Building and Testing a Fuzzy, Contextual Gene Finder for Olfactory Receptor Genes

This assignment is to design, implement, and run an experiment with a gene finding program that searches for olfactory receptor genes.

FAQs

1. **What are olfactory receptor genes?** These are genes found in large numbers in all animals and that code for receptors for particular odor molecules. The receptors (usually situated in the nose) are used combinatorically to help us (and other animals) to distinguish particular complex smells. For example, dogs have many more of these genes than humans, and therefore, dogs detect a wider range of odors. However, humans have many pseudogenes (mutated, non-functional genes) for olfaction dating from a time when our very distant ancestors were more olfactory, less upright, and less visual.

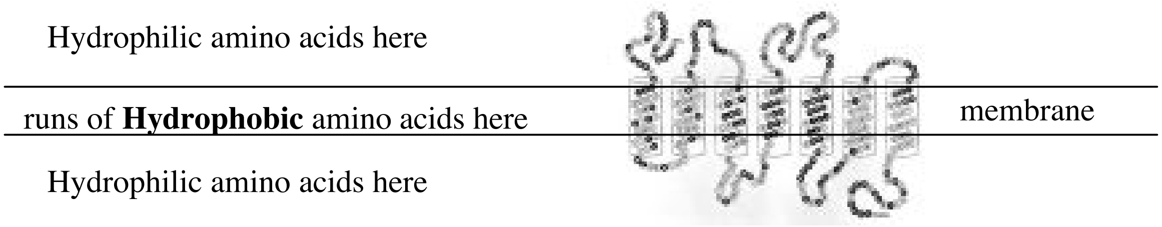
2. **What allows the specific binding interaction between a receptor and its odor?** This is still somewhat of a mystery. It appears that different parts of each receptor can bind to particular odorant molecules. Furthermore we (humans) have 100s of different receptors. Used in various combinations, our 100s of receptors can detect thousands of odors.

3. **Why do we need an olfactory gene finder? Haven't all the olfactory genes been found?** All of the olfactory genes have not been found because there is such a wide range of them and the "rules" are still being worked out for what exactly constitutes an olfactory gene. Actually, GENE FINDERS in general are difficult to build. Needless to say, a perfect gene finder has yet to be built!

4. **Why FUZZY?** "Fuzzy" implies a built-in flexibility that allows for a range of "spellings". For example, a "fuzzy" word finder would allow you to spell "elephant" in several other recognizable ways: elefant, elefint, elephint, etc.

> **fuzzy**, *a.* **c.** *Computing* and *Logic*. (Of a set) defined so as to allow for imprecise membership criteria and for gradations of membership; pertaining or belonging to such a set; ***fuzzy logic***, the logic of fuzzy sets and fuzzy concepts; ***fuzzy matching***, (the facility for) matching items which are similar but not identical. (Oxford English Dictionary [OED], http://dictionary.oed.com)

5. **Why CONTEXTUAL?** "Contextual" implies a requirement that the finder recognize parts of a gene only if other parts are also present. The example below of "Seven Transmembrane Proteins" suggests that we are looking for seven groups of consecutive hydrophobic ("water hating") amino acids separated by groups of hydrophilic ("water loving") amino acids. Each group of (or run of consecutive) **hydrophobic** amino acids are the amino acids found **within** the membrane. The number of hydrophobic amino acids within each group varies.


Hydrophilic amino acids here
runs of **Hydrophobic** amino acids here
membrane
Hydrophilic amino acids here

6. **What PRACTICAL information and research will be needed to construct an olfactory gene finder?**
Olfactory Receptors (ORs) are positioned in the membranes of cells (usually in noses) and look something like pieces of string that have been woven through the membrane SEVEN TIMES. Olfactory receptors are a type of "Seven Transmembrane Protein". The parts of the protein that are embedded in the membrane are comprised of some significant number (or FREQUENCY) of HYDROPHOBIC amino acids. The protein has SEVEN such hydrophobic sections. The SPACES between the hydrophobic units are also significant.

Amino acids are coded for by **TRIPLETS** of nucleotides (A,T,C,G) which are often represented as SYNONYMS. That is, there may be more than one DNA triplet for each amino acid. See the [triplet-nucleotide to amino acid] mapping on the last page of this specification. However for this program, **you must use the SINGLE letter code**, which is:

Glycine: (Gly) G
Proline: (Pro) P
Alanine: (Ala) **A***
Valine: (Val) **V***
Leucine: (Leu) **L***
Isoleucine: (Ile) **I***
Methionine (START) (Met) **M***
Cysteine: (Cys) C
Phenylalanine: (Phe) **F***
Tyrosine: (Tyr) **Y***

Tryptophan (Trp) **W***
Histidine: (His) H
Lysine: (Lys) K
Arginine: (Arg) R
Glutamine: (Gln) G
Asparagine: (Asn) N
Glutamic Acid: (Glu) E
Aspartic Acid: (Asp) D
Serine: (Ser) S
Threonine: (Thr) T

* means that the amino acid is relatively hydrophobic (water hating). A mnemonic aid to remember the amino acids that are hydrophobic is: "**VW FAMILY**", which uses all of the letters of their single-letter codes. We also like: "**FM WAVILY**".

(*Note: we found these "anagrams" using our regular expression anagram program, but of course!*)

INPUT:
A FASTA formatted file containing a DNA gene sequence from a vertebrate (e.g., human, mouse, fish, frog, bird) obtained from NCBI. (See your associated BLAST lab for help locating a good gene to use).

Like all FASTA formatted files, the first line of the file should begin with a **>** and contain a description of the file, e.g.

```
> Homo sapiens similar to seven transmembrane protein.
```

The DNA gene sequence *must* start with ATG and end with a STOP codon.

OUTPUT: (see sample output for further clarification)
0. The Title of your report, including a cool name for your program
1. The organism name; it should be a vertebrate (e.g., human, mouse, fish, frog, bird, etc)
2. The accession number, name of the sequence, and short definition of "what" it is from
3. The entire DNA sequence, 70 nucleotides per line
4. The associated translated Amino Acid sequence, 40 amino acids per line
5. A readable (helpful) description in English of your regular expression that you used
6. Print the actual regEx that you use in your program to locate hydrophobic regions.

7. Report each of the hydrophobic regions that you find with your regEx;
for example, for each hydrophobic region, include the information shown below:

```
** FOUND(1) of length   5 at location 10:   MFVLL
** FOUND(2) of length  11 at location 25:   FLFLLFLLVYV
** FOUND(3) of length   5 at location 42:   LLIMV
```

etc etc to (approximately) seven (Note: you may not find exactly seven; this is *fuzzy!*).

A section that prepares for and uses your RANDOM DNA sequence.

8. The number of As, Cs, Gs, and Ts in the original DNA sequence.
9. The percentage of As, Cs, Gs, and Ts in the DNA sequence (each percentage printed with two places after the decimal point). Note: take care to account for possible nucleotides that are not A, C, G, or T.

10. The randomized DNA sequence, 70 nucleotides per line
11. The translation of the randomized sequence (that is, the amino acids), 40 AAs per line
12. Report each of the hydrophobic regions that you find with your regEx; for example:

```
** FOUND(1) of length 5 at location 132:  AILVL
** FOUND(2) of length 6 at location 183:  YALAAV
```

Note: You will be working on adjusting your regEx to get the longest possible hydrophobic strings that come in multiples of (about) seven. If you get too many hydrophobic groupings, change your regEx to look for groups that are longer; if your regex is producing too few groups (< 7), look for shorter groups.

This is not just trial and error. You know what the hydrophobic amino acids are. Print your sequence, circle, and count them! Once you settle on an acceptable **range of lengths**, run your program one more time and save your output.

DETAILS:

(0) Your program must include the following **subroutines**. As indicated below, we have written many of these subroutines for you. Your starter kit contains those Perl subroutines. Other subroutines are partially written and need to be completed by you, while you will have to write some of the others from scratch. We have listed the subroutines in the order that they are called when you are working on the DNA sequence. Except for calls #1, #2, and #3, you will repeat the calls #4 through #10 after you generate and are handling your randomly generated sequence.

You write (some of or all of)	Done for you
	`(1) getAminoAcidLookUpTable()`
	`(2) getDNA($filename)`
`(3) printTitle(…`	
`(4) printSequence($DNAsequence, $NUC_WIDTH)`	
	`(5) translate($DNAsequence)`
`(6) printSequence($AAsequence, $AA_WIDTH)`	
`(7) countNucleotides($DNAsequence, "A")`	
`(8) countNucleotides($DNAsequence, "C")`	
`(9) countNucleotides($DNAsequence, "G")`	
`(10) countNucleotides($DNAsequence, "T")`	

`(11) generateRandomSequence($DNAsequence,`
`$percentA, $percentC,`
`$percentG, $percentT)`	

Notice that some of the subroutines are called more than one time, for example, you can use the `printSequence` subroutine to print four different sequences (DNA and its associated AA sequence and the random and its associated AA sequence). You need not write four different subroutines! The same subroutine can handle the four different sequences, each sequence passed as an argument. This is one of the beauties of subroutines. Likewise, notice that you can call `countNucleotides` multiple times, each time changing the arguments that you send to the subroutine.

(1) Your subroutines should follow these **specifications** exactly. Although there are other ways you could write these subroutines, you must follow the directions as given here to receive full credit.

```
# printTitle: Subroutine to print the title at the top of the output
# 3 arguments:    organismName, accession number, organism definition
# RETURNS:        nothing
```

```
# printSequence:  Subroutine to print a sequence (first argument) with
#                       the (second argument) number of characters per line
# 2 arguments:    sequence to print (could be DNA, RNA, or Amino Acid sequence)
#                 and the number of characters to print per line then newline
# RETURNS:  nothing
```

```
# countNucleotides: Subroutine to COUNT and RETURN the number of nucleotides
#                      (second arg) in the given sequence (first arg)
# two arguments:  sequence to search and the specific nucleotide to count
# RETURNS: number of times the second argument appears in the sequence
#
# A sample of how you might "call" this subroutine from your program:
#                  $numberOfAs = countNucleotides("ACGTACGT", "A");
#                  print "$numberOfAs";    # --> would print two(2), two As
```

```
# generateRandomSequence: Generate a random sequence of the same length as
#                           the first argument using the A,C,G,T nucleotide
#                           percentages in the 2nd, 3rd, 4th, and 5th args.
# 5 arguments:  the original sequence (could be DNA, AA, random, etc),
#                              and %As, %Cs, %Gs, and %Ts
# RETURNS:      random sequence
```

(2) If the getDNA subroutine returns an empty DNA sequence, your program should "trap" this, report a warning message to the user, and exit.

(3) Remember, to generate a random number in Perl from 1 to 100:

```
my $number;
$number = rand(100) + 1;
```

(4) Remember, to concatenate (`.`) a (randomly generated) letter onto the end of a sequence:

```
$randomSequence = $randomSequence . "A";  # add on an "A" nucl.
```

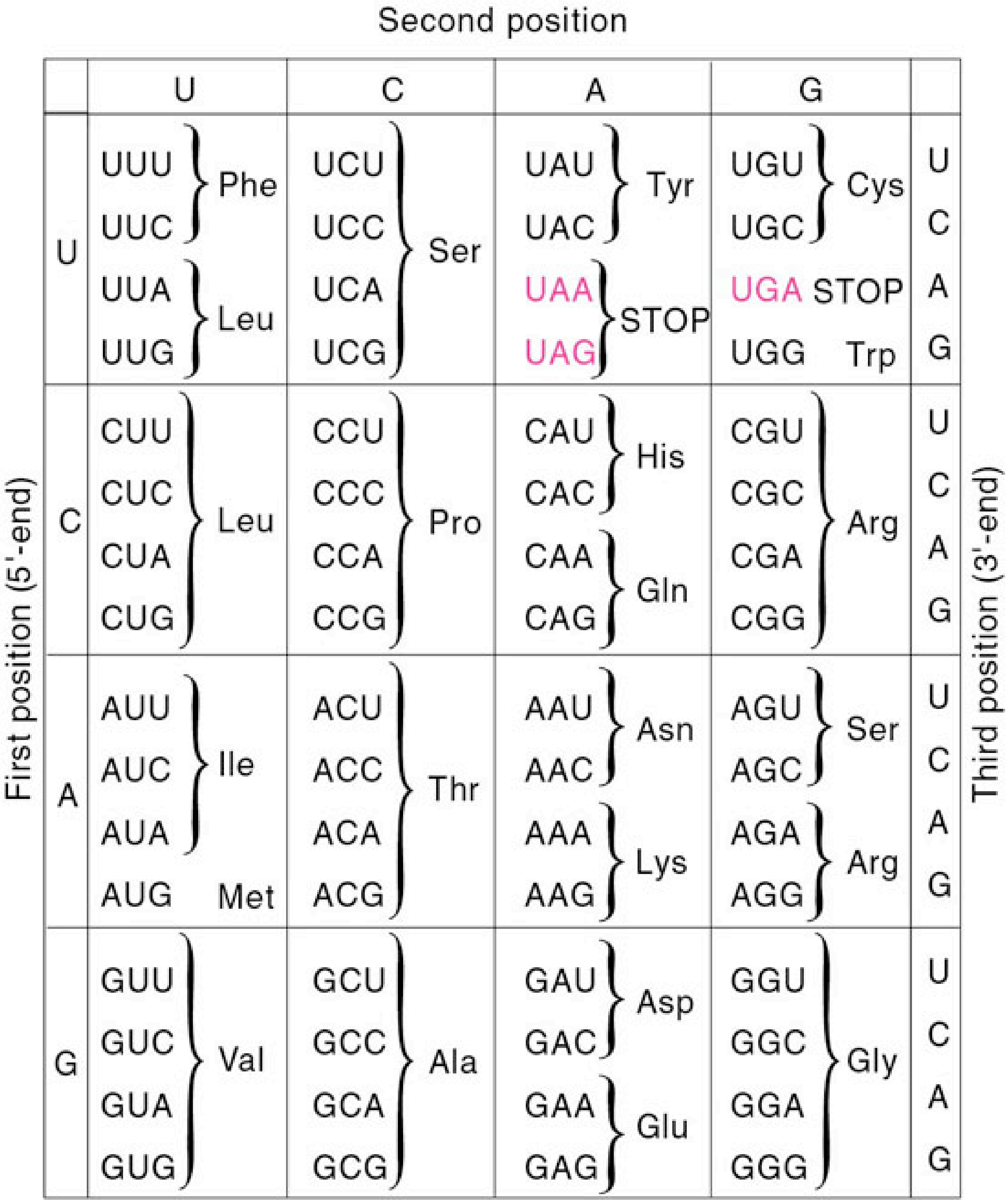
(5) HINT: There is no "correct" range for the number of amino acids within each group of hydrophobic amino acids. There may be six(6) amino acids in one group (e.g., the first group passing through the membrane) and there may be fifteen(15) amino acids in another group (e.g., the second group passing though the membrane). You should *not* expect each of the seven groups to be the same length.

(6) If your program does *not* find any groups of hydrophobic amino acids (especially possible in your random sequence), your program must print a message to the effect: `"No hydrophobic amino acids found with the given range."`

(7) You may be wondering: "Hey, what about the transcription step that produces the mRNA?" We are assuming the starting DNA sequence *is* the known coding sequence (including the correct reading frame, beginning with a start codon, and ending with the stop codon). So, the starting DNA sequence *is* the gene, ready to be translated into amino acids.

Sample output:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    OLF: Olfactory Locator Finder

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ORGANISM: Human
ACCESSION NUMBER: XM 000000
DEFINITION: Homo sapiens similar to seven transmembrane helix receptor
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

GENE DNA SEQUENCE:
======================================================================
ATGGAACCACAGAACA…

GENE Amino Acid SEQUENCE:
======================================================================
MEPQNTTQVSMFVLL…

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Describe your regEx in English HERE ....

Print your regEx HERE

Now looking for HYDROPHOBIC Amino Acids ....

** FOUND(1) of length  5 at location  10:        MFVLL
** FOUND(2) of length 11 at location  25:        FLFLLFLLVYV
** FOUND(3) of length  5 at location  42:        LLIMV
** FOUND(4) of length  5 at location  58:        MYFLL
** FOUND(5) of length  7 at location 143:        LVVAAWV
** FOUND(6) of length  5 at location 159:        LALIL
** FOUND(7) of length 12 at location 204:        LLVIIWFLLLLI
** FOUND(8) of length  6 at location 219:        VILVML
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



RANDOM DNA SEQUENCE of same length and same A,C,G,T ratios as the test
sequence:
======================================================================
CTCTTGCAGGTTCT…

RANDOM Amino Acid SEQUENCE:
======================================================================
LLQVLCFPARPVY…

Now looking for HYDROPHOBIC Amino Acids ....

** FOUND(1) of length 5 at location 87:    FFVVM
** FOUND(2) of length 5 at location 99:    VLLLM
```



Second position

First position (5'-end)	U		C		A		G		Third position (3'-end)
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leu	UCA		UAA	STOP	UGA	STOP	A
	UUG		UCG		UAG		UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

NOTE: Each nucleic acid contains four types of base. The same two purines, adenine (A) and guanine (G), are present in both DNA and RNA. The two pyrimidines in DNA are cytosine (C) and thymine (T); **in RNA, uracil (U) is found instead of thymine (T)**.