```
#!/usr/bin/python
import os
import sys
import glob
import string
import operator
from string import atof
from itertools import imap
from Bio import SeqIO

def hamming(str1, str2):
    assert len(str1) == len(str2)
    return sum(imap(operator.ne, str1, str2))

def rc(seq):
  complements = string.maketrans('acgtrymkbdhvACGTRYMKBDHV',
'tgcayrkmvhdbTGCAYRKMVHDB')
  rcseq = seq.translate(complements)[::-1]
  return rcseq

def translation(seq):
  dnamap = {"TTT":"F", "TTC":"F", "TTA":"L", "TTG":"L",
    "TCT":"S", "TCC":"S", "TCA":"S", "TCG":"S",
    "TAT":"Y", "TAC":"Y", "TAA":"_", "TAG":"_",
    "TGT":"C", "TGC":"C", "TGA":"_", "TGG":"W",
    "CTT":"L", "CTC":"L", "CTA":"L", "CTG":"L",
    "CCT":"P", "CCC":"P", "CCA":"P", "CCG":"P",
    "CAT":"H", "CAC":"H", "CAA":"Q", "CAG":"Q",
    "CGT":"R", "CGC":"R", "CGA":"R", "CGG":"R",
    "ATT":"I", "ATC":"I", "ATA":"I", "ATG":"M",
    "ACT":"T", "ACC":"T", "ACA":"T", "ACG":"T",
    "AAT":"N", "AAC":"N", "AAA":"K", "AAG":"K",
    "AGT":"S", "AGC":"S", "AGA":"R", "AGG":"R",
    "GTT":"V", "GTC":"V", "GTA":"V", "GTG":"V",
    "GCT":"A", "GCC":"A", "GCA":"A", "GCG":"A",
    "GAT":"D", "GAC":"D", "GAA":"E", "GAG":"E",
    "GGT":"G", "GGC":"G", "GGA":"G", "GGG":"G", "XXX":"X"}
  pep = []
  i = 0
  while i < len(seq):
    codon = seq[i:i+3]
    aa = dnamap[codon]
    pep.append(aa)
    i = i + 3
  pep = ''.join(pep)
  return pep
```

```
def callmut(roi,ref,offset):
  assert(len(roi)==len(ref))
  muts = []
  for n in range(len(ref)):
    if roi[n] != ref[n]: muts.append(ref[n]+str(n+offset)+roi[n])
  return muts

#READ IN REFERENCE SEQUENCE
refhash = {}
infile = open('Fasta/NS5A.fa')
for line in infile.xreadlines():
  if '>' in line:
    ID = line.rstrip().replace('>','')
    refhash[ID] = ''
  else:
    refhash[ID] += line.rstrip()
infile.close()

#READ IN OFFSET
offhash_bp = {}
infile = open('Fasta/offset_bp')
for line in infile.xreadlines():
  line = line.rstrip().rsplit("\t")
  offhash_bp[line[0]] = int(line[1])
infile.close()
#READ IN OFFSET
offhash_pep = {}
infile = open('Fasta/offset.csv')
for line in infile.xreadlines():
  line = line.rstrip().rsplit("\t")
  offhash_pep[line[0]] = int(line[1])
infile.close()

#READ IN BARCODE SEQUENCE
bchash = {}
infile = open('Fasta/Barcode_NS5A.fa')
for line in infile.xreadlines():
  if '>' in line:
    ID = line.rstrip().replace('>','')
  else:
    bchash[line.rstrip()] = ID
infile.close()
#print bchash

#FORMAT MUTHASH
```

```
Muthash = {}
for bc in bchash.keys():
  Muthash[bc] = {}

#MAIN - MAPPING
count = [0,0,0,0,0]
fqfiles = sorted(glob.glob('fastq/*_1_*'))
allmut  = []
for fqfile in fqfiles:
  print fqfile
  fqfile1  = fqfile #fqfile1 is the filename of the forward read file
  fqfile3  = fqfile1.replace('_1_','_3_') #fqfile3 is the filename of the
reverse read file
  fqfile2  = fqfile1.replace('_1_','_2_')
  record3s = SeqIO.parse(fqfile3, "fastq")
  record2s = SeqIO.parse(fqfile2, "fastq")
  for record1 in SeqIO.parse(fqfile1, "fastq"): #loop through the record in
forward read file
    record3 = record3s.next() #read the reverse read file in parallel
    record2 = record2s.next() #read the reverse read file in parallel
    assert(record1.id[0:-1]==record3.id[0:-1]) #make sure ID forward and
reverse matches
    seq1 = str(record1.seq) #seq1 represents sequence of forward read
    seq3 = str(record3.seq) #seq3 represents sequence of reverse read
    seq2 = str(record2.seq)[0:6] #seq3 represents sequence of reverse read
    rgn  = ''
    dirt = ''
    pos1 = ''
    pos3 = ''
    roi  = '' #a substring of the read
    mm   = ''
    bc   = seq2
#    bc1  = seq1[0:3] #bc1 represents the first 3 bases in the forward read
#    bc3  = seq3[0:3] #bc3 represents the first 3 bases in the reverse read
#    if bc1 != bc3: continue #MAKING SURE FORWARD AND REVERSE READ HAS THE
SAME BARCODE
    if bc not in bchash.keys():
      count[0]+=1
#      print bc
      continue #MAKING SURE BARCODE MATCHES ONE OF THE BARCODES BEING USED
    for ref in refhash.keys():
      stop   = ''
      refseq = refhash[ref]
#      print 'refseq',refseq
#      print 'seq1',seq1
#      print seq1
```

```
#      print 'refseqs'
#      print "\n".join(refhash.values())
      for n in range(0,len(seq1)-len(refseq)+1):
#        print seq1[n:n+len(refseq)]
#        print hamming(refseq,seq1[n:n+len(refseq)])
        if hamming(refseq,seq1[n:n+len(refseq)]) <= 5: rgn=ref; dirt='F';
pos1=n; stop='yes'; roi = seq1[n:n+len(refseq)]; break
        elif hamming(rc(refseq),seq1[n:n+len(refseq)]) <= 5: rgn=ref;
dirt='R'; pos1=n; stop='yes'; roi = seq1[n:n+len(refseq)]; break
      if stop == 'yes': break
    if roi == '':
      count[1]+=1
      continue

    if rc(roi) not in seq3:  count[2]+=1; continue #MAKING SURE THE
SEQUENCE MATCHES THAT OF THE REVERSE READ
    if dirt == 'R': roi = rc(roi)
    roipep = translation(roi)
    refpep = translation(refhash[ref])
    offset_pep = offhash_pep[ref]
    refseq=refhash[ref]
    offset_bp = offhash_bp[ref]
    muts_pep   = callmut(roipep,refpep,offset_pep) #a list of mutations
    muts   = callmut(roi,refseq,offset_bp) #a list of mutations
    if len(muts) > 1: count[3]+=1;continue #MAKING SURE ONE OR NONE
MUTATION PER READ
    elif len(muts) == 1:
       count[4]+=1
       if len(muts_pep)==0:
         mut = muts[0]+'\t'+'WT'
       else:
         mut = muts[0]+'\t'+muts_pep[0]
    else: mut = 'WT'
    mut = mut+"\t"+rgn
    if mut in Muthash[bc].keys(): Muthash[bc][mut] += 1
    else: Muthash[bc][mut] = 1
    if mut not in allmut: allmut.append(mut)
print 'done parsing'

allmut = sorted(allmut)
for bc in Muthash.keys():
  outfile = open('result_NS5A/'+bchash[bc],'w')
  for mut in allmut:
    if mut in Muthash[bc].keys(): outfile.write(mut+"\t"+str(Muthash[bc]
[mut])+"\n")
    else: outfile.write(mut+"\t"+'0'+"\n")
```

```
  outfile.close()
outfile.close()

print count
```