Jacek Radajewski (43612772)
SCIE2100 Prac 3
14/04/2014

Question 1

```
def alignGlobal(seqA, seqB, substMatrix, gap=-1):

.
.
.

    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    # starting at the bottom right hand corner of the matrix trace back to the origin 0,0
    while i > 0 and j > 0:
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        if S[i, j] == S[i - 1, j] + gap:
            alignA = seqA[i - 1] + alignA
            alignB = "-" + alignB
            i -= 1
        elif S[i, j] == S[i, j - 1] + gap:
            # Got here by a gap in sequence A (go left)
            alignA = "-" + alignA
            alignB = seqB[j - 1] + alignB
            j -= 1
        else:
            # Got here by aligning the bases (go diagonally)
            alignA = seqA[i - 1] + alignA
            alignB = seqB[j - 1] + alignB
            i -= 1
            j -= 1
    # Fill in the rest of the alignment if it begins with gaps
    # (i.e., traceback all the way to S[0, 0])
    while i > 0:
        # Go up
        alignA = seqA[i - 1] + alignA
        alignB = "-" + alignB
        i -= 1
    while j > 0:
        # Go left
        alignA = "-" + alignA
        alignB = seqB[j - 1] + alignB
        j -= 1
    alignment = Alignment([Sequence(alignA, seqA.alphabet, seqA.name, gappy=True), Sequence(alignB, seqB.alphabet, seqB.name,
gappy=True)])
    alignment.gap_penalty = gap # save the gap for future analysis
    return alignment
```

Question 2

```
def alignLocal(seqA, seqB, substMatrix, gap=-1):
.
.
.
    # Stop when we hit the beginning of at least one sequence
    # Local: also stop when we hit a score 0
    while i > 0 and j > 0 and S[i, j] > 0:
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        if S[i, j] == S[i - 1, j] + gap:
            alignA = seqA[i - 1] + alignA
            alignB = "-" + alignB
            i -= 1
        elif S[i, j] == S[i, j - 1] + gap:
            alignA = "-" + alignA
            alignB = seqB[j - 1] + alignB
            j -= 1
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        else:
            # Got here by aligning the bases (go diagonally)
            alignA = seqA[i - 1] + alignA
            alignB = seqB[j - 1] + alignB
            i -= 1
            j -= 1
    return Alignment([Sequence(alignA, seqA.alphabet, seqA.name, gappy=True), Sequence(alignB, seqB.alphabet, seqB.name,
gappy=True)])
```

Question 3

Two differences between Needleman-Wunsch global and Smith-Waterman local alignments and their corresponding python implementation differences are described below.

Negative scores

One way that Smith-Waterman local alignment is different to Needleman-Wunsch global alignment is that it rejects negative scores. This difference is implemented in the code by adding 0 to the list from which we select the maximum as shown in the code below.

Global

```
S[i, j] = max([match, delete, insert])
```

Local

```
S[i, j] = max([match, delete, insert, 0])
```

Traceback start position

Global

Needleman-Wunsch traceback starts at the bottom-right-hand corned of the matrix ie last element. The code for obtaining the traceback starting point coordinates is shown below.

```
    i = lenA
    j = lenB
```

Furthermore the following code applies only to global alignment as local alignment would never start with a gap

```
    # Fill in the rest of the alignment if it begins with gaps
    # (i.e., traceback all the way to S[0, 0])
    while i > 0:
        # Go up
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        alignA = seqA[i - 1] + alignA
        alignB = "-" + alignB
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        i = i - 1
    while j > 0:
        # Go left
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        alignA = "-" + alignA
        alignB = seqB[j - 1] + alignB
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        j = j - 1
```

Local

Unlike Needleman-Wunsch global alignment, Smith-Waterman traceback does not start at the 'end', bottom-right corner of the matrix. Instead, local alignment traceback starts at the highest score. The coordinates of the highest score are obtained by the following code:

```
i = 0
j = 0
```

```
for ii in range(1, lenA + 1):
    for jj in range(1, lenB + 1):
        if S[ii, jj] > S[i, j]:
            i = ii
            j = jj
```

Question 7

Analysis of the HQ659871.1 and JX416721.1 alignments

In order to complete questions 5, 6 & 7 of the practical I have performed local and global alignments using five different matrices and gap penalties ranginng from -1 to -19. Furthermore, the above analysis was performed for all three reading frames with a seperate output file generated for each reading frame.

The approach was to iterate over the matrices and gaps as described above and collate all of the resulting alignments so they can be displayed together, on a single HTML page, for easy comparison. This was done by generating the output of all collated alignments and transforming the result into HTML using a XSLT stylesheet. Single HTML file contained results for various matrices and gap penalties used, but seperate files were used for local and global alignments as well as different reading frames.

In order to aid the analysis of the alignments start codons (including alternative) AUG, GUG and UUG were highlighted green, stop codons UAG, UGA and UAA were highlighted blue and gaps were highlighted red

Note that although screen shots of these alignments have been provided throughout question 7 (and question 10) each screen shot contains only a small part of the set of alignments. For example it was impossible (even using browser window spanning 3 monitors) to produce a screen shot of the complete residue with the start and stop codons present in the screen. The only sensible way to view the results is to actually view the HTML files.

Below is the listing of the main part of question 7 with the rest of the supporting code in Appendix B.

```
'''
Created on 08/04/2014

@author: s4361277
question 7
'''
from sequence import *
from alignments import AlignmentCollection, AlignmentThread

HQ659871_1 = getSequence("HQ659871.1", "genbank")
JX416721_1 = getSequence("JX416721.1", "genbank")

matrix_names = ["dna.matrix", "dna.matrix2", "dna.matrix3", "dna.matrix4", "dna.matrix.unknown"]

# list of matrices though which we shall iterate to obtain various alignments
matrices = []
for matrix_name in matrix_names:
    matrix = readSubstMatrix(matrix_name, DNA_Alphabet)
    print "Adding matrix: " + matrix_name + "\n", matrix
    matrices.append(matrix)

local_alignments = AlignmentCollection("question 7 local")
global_alignments = AlignmentCollection("question 7 global")
```

```
print "number of cells in alignment matrix = ", len(HQ659871_1) * len(JX416721_1)

threads = []

print "HQ659871_1 = ", len(HQ659871_1)
print "JX416721_1 = ", len(JX416721_1)

for matrix in matrices:
    print "processing matrix:", matrix.name
    for gap_penalty in range(-1, -20, -1):
        local_alignments.add_alignment(alignLocal(HQ659871_1, JX416721_1, matrix, gap_penalty))
        global_alignments.add_alignment(alignGlobal(HQ659871_1, JX416721_1, matrix, gap_penalty))

local_alignments.dump_xml_and_html("question_7_local_alignments")
global_alignments.dump_xml_and_html("question_7_global_alignments")
```

The five matrices used in the analysis were:

dna.matrix - original supplied matrix

```
A   6
C  -5   6
G  -1  -5   6
T  -5  -1  -5   6
    A   C   G   T
```

dna.matrix2 - slight increase to the most negative scores

```
A   6
C  -4   6
G  -1  -4   6
T  -4  -1  -4   6
    A   C   G   T
```

dna.matrix3 – slignificant increase to the most negative scores

```
A   6
C  -2   6
G  -1  -2   6
T  -2  -1  -2   6
    A   C   G   T
```

matrix: dna.matrix4 - all negative scores are equal

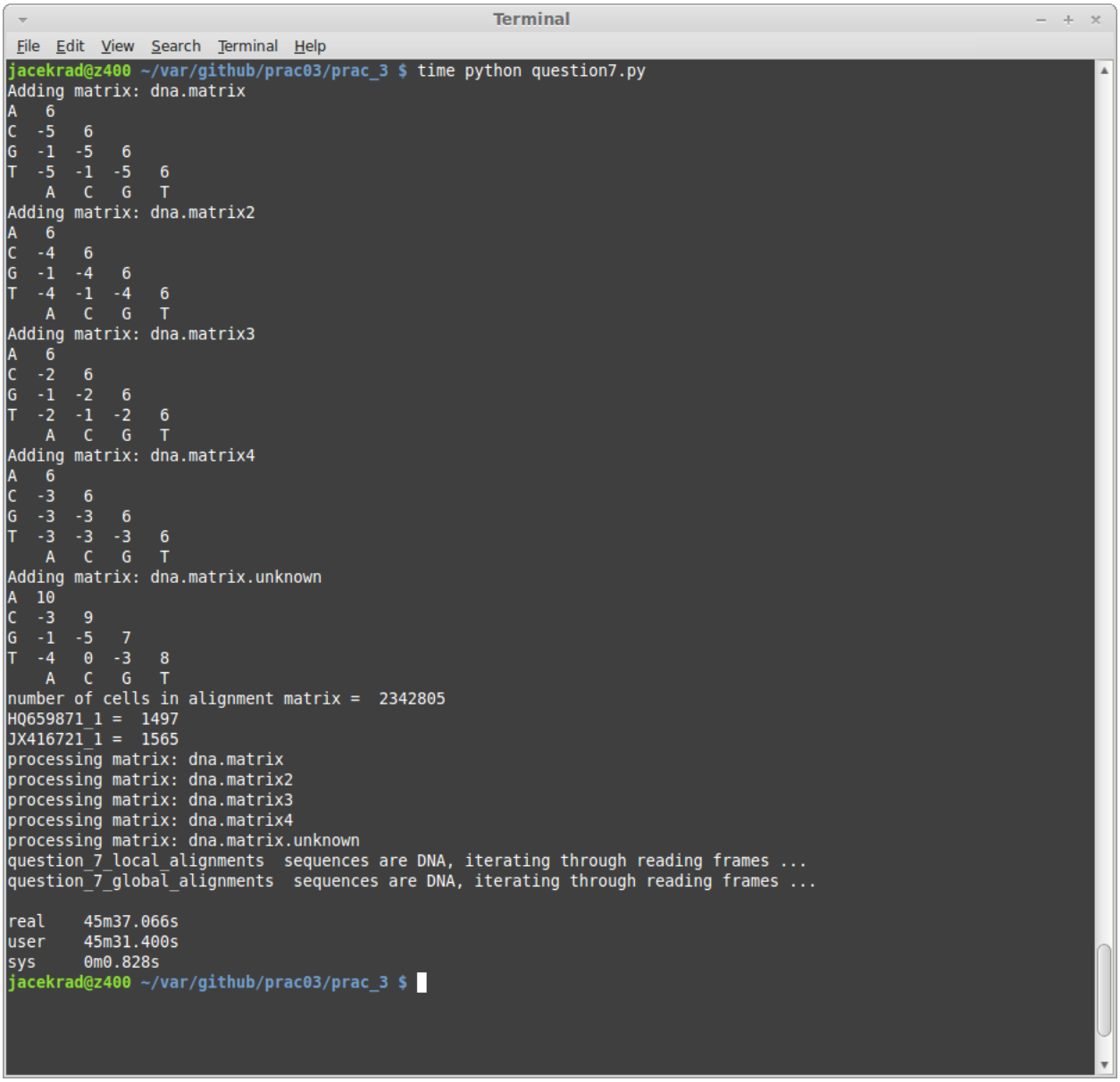
```
A   6
C  -3   6
G  -3  -3   6
T  -3  -3  -3   6
    A   C   G   T
```

dna.matrix.unknown - found on various sites including wikipedia

```
A  10
C  -3   9
G  -1  -5   7
T  -4   0  -3   8
    A   C   G   T
```

part a

In order to globally align two sequences of length *m* and *n,* a *m+1 x n+1* matrix needs to be constructed *m x n* cells evaluated using the algorithm. Our sequences have lengths: HQ659871_1: 1497 and JX416721_1: 1565 which generates a matrix with **2342805** + 1297 + 1565 + 1 cells. The size of this matrix determines the significant overall wall clock time of running the alignment. Below is a screen shot of running global and local algorithms for 5 different substitution matrices and 19 different gap penalties. The wall clock time of the full analysis was 45:37 with the CPU time only a few seconds less showing that almost all of the time was spent calculating.

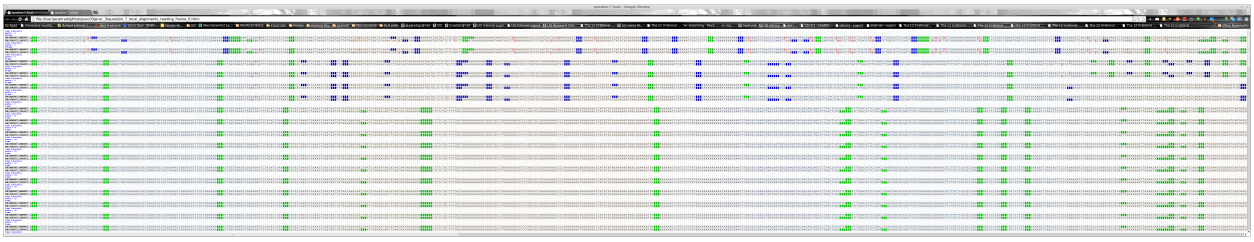


part b

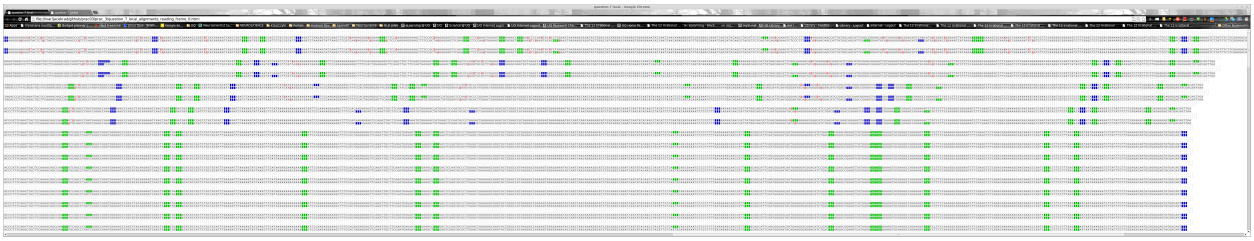
Basic evaluation criteria of finding best alignment was to find the longest alignment starting with a start codon (including alternative AUG, GUG or UUG) and ending with a stop codon (UAG, UGA or UAA) as well as having as little gaps as possible. Screen shots below shows multiple alignments of the two sequences, all using the default (supplied) substitution matrix (dna.matrix), various gap penalties and read frame 0. Reading frames 1 and 2 have been evaluated but did not show alignments anywhere as good as frame 0. See Appendix C for examples

Local

Head of the alignment set

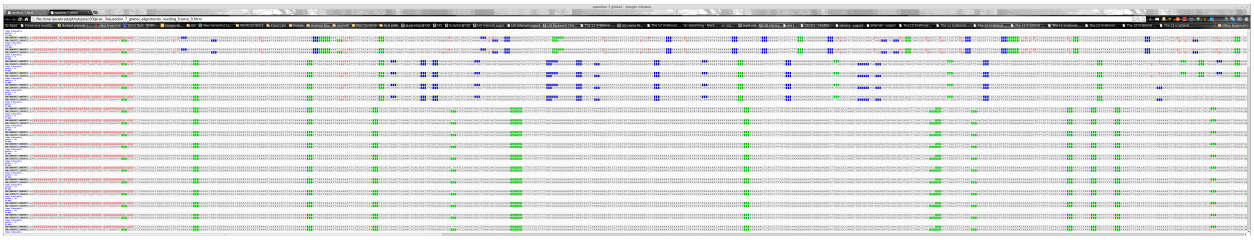


Tail of the alignment set

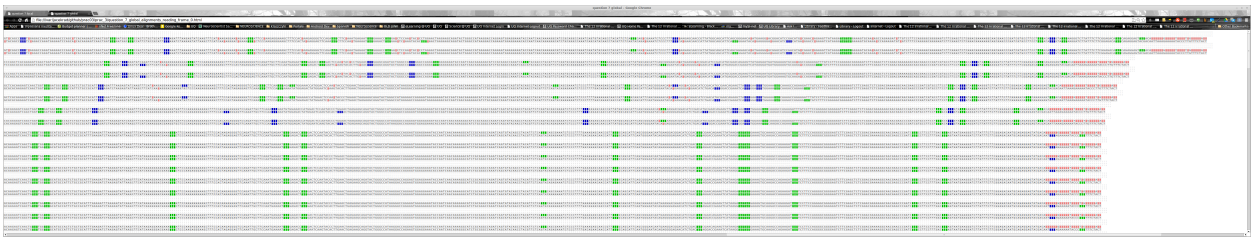


Global

Head of the global alignment set



Tail of the global alignment set



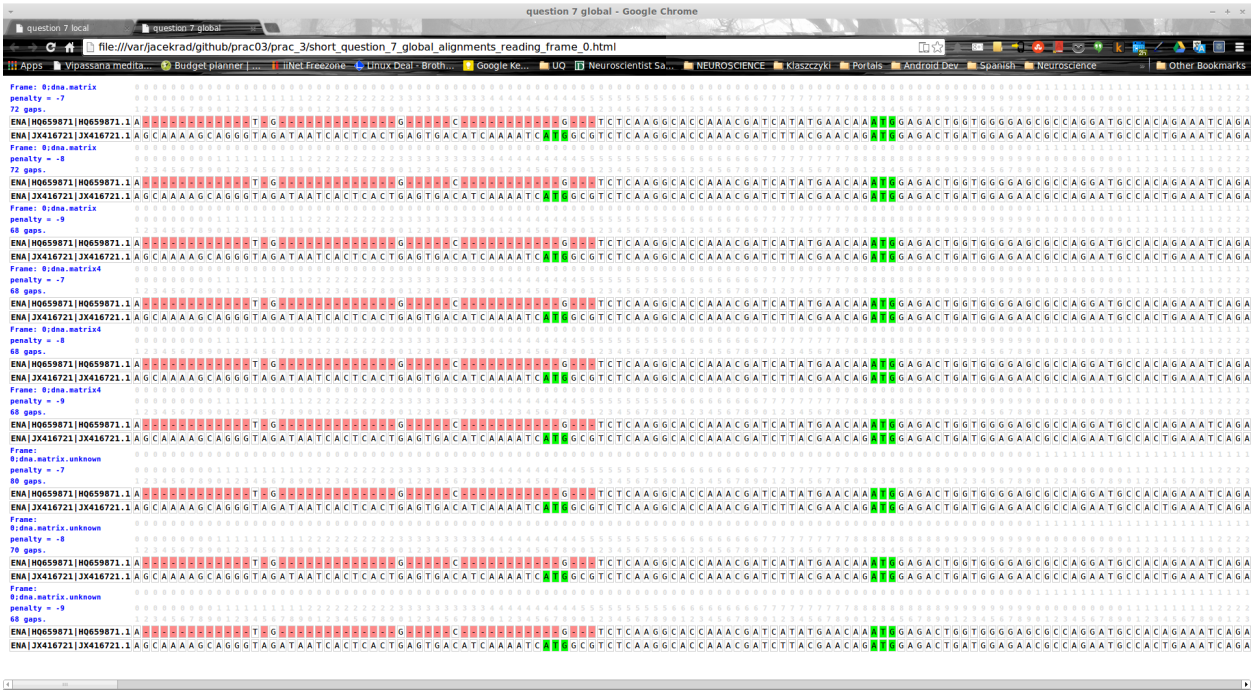
Conclusion

In global thel alignment set, seventh (gap penalty -7) alignment has a AUG start codon at position 82 and a TGA stop codon at position 1003 giving it lengthy of 921 bases or 307 residues. Alignment set with gap penalty -9 is about 50% longer but HQ659871 sequence is missing a stop codon. It would seem that gap penalty of -7 is most appropriate for global alignment. See part C for final conclusion on best local alignment.

Looking at the head of the local set we can see that the seventh alignment (gap penalty -7) differs from the previous 6 as it is not full of stop codons (blue). Looking at the tail of the alignment set we can see 7th and 8th alignments (gap penalty -7 and -8 respectively) do have stop codons present. It is not until gap penalty of -9 that we get a full alignment from beging to end without interuption by stop codons. From then on (gap penalties -10 to -19) alignments do not change. It can be concluded that gap penalty of -9 is the most sensable for local alignment.

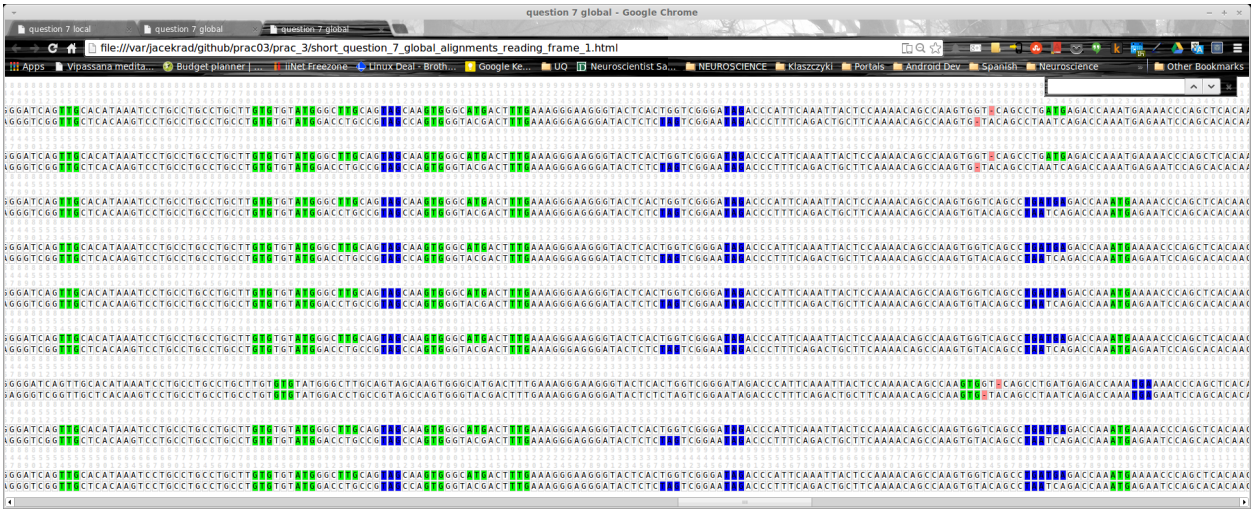
Part C

Taking a closer look at the global alignment set, focusing on gap penalties between -7 and -9 we can see the start codon ATG aligns at position 82. Position 82 is therefore the most likeley place of the start of the protein.



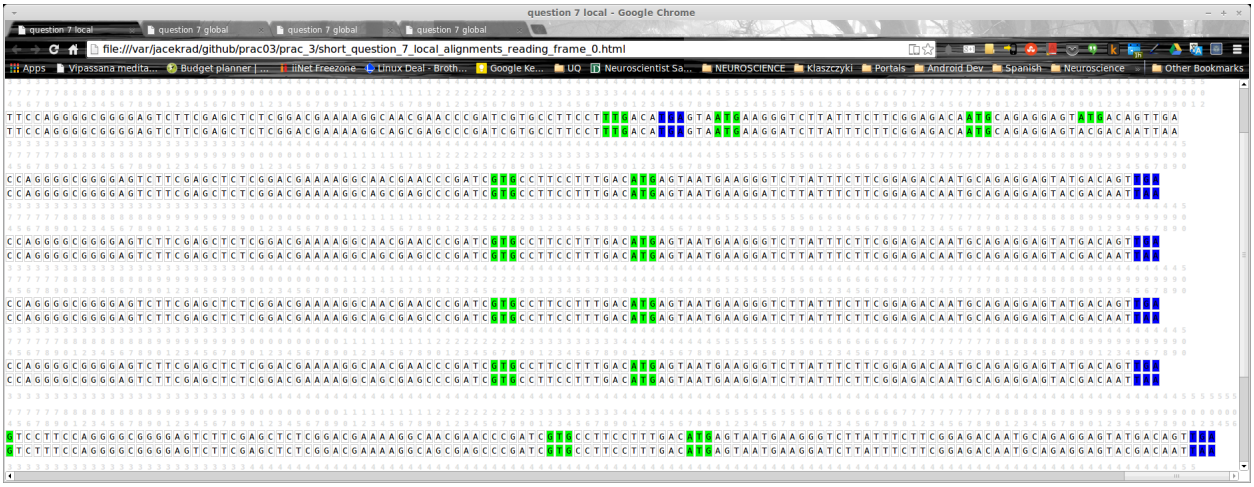
Other start codons have been detected throughout the alignments. One interesting candidate is when reading frame 1 is used instead of 0. In the screenshot below it is the third alignment from the bottom. It only appears in dna.matrix.unknown (matrix from wikipedia) and gap penalty of -7. The potential protein start could be the alternative start codon TTG (position 143) which is followed by another alternative start codon GTG

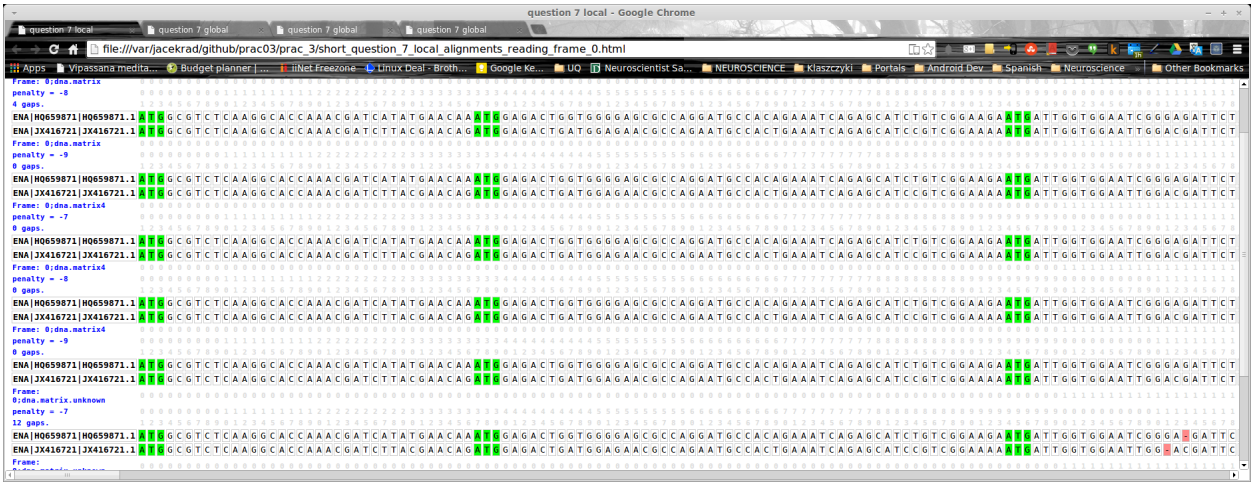
The following screenshot shows the tail of the alignment with the TGA stop codon at position 1004 giving the total potential gene the length of 861 bases or 287 amino acid residues.



If the above was the actual correct alternative start codon it would produce completeley different protein

Even though the above is a potential candidate for a good alignment, local alignments in reading frame 0 for dna.matrix/gap penalty -9 and dna.matrix4/penalties -7 to -9 all produced aligned sequences 1494 bases or 498 residues long. See screenshots below:


question 7 local - Google Chrome
question 7 local
question 7 global
question 7 global
question 7 global
file:///var/jacekrad/github/prac03/prac_3/short_question_7_local_alignments_reading_frame_0.html
Apps
Vipassana medita...
Budget planner | ...
iiNet Freezone
Linux Deal - Broth...
Google Ke...
UQ
Neuroscientist Sa...
NEUROSCIENCE
Klaszczyki
Portals
Android Dev
Spanish
Neuroscience
Other Bookmarks


question 7 local - Google Chrome
question 7 local
question 7 global
question 7 global
question 7 global
file:///var/jacekrad/github/prac03/prac_3/short_question_7_local_alignments_reading_frame_0.html
Apps
Vipassana medita...
Budget planner | ...
iiNet Freezone
Linux Deal - Broth...
Google Ke...
UQ
Neuroscientist Sa...
NEUROSCIENCE
Klaszczyki
Portals
Android Dev
Spanish
Neuroscience
Other Bookmarks
Frame: 0;dna.matrix
penalty = -8
4 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame: 0;dna.matrix
penalty = -9
0 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame: 0;dna.matrix4
penalty = -7
0 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame: 0;dna.matrix4
penalty = -8
0 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame: 0;dna.matrix4
penalty = -9
0 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame: 0;dna.matrix.unknown
penalty = -7
12 gaps.
ENA|HQ659871|HQ659871.1
ENA|JX416721|JX416721.1
Frame:

Question 10

Question 10 was processed in a manner similar to that of question 7. The following is the code:

```
'''
Created on 08/04/2014

@author: s4361277
'''
from sequence import *
from alignments import AlignmentCollection, AlignmentThread
import util

p08684_filename="p08884.fasta"
q16802_filename="q16802.fasta"


# question 8
util.searchAndSave("P08684", p08684_filename)
util.searchAndSave("Q16802", q16802_filename)

matrix_names = ["blosum62.matrix", "p450.matrix"]

# list of matrices though which we shall iterate to obtain various alignments
matrices = []
for matrix_name in matrix_names:
    matrix = readSubstMatrix(matrix_name, Protein_Alphabet)
    print "Adding matrix: " + matrix_name + "\n", matrix
    matrices.append(matrix)

# read our residue sequences from FASTAs
P08684 = readFastaFile(p08684_filename)[0]
Q16802 = readFastaFile(q16802_filename)[0]

local_alignments = AlignmentCollection("question 7 local")
global_alignments = AlignmentCollection("question 7 global")

for matrix in matrices:
    print "processing matrix:", matrix.name
    for gap_penalty in range(-1, -20, -1):
        local_alignments.add_alignment(alignLocal(P08684, Q16802, matrix, gap_penalty))
        global_alignments.add_alignment(alignGlobal(P08684, Q16802, matrix, gap_penalty))

local_alignments.dump_xml_and_html("question_10_local_alignments")
global_alignments.dump_xml_and_html("question_10_global_alignments")
```

Part a

High level screen shots of both local and global alignment sets for both blosum62 and p450 matrix are shown below. There are differences in the two but these are not significant. More detailed (zoomed) screen shots are available in Appendix D.

Global blosum62

Global p450

Local blosum62

Local p450

Part b

Analysing the global alignment sets we can see that for the same gap penalty, those generated using the blosum62 matrix produces fewer gaps in the alignment than those generated using the p450 matrix.

In local alignment sets matrix p450 produced (mostly) fewer gaps than blosum62 for the same gap penelties, but the resulting sequences were also mostly shorter.

There does not seem to be an overwhelming difference between blosum62 and p450 generated alignment sets, but it may be concluded that overall, blosum62 produces slightly better alignments than p450.

Part c

The result obtained in the comparison of the two alignment sets is suprising (to me based on my limited understanding). I would have expected alignments generated using the more tailor p450 matrix to contain fewer gaps than those generated using the stock standard blosum62

Part d

Gap penalty plays a huge role and has very significant impact on the alignment that is generated. For example looking at local alignments we can see that gap penalty of -1 generates a sequence three times as long than gap penalty of -17. This is true for both blosum62 and p450 generated sequences.

Furthermore in the local blosum62 alignments, gap penalty of -1 generates sequences with a total of 260 gaps and gap penalty of -11 just 5 gaps (and a much shorter sequence). Even larger difference in the number of resulting gaps was seen with p450 matrix. Gap penalty of -1 produced an alignment with 318 gaps and penalty of -11 produced just 3. This change in number of gaps bteween low and high gap penalties is also reflected in global alignments.

Appendix A – Unit tests for Questions 1 & 2

```
'''
Created on 08/04/2014

@author: s4361277
unit tests:
prac 3
questions 1 & 2
'''
import unittest

from sequence import Sequence, Protein_Alphabet, readSubstMatrix, alignGlobal, alignLocal

class SequenceTest(unittest.TestCase):

    def setUp(self):
        self.seqA = Sequence("THISLINE", Protein_Alphabet, name="SeqA")
        self.seqB = Sequence("ISALIGNED", Protein_Alphabet, name="SeqB")
        self.blosum62_matrix = readSubstMatrix("blosum62.matrix", Protein_Alphabet)

    def tearDown(self):
        pass

    def testQuestion1_8(self):
        """ test question 1; global alignment with a gap penalty of -8 """
        aln_global_8 = alignGlobal(self.seqA, self.seqB, self.blosum62_matrix, -8)
        expected1 = aln_global_8.seqs[0].getSeqString()
        expected2 = aln_global_8.seqs[1].getSeqString()
        self.assertEquals("THISLINE-", expected1)
        self.assertEquals("ISALIGNED", expected2)

    def testQuestion1_4(self):
        """ test question 1; global alignment with a gap penalty of -4 """
        aln_global_4 = alignGlobal(self.seqA, self.seqB, self.blosum62_matrix, -4)
        expected1 = aln_global_4.seqs[0].getSeqString()
        expected2 = aln_global_4.seqs[1].getSeqString()
        self.assertEquals("THIS-LI-NE-", expected1)
        self.assertEquals("--ISALIGNED", expected2)

    def testQuestion2_8(self):
        """ test question 2; local alignment with a gap penalty of -8 """
        aln_local_8 = alignLocal(self.seqA, self.seqB, self.blosum62_matrix, -8)
        expected1 = aln_local_8.seqs[0].getSeqString()
        expected2 = aln_local_8.seqs[1].getSeqString()
        self.assertEquals("SLI-NE", expected1)
        self.assertEquals("ALIGNE", expected2)

    def testQuestion2_4(self):
        """ test question 2; local alignment with a gap penalty of -4 """
        aln_local_4 = alignLocal(self.seqA, self.seqB, self.blosum62_matrix, -4)
        expected1 = aln_local_4.seqs[0].getSeqString()
        expected2 = aln_local_4.seqs[1].getSeqString()
        self.assertEquals("IS-LI-NE", expected1)
        self.assertEquals("ISALIGNE", expected2)

if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

Appendinx B.1 – `Alignment.toXML()`

```
    def toXML(self, reading_frame = 0):
        """ Attempt to dump an XML representation of this alignment
        """

        xml = ""

        # output reading from for DNA alphabets only
        if self.alphabet == DNA_Alphabet:
            xml += "<sequence-alignment gap-penalty=\"" + str(self.gap_penalty) + "\" \
                + " matrix-name=\"" + str(self.matrix_name) + "\" \
                + " reading-frame=\"" + str(reading_frame) + "\">"
        else:
            xml += "<sequence-alignment gap-penalty=\"" + str(self.gap_penalty) + "\" \
                + " matrix-name=\"" + str(self.matrix_name) + "\">"
        xml += "<sequences>\n"
        maxNameLength =  self.getnamelen()
        xml += ''.ljust(maxNameLength) + ' '
        for sequence in self.seqs:
            xml += "<sequence>\n"
            xml += "<name>" + sequence.name.ljust(maxNameLength) + "</name>\n"
            seq_length = len(sequence)
            position = 0
            sof = False # start of frame flag
            for sym in sequence:
                if ((position - reading_frame) % 3) == 0:
                    sof = True
                else:
                    sof = False
                if sym == '-':
                    xml += "<GAP/>"
                elif sof and seq_length > (position + 2) and \
                        sym == 'A' and sequence[position + 1] == 'T' and sequence[position + 2] == 'G':
                    xml += "<" + sym + " codon=\"start\" />"
                elif sof and seq_length > (position + 2) and \
                        sym == 'G' and sequence[position + 1] == 'T' and sequence[position + 2] == 'G':
                    xml += "<" + sym + " codon=\"start\" />"
                elif sof and seq_length > (position + 2) and \
                        sym == 'T' and sequence[position + 1] == 'T' and sequence[position + 2] == 'G':
                    xml += "<" + sym + " codon=\"start\" />"
                elif sof and seq_length > (position + 2) and \
                        sym == 'T' and sequence[position + 1] == 'A' and sequence[position + 2] == 'G':
                    xml += "<" + sym + " codon=\"stop\" />"
                elif sof and seq_length > (position + 2) and \
                        sym == 'T' and sequence[position + 1] == 'G' and sequence[position + 2] == 'A':
                    xml += "<" + sym + " codon=\"stop\" />"
                elif sof and seq_length > (position + 2) and \
                        sym == 'T' and sequence[position + 1] == 'A' and sequence[position + 2] == 'A':
                    xml += "<" + sym + " codon=\"stop\" />"
                else:
                    xml += "<" + sym + "/>"
                position += 1
            xml += "</sequence>\n"
        xml += "</sequences></sequence-alignment>\n"
        return xml
```

Appending B.2 – alignments.py

```
'''
Created on 12/04/2014

@author: jacekrad
'''

from lxml import etree
import threading as thread
from sequence import *


class AlignmentCollection():
    '''
    simple collection of multiple alignments
    main purpose of this class is to allow us to compare multiple alignments
    with each other analysing gaps etc
    '''

    def __init__(self, name):
        '''
        Constructor
        '''
        self.name = name
        self.alignments = []
        self.is_dna = False

    def add_alignment(self, alignment):
        """ append an alignment to this collection """
        self.alignments.append(alignment)

    def toXML(self, reading_frame = 0):
        """ return XML representation of this collection of alignments """
        if self.is_dna:
            xml = "<alignments name=\"" + self.name + "\" reading_frame=\"" + str(reading_frame) + "\">\n"
        else:
            xml = "<alignments name=\"" + self.name + "\">\n"
        for alignment in self.alignments:
            xml += alignment.toXML(reading_frame = reading_frame)
        xml += "</alignments>\n"
        return xml

    def dump_xml_and_html(self, prefix = "output"):
        """ dump raw XML representation of these alignments as well as
        transformed into HTML output into files named by the prefix """

        # if these are alignments of DNA then we generate 3 of each,
        # one per reading frame
        if self.alignments[0].seqs[0].alphabet == DNA_Alphabet:
            self.is_dna = True

        if self.is_dna:
            print prefix, " sequences are DNA, iterating through reading frames ..."
            for reading_frame in range(0, 3):
                xml_filename = prefix + "_reading_frame_" + str(reading_frame) + ".xml"
                html_filename = prefix + "_reading_frame_" + str(reading_frame)+ ".html"
                xslt_filename = "alignments_html.xsl"
                # dump XML
                open(xml_filename, "w").write(self.toXML(reading_frame))
                #dump HTML
                open(html_filename, "w").write(self.transform(xml_filename, xslt_filename))

        else:
            xml_filename = prefix + ".xml"
            html_filename = prefix + ".html"
            xslt_filename = "alignments_html.xsl"
            # dump XML
            open(xml_filename, "w").write(self.toXML())
            #dump HTML
            open(html_filename, "w").write(self.transform(xml_filename, xslt_filename))


    def transform(self, xmlPath, xslPath):
        xslRoot = etree.fromstring(open(xslPath).read())
        transform = etree.XSLT(xslRoot)
        xmlRoot = etree.fromstring(open(xmlPath).read())
        transRoot = transform(xmlRoot)
        return etree.tostring(transRoot)


class AlignmentThread (thread.Thread):

    def __init__(self, HQ659871_1, JX416721_1, matrix_filename, gap_penalty, alphabet =  DNA_Alphabet):
        thread.Thread.__init__(self)
        self.seqA = HQ659871_1
        self.seqB = JX416721_1
        self.matrix = readSubstMatrix(matrix_filename, alphabet)
        self.gap_penalty = gap_penalty
        self.result_alignment =  None

    def run(self):
        print "Starting " + self.name
        self.result_alignment = alignGlobal(self.seqA, self.seqB, self.matrix, self.gap_penalty)
        print self.name, " completed"
```

Appendix B.3 alignments_html.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs" version="2.0">

    <xsl:output method="xhtml" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/">
        <html>
            <head>
                <link href="alignments.css" rel="stylesheet" type="text/css"/>
                <title>
                    <xsl:value-of select="alignments/@name"/>
                </title>
            </head>
            <body>
                <table>
                    <xsl:apply-templates select="alignments/sequence-alignment"/>
                </table>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="sequence-alignment">
        <tbody class="alignment">
            <tr>
                <th class="desc">
                    <xsl:if test="@reading-frame">
                        <xsl:text>Frame: </xsl:text>
                        <xsl:value-of select="@reading-frame"/>
                        <xsl:text>;</xsl:text>
                    </xsl:if>
                    <xsl:value-of select="@matrix-name"/>
                </th>
                <xsl:for-each select="sequences/sequence[1]/node()">
                    <th>
                        <xsl:value-of select="floor((position() mod 1000) div 100)"/>
                    </th>
                </xsl:for-each>
            </tr>
            <tr>
                <th class="desc">penalty = <xsl:value-of select="@gap-penalty"/></th>
                <xsl:for-each select="sequences/sequence[1]/node()">
                    <th>
                        <xsl:value-of select="floor((position() mod 100) div 10)"/>
                    </th>
                </xsl:for-each>
            </tr>
            <tr>
                <th class="desc"><xsl:value-of select="count(.//GAP)"/> gaps.</th>
                <xsl:for-each select="sequences/sequence[1]/node()">
                    <th>
                        <xsl:value-of select="position() mod 10"/>
                    </th>
                </xsl:for-each>
            </tr>
            <xsl:for-each select="sequences/sequence">
                <tr class="alignment">
                    <td class="seq_name">
                        <xsl:value-of select="name"/>
                    </td>
                    <xsl:apply-templates select="node()[local-name() != 'name']"/>
                </tr>
            </xsl:for-each>
        </tbody>
    </xsl:template>

    <xsl:template match="GAP">
        <td class="gap alignment">-</td>
    </xsl:template>

    <xsl:template match="A[@codon = 'start']">
        <td class="atg_codon">A</td>
    </xsl:template>

    <xsl:template match="T[@codon = 'start']">
        <td class="atg_codon">T</td>
    </xsl:template>

    <xsl:template match="G[@codon = 'start']">
        <td class="atg_codon">G</td>
    </xsl:template>

    <xsl:template match="T[preceding-sibling::node()[1][@codon = 'start']]">
        <td class="atg_codon">T</td>
    </xsl:template>

    <xsl:template match="G[preceding-sibling::node()[2][@codon = 'start']]">
        <td class="atg_codon">G</td>
    </xsl:template>

    <xsl:template match="T[@codon = 'stop']">
        <td class="stop_codon">T</td>
    </xsl:template>

    <xsl:template match="A[preceding-sibling::node()[1][@codon = 'stop']]">
        <td class="stop_codon">A</td>
    </xsl:template>

    <xsl:template match="A[preceding-sibling::node()[2][@codon = 'stop']]">
        <td class="stop_codon">A</td>
    </xsl:template>

    <xsl:template match="G[preceding-sibling::node()[1][@codon = 'stop']]">
```

```
        <td class="stop_codon">G</td>
    </xsl:template>

    <xsl:template match="G[preceding-sibling::node()[2][@codon = 'stop']]">
        <td class="stop_codon">G</td>
    </xsl:template>

    <xsl:template match="M">
        <td class="atg_codon">M</td>
    </xsl:template>

    <!-- fallback match of non-special letters -->
    <xsl:template match="A | B| C | D | E | F | G | H | I | J | K | L | N | O | P | Q | R | S| T | U | V | X | Y | Z">
        <td class="alignment">
            <xsl:value-of select="local-name()"/>
        </td>
    </xsl:template>

</xsl:stylesheet>
```

Appendix B.4 – alignment.css

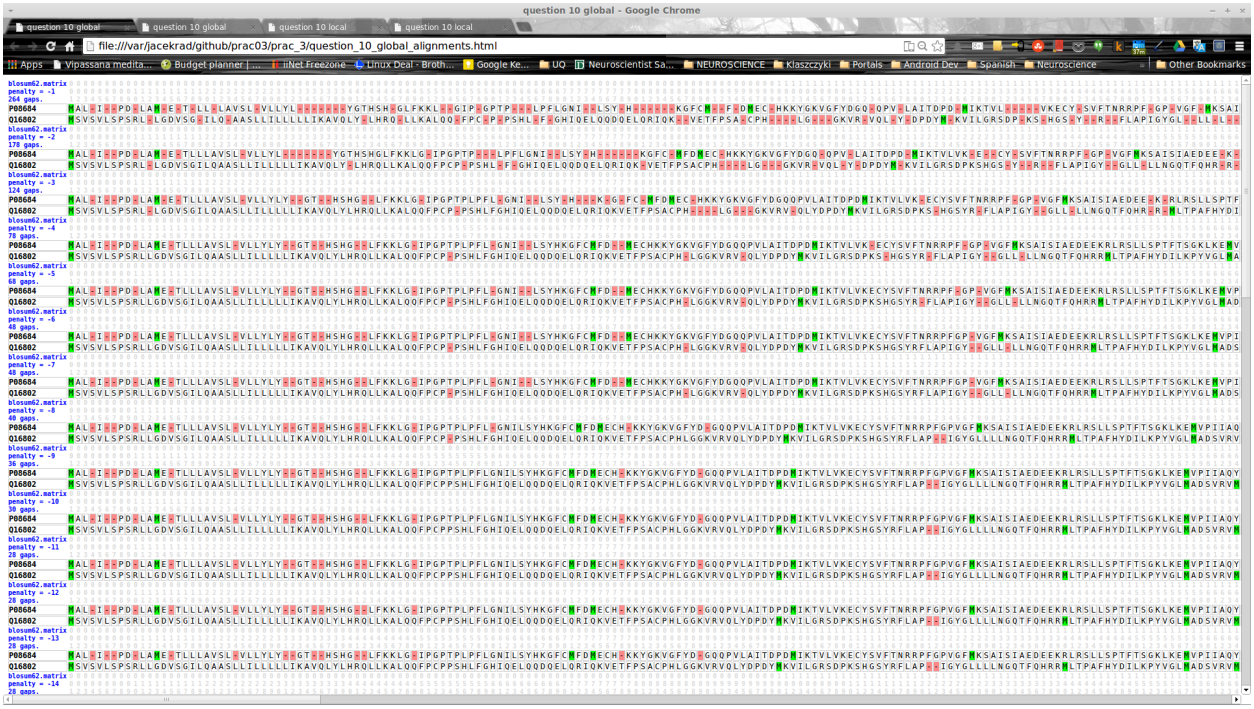
```
td {
    font-family:monospace;
    border:1px solid #CCC;
}
th {
    font-family:monospace;
    font-size:smaller;
    color:#DDD;
}
.desc {
    color: blue;
    text-align:left;
}
.gap {
    background-color:#FF8888;
}
.atg_codon {
    background-color:#00FF00;
}
.stop_codon {
    background-color:#0000FF;
}
.seq_name {
    font-weight:bold;
}
.alignment {}
```

Appendix C – Selected Question 7 allignment sets

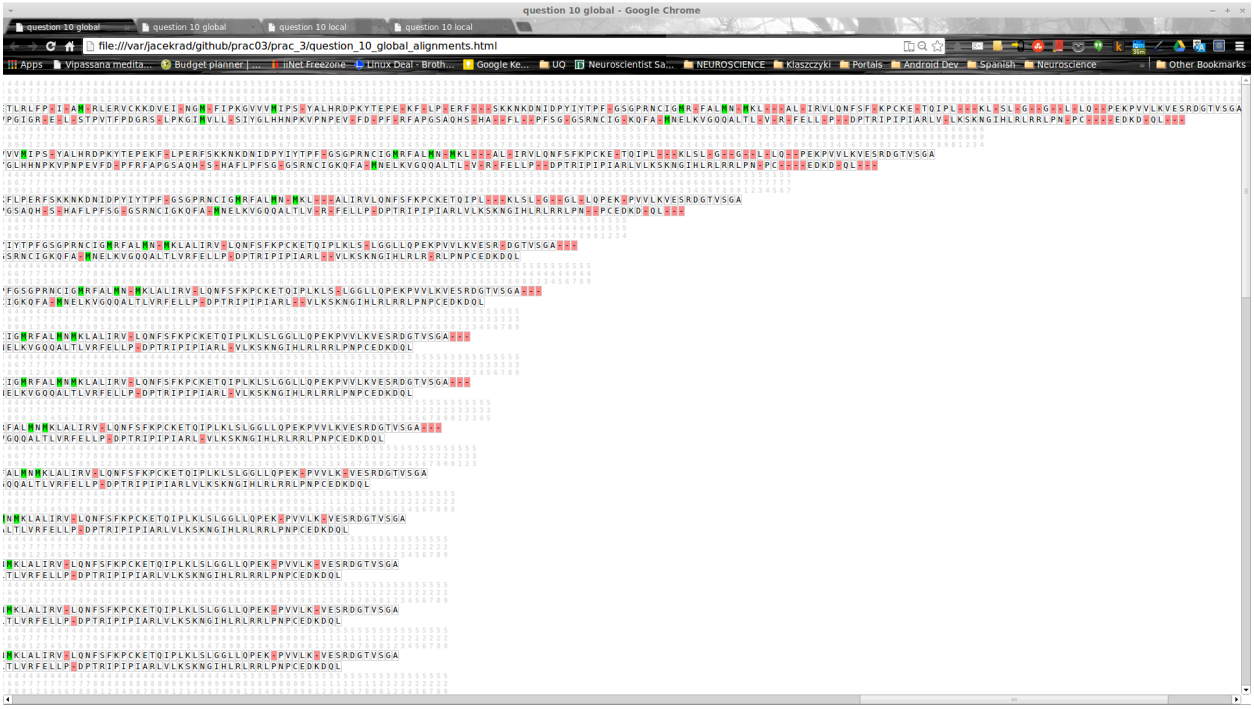
C.1 Global alignment, gaps -7 to -9 using reading frame 2

Appendix D – Question 10 screen shots

D.1 – global, blosum62, head

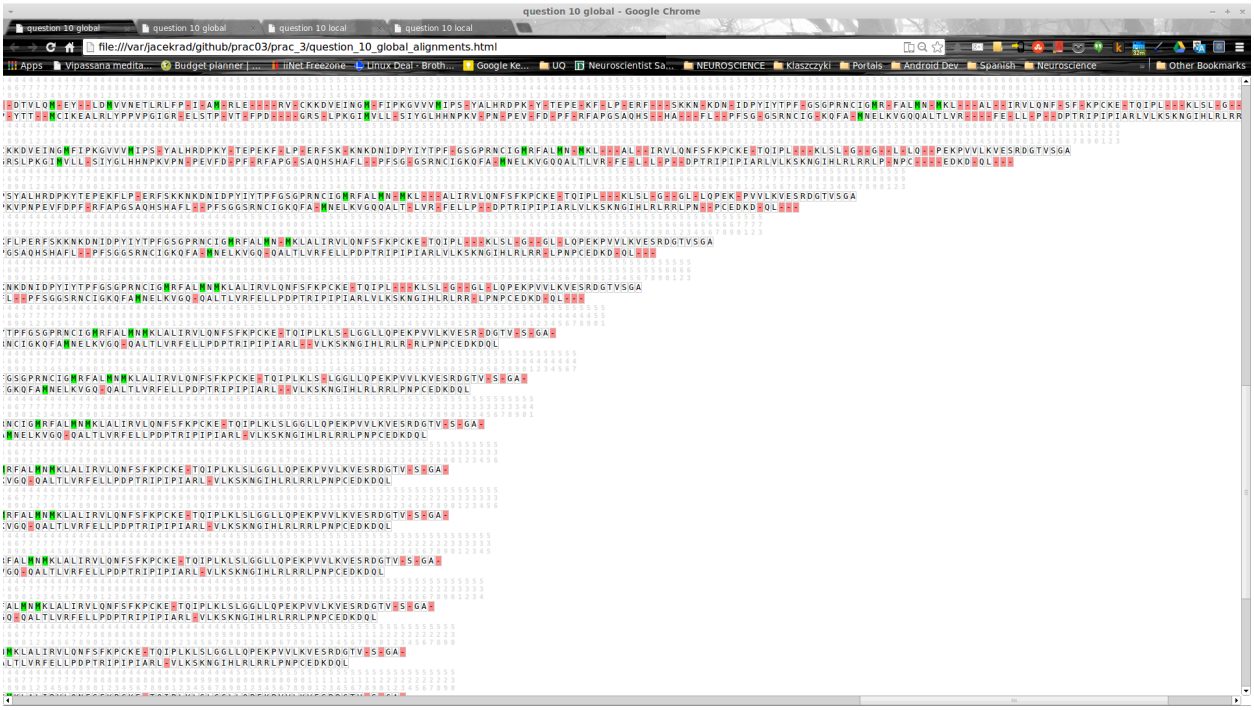


D.2 – global, blosum62 tail



Appendix D.3 – global, p450 head

Appendix D.4 – global, p450 tail



Appendix D.5 – local, blosum62 head

Appendix D.6 – local, blosum62, tail

Appendix D.7 – local, p450 head

Appendix D.8 – local, p450 tail