T")
translate[["H"]] <- c("A","C","T")
translate[["V"]] <- c("A","C","G")
translate[["N"]] <- c("A","C","G","T")

df <- data.frame(GenBankOrigin = NA, Backtranambig = NA, Candidates = NA, MatchLetter = NA, Match = NA)

for(i in 1:(nchar(GenBankOrigin)-1)){
  df[i,1] <- strsplit(GenBankOrigin, "")[[1]][i]
```

```
  df$Backtranambig[i] <- strsplit(Backtranambig, "")[[1]][i]
  df$Candidates[i] <- paste0(translate[[df$Backtranambig[i]]], collapse = ",")
  df$Match[i] <- df$GenBankOrigin[i] %in%  translate[[df$Backtranambig[i]]]
  if(df$Match[i]) df$MatchLetter[i] <- df$GenBankOrigin[i]
}

head(df,25)
```

```
##    GenBankOrigin Backtranambig Candidates MatchLetter Match
## 1              T             Y        C,T           T  TRUE
## 2              C             T          T        <NA> FALSE
## 3              T             N    A,C,G,T           T  TRUE
## 4              C             Y        C,T           C  TRUE
## 5              C             T          T        <NA> FALSE
## 6              T             N    A,C,G,T           T  TRUE
## 7              A             G          G        <NA> FALSE
## 8              G             G          G           G  TRUE
## 9              G             N    A,C,G,T           G  TRUE
## 10             A             G          G        <NA> FALSE
## 11             G             A          A        <NA> FALSE
## 12             A             Y        C,T        <NA> FALSE
## 13             T             G          G        <NA> FALSE
## 14             G             A          A        <NA> FALSE
## 15             A             Y        C,T        <NA> FALSE
## 16             C             C          C           C  TRUE
## 17             C             A          A        <NA> FALSE
## 18             A             R        A,G           A  TRUE
## 19             A             Y        C,T        <NA> FALSE
## 20             C             T          T        <NA> FALSE
## 21             T             N    A,C,G,T           T  TRUE
## 22             T             T          T           T  TRUE
## 23             T             A          A        <NA> FALSE
## 24             A             Y        C,T        <NA> FALSE
## 25             T             A          A        <NA> FALSE
```

Backtranambig gives as a sequence to test for compability when ambiguity is suspected. Ambiguous translations can occur because several codons can translate into the same amino acid. The sequences displayed above does not seem to be compatible as the bases does not match.

This is likely caused by the fact that GenBankOrigin is set to start at the base of index 2, not 1. Below is the code rewritten with a shift +1 to the index.

```
df <- data.frame(GenBankOrigin = NA, Backtranambig = NA, Candidates = NA, MatchLetter = NA, Match = NA)

for(i in 1:(nchar(GenBankOrigin)-1)){
  df[i,1] <- strsplit(GenBankOrigin, "")[[1]][i+1] # +1 here
  df$Backtranambig[i] <- strsplit(Backtranambig, "")[[1]][i]
  df$Candidates[i] <- paste0(translate[[df$Backtranambig[i]]], collapse = ",")
  df$Match[i] <- df$GenBankOrigin[i] %in%  translate[[df$Backtranambig[i]]]
  if(df$Match[i]) df$MatchLetter[i] <- df$GenBankOrigin[i]
}

head(df,25)
```

```
##    GenBankOrigin Backtranambig Candidates MatchLetter Match
## 1              C             Y        C,T           C  TRUE
## 2              T             T          T           T  TRUE
## 3              C             N    A,C,G,T           C  TRUE
## 4              C             Y        C,T           C  TRUE
## 5              T             T          T           T  TRUE
## 6              A             N    A,C,G,T           A  TRUE
## 7              G             G          G           G  TRUE
## 8              G             G          G           G  TRUE
## 9              A             N    A,C,G,T           A  TRUE
## 10             G             G          G           G  TRUE
## 11             A             A          A           A  TRUE
## 12             T             Y        C,T           T  TRUE
## 13             G             G          G           G  TRUE
## 14             A             A          A           A  TRUE
## 15             C             Y        C,T           C  TRUE
## 16             C             C          C           C  TRUE
## 17             A             A          A           A  TRUE
## 18             A             R        A,G           A  TRUE
## 19             C             Y        C,T           C  TRUE
## 20             T             T          T           T  TRUE
## 21             T             N    A,C,G,T           T  TRUE
## 22             T             T          T           T  TRUE
## 23             A             A          A           A  TRUE
## 24             T             Y        C,T           T  TRUE
## 25             A             A          A           A  TRUE
```

Now, we can see that our sequences are compatible. But up until now, only the GenBank-origin sequence has been tested for compability with itself.

Below, the backtranslated (Backtranseq) should be checked for compability with the GenBank-origin.

```
df <- data.frame(Backtranseq = NA, Backtranambig = NA, Candidates = NA, MatchLetter = NA, Match = NA)

for(i in 1:(nchar(GenBankOrigin)-1)){
  df[i,1] <- strsplit(Backtranseq, "")[[1]][i]
  df$Backtranambig[i] <- strsplit(Backtranambig, "")[[1]][i]
  df$Candidates[i] <- paste0(translate[[df$Backtranambig[i]]], collapse = ",")
  df$Match[i] <- df$Backtranseq[i] %in%  translate[[df$Backtranambig[i]]]
  if(df$Match[i]) df$MatchLetter[i] <- df$Backtranseq[i]
}

head(df,25)
```

```
##    Backtranseq Backtranambig Candidates MatchLetter Match
## 1            C             Y        C,T           C  TRUE
## 2            T             T          T           T  TRUE
## 3            G             N    A,C,G,T           G  TRUE
## 4            C             Y        C,T           C  TRUE
## 5            T             T          T           T  TRUE
## 6            G             N    A,C,G,T           G  TRUE
## 7            G             G          G           G  TRUE
## 8            G             G          G           G  TRUE
## 9            C             N    A,C,G,T           C  TRUE
```

```
## 10          G           G           G           G  TRUE
## 11          A           A           A           A  TRUE
## 12          T           Y         C,T           T  TRUE
## 13          G           G           G           G  TRUE
## 14          A           A           A           A  TRUE
## 15          T           Y         C,T           T  TRUE
## 16          C           C           C           C  TRUE
## 17          A           A           A           A  TRUE
## 18          G           R         A,G           G  TRUE
## 19          C           Y         C,T           C  TRUE
## 20          T           T           T           T  TRUE
## 21          G           N     A,C,G,T           G  TRUE
## 22          T           T           T           T  TRUE
## 23          A           A           A           A  TRUE
## 24          C           Y         C,T           C  TRUE
## 25          A           A           A           A  TRUE
```

This shows that the Backtranslated sequences does seem inaccurate because of ambiguous translations, although it is proven compatible. For example, the first 9 (10 because of +1 shift) bases of the GenBank-sequence ($T$) CTC CTA GGA translates to the same amino acids as the backtranslated sequence CTG CTG GGC.

The translational ambiguousess is one-way, hence the translation from DNA to Protein is seemingly more accurate than Protein to DNA. It is also important to translate using correct index shift and correct codon table.

# Q 3

## Q 3.1

*Read up on C. elegans and in a few sentences describe it and why it is such an important organism for the scienti c community.*

**Caenorhabditis elegans** is a type of transparent worm, about 1mm in length. It has been extensively studied, and in 1998 it became the first animal for which the entire genome is sequenced. It is frequently used as a model organism in part because it is one of the simplest organisms with a nervous system.

## Q 3.2

*Use the nucleotide BLAST tool to construct a schematic diagram that shows the arrangement of introns and exons in the genomic sequence. In the BLAST tool, choose database RefSeq Genome Database and remember that the species source of the genomic sequence is Caenorhabditis elegans. Use the GenomeDataViewer button. Alternatively you may use https://wormbase.org/tools/blast_blat.*

We now use the BLAST tool to create a diagram over the arrangement of exons and introns in the genomic sequence that most matches one that is given to us, corresponding to a particular gene. The gene that most matches the provided is called "ced-9". The diagram is available at **this link**. The link tends to expire, so here is an image of the result:

## Q 3.3

*How are the sequences numbered in the alignment (i.e., pairing of query and database sequences)? Are the directions the same? What would happen if you reverse complement your query sequence*

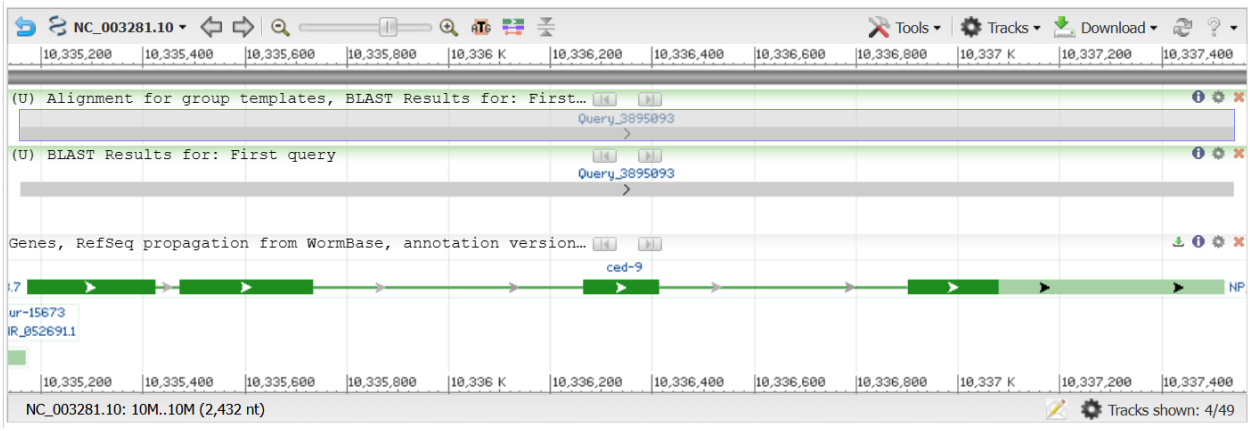


Figure 1: Diagram

The sequences appear to be numbered relative to the whole chromosome sequence on which the gene is found. The directions are the same. With a reverse complemented query sequence, we still obtain a match on the same gene, although the diagram indicates that the direction is opposite to the gene direction.

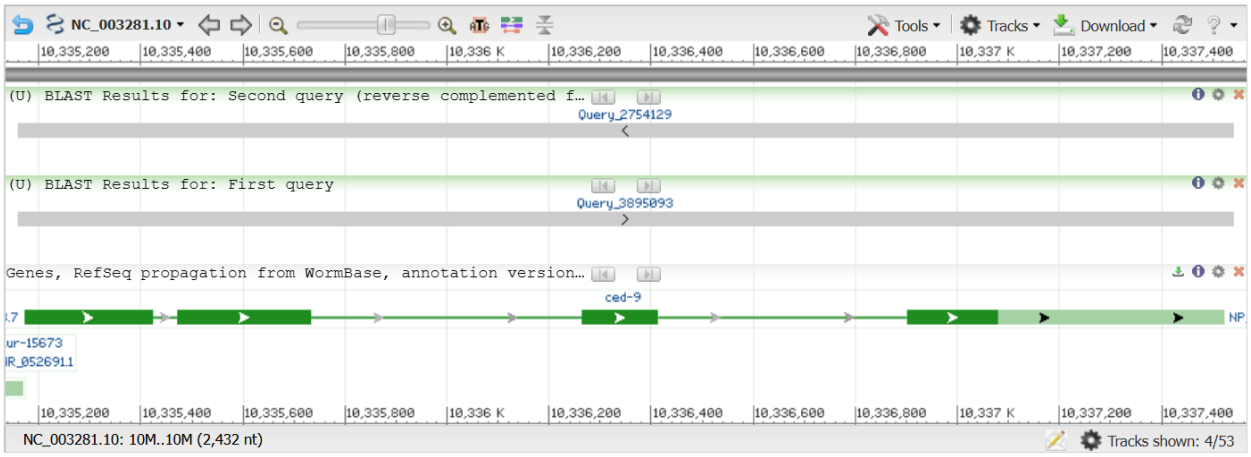


Figure 2: Diagram

## Q 3.4

*On what chromosome and what position is the query sequence found? At which position does the gene begin and end in your query sequence?*

The query sequence is found from position 10,335,160, to position 10,337,540 on the Chr III chromosome (NC_003281.10). The gene begins and ends at positions 10,335,172 and 10,337,522 respectively. Thus the query extends somewhat beyond the start and end of the gene.

## Q 3.5

*Extract the DNA code of each exon and using transeq find the protein code of the gene. You can also use blastx or blast_blat to obtain protein sequences. How do they compare to your translation?*

There are four exons in the gene, and using the transeq tool, we obtain their amino acid sequences (the proteins). We can then apparently simply append them, since the exons have the same directions. The protein code of the gene is thus:

MTRCTADNSLTNPAYRRRTMATGEMKEFLGIKGTEPTDFGINSDAQDLPSPSRQASTRRM

SIGESIDGKINDWEEPRLDIEGFV

VDYFTHRIRQNGMEWFGAPGLPCGVQPEHEMMRVMGTIFEKKHAENFETFCEQLLAVPRI

SFSLYQDVVRTVGNAQTDQCPMSYGRL

IGLISFGGFVAAKMMESVELQGQVRNLFVYTSLFIKTRIRNNWKEHNRSW

DDFMTLGKQMKEDYERAEAEKVGRRKQNRRWSMIGAGVTAGAIGIVGVVVCGRMMFSLK*
RIQFV*IINLCTTPYI*ISFLLTDSLIL*TGRSGKARPQITALCVDLRFYCNFFRLPFFL AKPYFRVISTF-
PCSVHFVKNPETLTFLAVA*PPASLPHFQSTPVSQ*FIFTLTVSFRVAS SNSPQIPVRVRDFVFIFFKLF-
SLYNNKX

When performing the search with blastx, we obtain a protein that seems to correspond to only the first exons protein sequence.

## Q 3.6

*What gene is in the query sequence? Hovering over an exon you should see links to View GeneID and View WormBase. These point to pages with more information on the gene. Follow them and write a few sentences about the gene.*

The gene is called ced-9. According to Wormbase, it enables GTPase activator activity and protein sequestering activity. There are Human ortholog genes that are involved in Alzheimers's disease, B-cell lymphoma, and other diseases.