Project 2 (Bioinformatics Programing Course)

Table of Contents

Problem definition:

Input:

A protein sequence accession number (or ID). You can choose whether you use swiss or genebank records.

Problem:

Using id or accession number, download a protein sequence object and find out from its features and annotations it's variants. Produce a file containing all variant sequences in FASTA format. This means, that you need to produce each mutation one by one to the original sequence. On title line of each fasta record should be mentioned the change. For example the Swiss-Prot sequence btk_human contains more than 100 variant features. Only few sequences in Swiss-Prot contain variant features.

Output:

Mutated fasta records

Solution principle:

The solution is straightforward, the user should pass some basic info to the software and the software should handle all the procedures until the end, and produce all the variants of that protein in a FASTA file. There are three things the software needs to operate, which are the user's email address for sending to Entrez, the accession number for the protein that the user wants the variants for, and finally the FASTA file name which the user prefers to have results in.

The solution is to query the accession number to the Entrez and get the protein information. These information can be found in UniProtKB which is more eye-friendly. After getting the whole information, the software tries to go through the features of this protein and extract the mutations or possible variants which are already mentioned in the qualifiers section of this protein features. Not all the qualifiers are mutation, so the software should be able to handle other info and only take action on the appropriate records. Then the software apply the mutation regarding of the location and aminoacid change mentioned in the record, and then will start to collect all these variants in appropriate format and later it will write all these data into a FASTA file. After gathering all the variants in appropriate format, the software will write all the collected variants in a single FASTA file and then terminates running the program.

Instructions:

This software accept arguments from command line and also within it self while running, so if it finds some of the pushed arguments incorrect or incomplete, it will ask the user to insert them again.

There are three inputs this software needs to be functional:

1. Valid Email address
2. Valid accession number
3. Result's file filename

Valid email address:

The software will not check the validity of the entered email address, but it will check that the email has the correct format and if the entered email address does not meet the required details, the software will ask the suer to enter an email address in proper format.

Valid accession number:

If the query with the entered accession number fails due to the incorrect accession number, the software will ask the user again for accession number.

Result's file filename

Providing the name is obligatory, but writing the file extension is optional. If the user does not include .fasta extension at the end of the file name, the software will add it to the file name.
Also the software will check if a file with the same name already exists and if it does, user will see a caution alert about whether user wants to change the file name or wants to over-write on previous file.

The software accepts some arguments through the command line and also there is a help argument which user can use by adding either "`--help`" or "`-h`". For pushing the email address you can use "`--email`", for pushing accession number use "`--accession`" and for result's file name use "`--filename`". These parts may be left undefined and the software will ask the user to define them within the software's TUI (Text User Interface).

Here is an example to clarify the commands and arguments above:

```
$ python VariantGenerator.py --accession=Q06187 --email=sample@example.com --
filename=myResultFile
```

Which means that user is looking forward to generate all the Natural Variances for a protein with certain accession number and also willing to have the results in a file named "myResultFile.fasta".

User can also skip all the arguments above by typing the command below:

```
$ python  VariantGenerator.py
```

So some question will be asked within the software:

```
Please enter a valid email address: {for example: sample@example.com}
 >
```

```
Please Type the Accession Number of the Protein: { for example: Q06187 }
  >
```

```
lease Type the FASTA file name: { for example: results.fasta  or  result }
  >
```

After gathering all the necessary information, the software will start downloading protein file from ENTREZ and if the accession number be valid and the application finds the corresponding page, it will start processing the information of the protein and will produce mentioned natural variants.

For better user experience, the software will show a progress-bar and also present the process-completion percentage beside the progress-bar, as shown below.

```
Status  - Extracting the variants and making mutations:
     [ ####################           ]   61%
```

If the under any circumstances the software face an error such as wrong aminoacid in specified location of variation, or any other errors, it will gather the information related to those errors and will notify the user after the processing the rest of variants.

Test runs:

Results of test runs with some inputs.

The Command I used for this demonstration:

```
$ python VariantGenerator.py --accession=Q06187 --email=sample@example.com --
filename=myResultFile
```

The command-line will look like the following picture if the program successfuly produce variants and present some variation problem in the end:

```
project02 — bash — 80×24
  > m.mahmoudian@gmail.com

Please Type the FASTA file name: { for example: results.fasta  or  result }
  > test7

Status  - Checking if the specified file exists:
Caution - The file you specified exists.
            Do you want to specify a new file name? {type Y or N}
  > n

Status  - Retrieving data from Entrez service

Status  - Extracting the variants and making mutations:
     [ ##############################  ]   99%
Status  - Start writing Variants in Fasta file.

Status  - Writing Variants in Fasta file successfully completed.

Message - Below you can see the list of unprocessed qualifiers:
          -> The feature number84can not be processed.
The note of this record is: Missing (in XLA; severe). /FTId=VAR_006226.
          -> The feature number97can not be processed.
The note of this record is: Missing (in XLA). /FTId=VAR_006229.
Mehrads-MacBook-Pro:project02 mehrad$
```

The result file (myResultFile.fasta) contained all the natural variants, but in the following lines I'm going to present few lines from it:

```
>AR_006216 mutation: L -> P | location: 10:11
MAAVILESIFPKRSQQKKKTSPLNFKKRLFLLTVHKLSYYEYDFERGRRGSKKGSIDVEK
ITCVETVVPEKNPPPERQIPRRGEESSEMEQISIIERFPYPFQVVYDEGPLYVFSPTEEL
RKRWIHQLKNVIRYNSDLVQKYHPCFWIDGQYLCCSQTAKNAMGCQILENRNGSLKPGSS
HRKTKKPLPPTPEEDQILKKPLPPEPAAAPVSTSELKKVVALYDYMPMNANDLQLRKGDE
YFILEESNLPWWRARDKNGQEGYIPSNYVTEAEDSIEMYEWYSKHMTRSQAEQLLKQEGK
EGGFIVRDSSKAGKYTVSVFAKSTGDPQGVIRHYVVCSTPQSQYYLAEKHLFSTIPELIN
YHQHNSAGLISRLKYPVSQQNKNAPSTAGLGYGSWEIDPKDLTFLKELGTGQFGVVKYGK
WRGQYDVAIKMIKEGSMSEDEFIEEAKVMMNLSHEKLVQLYGVCTKQRPIFIITEYMANG
CLLNYLREMRHRFQTQQLLEMCKDVCEAMEYLESKQFLHRDLAARNCLVNDQGVVKVSDF
GLSRYVLDDEYTSSVGSKFPVRWSPPEVLMYSKFSSKSDIWAFGVLMWEIYSLGKMPYER
FTNSETAEHIAQGLRLYRPHLASEKVYTIMYSCWHEKADERPTFKILLSNILDVMDEES
>AR_006217 mutation: K -> R | location: 11:12
MAAVILESIFLRRSQQKKKTSPLNFKKRLFLLTVHKLSYYEYDFERGRRGSKKGSIDVEK
ITCVETVVPEKNPPPERQIPRRGEESSEMEQISIIERFPYPFQVVYDEGPLYVFSPTEEL
RKRWIHQLKNVIRYNSDLVQKYHPCFWIDGQYLCCSQTAKNAMGCQILENRNGSLKPGSS
HRKTKKPLPPTPEEDQILKKPLPPEPAAAPVSTSELKKVVALYDYMPMNANDLQLRKGDE
```

```
YFILEESNLPWWRARDKNGQEGYIPSNYVTEAEDSIEMYEWYSKHMTRSQAEQLLKQEGK
EGGFIVRDSSKAGKYTVSVFAKSTGDPQGVIRHYVVCSTPQSQYYLAEKHLFSTIPELIN
YHQHNSAGLISRLKYPVSQQNKNAPSTAGLGYGSWEIDPKDLTFLKELGTGQFGVVKYGK
WRGQYDVAIKMIKEGSMSEDEFIEEAKVMMNLSHEKLVQLYGVCTKQRPIFIITEYMANG
CLLNYLREMRHRFQTQQLLEMCKDVCEAMEYLESKQFLHRDLAARNCLVNDQGVVKVSDF
GLSRYVLDDEYTSSVGSKFPVRWSPPEVLMYSKFSSKSDIWAFGVLMWEIYSLGKMPYER
FTNSETAEHIAQGLRLYRPHLASEKVYTIMYSCWHEKADERPTFKILLSNILDVMDEES
>AR_006218 mutation: S -> F | location: 13:14
MAAVILESIFLKRFQQKKKTSPLNFKKRLFLLTVHKLSYYEYDFERGRRGSKKGSIDVEK
ITCVETVVPEKNPPPERQIPRRGEESSEMEQISIIERFPYPFQVVYDEGPLYVFSPTEEL
RKRWIHQLKNVIRYNSDLVQKYHPCFWIDGQYLCCSQTAKNAMGCQILENRNGSLKPGSS
HRKTKKPLPPTPEEDQILKKPLPPEPAAAPVSTSELKKVVALYDYMPMNANDLQLRKGDE
YFILEESNLPWWRARDKNGQEGYIPSNYVTEAEDSIEMYEWYSKHMTRSQAEQLLKQEGK
EGGFIVRDSSKAGKYTVSVFAKSTGDPQGVIRHYVVCSTPQSQYYLAEKHLFSTIPELIN
YHQHNSAGLISRLKYPVSQQNKNAPSTAGLGYGSWEIDPKDLTFLKELGTGQFGVVKYGK
WRGQYDVAIKMIKEGSMSEDEFIEEAKVMMNLSHEKLVQLYGVCTKQRPIFIITEYMANG
CLLNYLREMRHRFQTQQLLEMCKDVCEAMEYLESKQFLHRDLAARNCLVNDQGVVKVSDF
GLSRYVLDDEYTSSVGSKFPVRWSPPEVLMYSKFSSKSDIWAFGVLMWEIYSLGKMPYER
FTNSETAEHIAQGLRLYRPHLASEKVYTIMYSCWHEKADERPTFKILLSNILDVMDEES
```

Table of variables and subroutines

All variables and subroutines and their purpose are in these table. Names of variables and subroutines have chosen so, that the name describes their purpose. (By order of appearance)

Variable Name	Purpose
AccessionNum	Stores the accession number which the user provides.
Entrez.email	Stores the verified email address which user inserted.
fastaFileName	Stores the FASTA file name user defined.
fileOverWrite	Stores the True and False of the user desision to over-write the result file, if the file was previously created and exists.
myErrorState	Stores the True and False state of Entrez query error.
handle	Stores the response from Entrez service.
seq_record	Stores the parsed result of handle variable.
masterSeq	Stores the main protein sequence without any variation or mutation.
myVariantList	Stores all the created variants and their information.
unknownRecords	Stores the information about the records that could not b processed by the software.
FeaturesLength	Stores the number of features the Entrez returned.
myFeature	Works within iteration. Each time stores a feature for analysing.
myLocation	Works within iteration. Each time stores the location of the mutation.
myNote	Works within iteration. Each time stores the 'note' section of each feature.
myVariantSeq	Works within iteration. Each time stores the variant protein sequence.
MyVariantTitle	Works within iteration. Each time stores the created

	title for each variant to be stored in FASTA file.
myVariant	Works within iteration. Each time stores the details of the created variation to be added to myVariantList.
myPercentageProgressBar	Stores the processed percentage of creating the natural variants.
myProcessedProgressBar	Stores the Number of # for filling the progress-bar.
MyRemainedProgressBar	Stores the number of space character for filling the unprocessed section of progress-bar.
output_handle	Is the handle for the output file (also known as 'result').
i	Iteration counter.

Subroutine Name	Purpose
verifyEmail	Verifies the email address which is typed by user. The only action this function does to verify the email address is to check if it has a correct format.
printError	This function prints errors in a standardize format when the user's input has some irregularities.
WriteFASTA	It get the sequences and result file name as an input, and write sequences in FASTA format in the result file. (if the file does not exist, it will get created).
getEntrezEmail	Asks user's email address for sending to Entrez.
getAccessionNumber	Asks users to insert desired accession number.
getFastaFileName	Asks user to enter a file name for result file.
checkFileExists	Gets a file name as an input and checks if the file is already exists. It will return the result as True or False.

Source code:

Commented program code listing

```
#!/usr/bin/python

#---------[ Info of this software - Begin ]---------#
# Author: Mehrad Mahmoudian
# Report bug: m.mahmoudian@gmail.com
#
# Version Number: 0.0.6.1
# Change Log:
#      - Add progressbar
#      - Add feature to check if the result file already
#        exists, and if the user is willing to overwrite
#        or specify new file name.
#      - Some comment modification
#      - TUI modification and uniformation (adding: Status, Caution, Warning and Message)
#
#---------[ Info of this software - End ]---------#


#--------[ Imports - Begin ]--------#
import argparse  # command line argument parser
import re # regular expression
from os import path # for checking existing directory
from os import makedirs # for creating directory
from Bio import Entrez
from Bio import SeqIO
from Bio.SeqRecord import SeqRecord
import sys
from math import ceil
import time
#--------[ Imports - End ]--------#

#---------[ Functions - Begin]----------#
def verifyEmail (inpt_str): # email validation function
          if len(inpt_str) > 7:
                    if re.match("^.+\\@(\\[?)[a-zA-Z0-9\\-\\.]+\\.([a-zA-Z]{2,3}|[0-9]{1,3})
(\\]?)$", inpt_str) != None:
                              return True
                    else:
                              return False
          else:
                    return False

def printError (inpt_field_name='Input data'): # prints errors related to the action
          print "\nWarning - The",inpt_field_name,"you entered was invalid, try again !\n"


def mkdir (dir_name):  # make directory if it does not exist in the given path
          if not path.exists(dir_name):
                    makedirs(dir_name)

def writeFASTA (file_seq, file_name) :
          if len(file_seq) > 0 :
                    output_handle = open(file_name, "w")
                    SeqIO.write(file_seq, output_handle, "fasta")
                    output_handle.close()
          else:
                    print 'Internal Warning - No sequence is pushed in the writeFASTA function'

def getEntrezEmail () :
```

```
        correctEmailAddr = False
        while correctEmailAddr != True :
                EntrezEmail = raw_input("\nPlease enter a valid email address: { for example:
sample@example.com }\n  > ")
                if(verifyEmail(EntrezEmail) == True):
                        correctEmailAddr = True
                else:
                        printError('Email address')
        return EntrezEmail

def getAccessionNumber () :
        correctAccessionNum = False
        while correctAccessionNum != True :
                AccessionNum = raw_input("\nPlease Type the Accession Number of the Protein:
{ for example: Q06187 }\n  > ")
                if len(AccessionNum)>2:
                        correctAccessionNum = True
                else:
                        printError('accession number')
        return AccessionNum

def getFastaFileName () :
        correctFileName = False
        while correctFileName != True :
                fastaFileName = raw_input("\nPlease Type the FASTA file name: { for example:
results.fasta  or  result }\n  > ")
                if len(fastaFileName) > 0:  # if the input file name be not empty
                        if fastaFileName[-6:len(fastaFileName)].lower() != '.fasta':  # if
the provided file name does not contain .fasta extention, then add to it
                                fastaFileName += '.fasta'
                        correctFileName = True
                else:
                        printError('FASTA file name')
        return fastaFileName

def checkFileExists (theFile) :
        # checks if the file exists, and if exists will return True
        try:
                with open(theFile): pass
                return True
        except IOError:
                return False
#---------[ Function - End ]----------#


#---------[ Main Code - Begin ]---------#
# [ getting argument from command line ]
parser = argparse.ArgumentParser(prog='Mutator')  # program name
parser = argparse.ArgumentParser(description='This is program is designed to create the mutation
mentioned in genebank profile of a gene by getting the accession number and create a FASTA file
output as a result.')  # program description
parser.add_argument('--version', action='version', version='%(prog)s 0.0.6.1')
parser.add_argument('--accession', action='store', help='Genebank accession number')
parser.add_argument('--email', action='store', help='Users email address for sending to Entrez')
parser.add_argument('--filename', action='store', help='Desired FASTA file name. FASTA extention
is optional')

args = parser.parse_args()



# [ getting Accession Number from user ]
```

```
if args.accession is None :
          AccessionNum = getAccessionNumber ()
elif len(args.accession) < 2 :
          AccessionNum = getAccessionNumber ()
else:
          AccessionNum = args.accession


# [ getting email address from user ]
if args.email is None :
          Entrez.email = getEntrezEmail ()
elif verifyEmail(args.email) == False :
          Entrez.email = getEntrezEmail ()
else:
          Entrez.email = args.email

# [ getting FASTA file name from user ]
if args.filename is None :
          fastaFileName = getFastaFileName ()
elif len(args.filename) < 1 :
          fastaFileName = getFastaFileName ()
else:
          fastaFileName = args.filename
          if fastaFileName[-6:len(fastaFileName)].lower() != '.fasta':  # if the provided file
name does not contain .fasta extention, then add to it
                    fastaFileName += '.fasta'

fileOverWrite = ''
while fileOverWrite == '' :
          print "\nStatus  - Checking if the specified file exists:"
          if checkFileExists(fastaFileName) == False :  # check whether the file exists
                    fileOverWrite = False
                    print "            The file does not exist, we are good to go."
          else:
                    fileOverWriteDecision = raw_input("Caution - The file you specified exists.\n
Do you want to specify a new file name? {type Y or N}\n  > ")
                    if fileOverWriteDecision.lower() == "n" :
                              fileOverWrite = True
                    elif fileOverWriteDecision.lower() == "y":
                              fastaFileName = getFastaFileName ()
                    else:
                              print "Your answer should be either Y or N."




# [ retrieving one record from Entrez service ]
print "\nStatus  - Retrieving data from Entrez service"
myErrorState = True
while myErrorState == True :
          handle = ""
          try:
                    handle = Entrez.efetch(db="protein", rettype="gb", retmode="text",
id=AccessionNum)
                    myErrorState = False
          except Exception as inst:
                    myErrorState = True
                    print inst
                    print "Warning - Seems the accession number is not correct, recheck the
accession number and try again."
                    AccessionNum = getAccessionNumber ()
if myErrorState == False :
```

```
        seq_record = SeqIO.read(handle, "genbank")  # use SeqIO.read when only one Seq
        handle.close()
else:
        sys.exit("""+-----------------------------------+
| Something unexpected happened !   |
|                                   |
| Please re-run the application     |
|    and if it happened again       |
|    please report it.              |
+-----------------------------------+""")

# [ extracting the sequence ]
print "\nStatus  - Extracting the variants and making mutations:"
masterSeq = seq_record.seq

myVariantList = []  # the variable for storing the mutant sequences and the related info for each
unknownRecords = []

featuresLength = len(seq_record.features)

for i in range(featuresLength):  # go through all features to find those who are variations
        myFeature = seq_record.features[i]
        if myFeature.type == 'Region':
                if 'region_name' in myFeature.qualifiers : # if the key (region_name) be found in the dictionary
                        if myFeature.qualifiers['region_name'][0] == 'Variant' :  # if it was variation, then:
                                myLocation = myFeature.location
                                myNote = myFeature.qualifiers['note']
                                if myNote[0][1:5] == ' -> ' :  # if it be a real variant and not something like 'Missing', then:
                                        if myNote[0][0] == masterSeq[myLocation.start] :  # if the note and sequence be exact match
                                                myVariantSeq = masterSeq[0:myLocation.start] + myNote[0][5] + masterSeq[myLocation.end:len(masterSeq)]  # inserting the mutated aminoacid in the original sequence
                                                myVariantTitle = "mutation: " + myNote[0][0:6] + " | location: " + str(myLocation.start) + ":" + str(myLocation.end)
                                                myVariant = SeqRecord(myVariantSeq, id=myNote[0][-10:-1], description=myVariantTitle)
                                                myVariantList.append(myVariant)  # appending the variant note and the mutated sequence in a list to wite them in a file later.
                                        else:
                                                unknownRecords.append('The location in notes does not match the aminoacid in the sequence ! \n the feature ID is '  + i)
                                else:
                                        unknownRecords.append('The feature number' + str(i) + 'can not be processed.\nThe note of this record is: '  + myNote[0])
        # progressbar stuff
        myPercentageProgressBar = int(ceil(i*100/featuresLength))
        myProcessedProgressBar = int(ceil(myPercentageProgressBar*30/100))
        myRemainedProgressBar = 30 - myProcessedProgressBar
        time.sleep(0.01)
        sys.stdout.write("\r     [ " + myProcessedProgressBar*'#' + myRemainedProgressBar*' ' + ' ]   ' + str(myPercentageProgressBar) + "%")
        sys.stdout.flush()


# [ writing the mutated sequences in FASTA format in FASTA file ]
print "\nStatus  - Start writing Variants in Fasta file."
output_handle = open(fastaFileName, "w")
SeqIO.write(myVariantList, output_handle, "fasta")
output_handle.close()
```

```
print "\nStatus  - Writing Variants in Fasta file successfully completed."

# [ Showing unknown records in features ]
print "\nMessage - Below you can see the list of unprocessed qualifiers:"
for i in range(len(unknownRecords)) :
          print '           ->', unknownRecords[i]

#---------[ Main Code - End ]---------#
```