PairwiseAlign

A home-made alignment tool

Álvaro Ponce Cabrera

	C	T	A	G	A	A	C	T	T
C	X						X		
T		X						X	X
A			X		X	X			
G				X					
T		X	X			X			
A			X		X	X			
C	X						X		
T		X						X	X
T		X						X	X

Introduction

- Align sequences is a hard computational challenge, and even the algorithms every scientist usually uses can't do it 100% eficiently.
- For these reasons I tried to create my own Align tool, in order to understand why it is so hard by taking the challenge on my own.
- I tried first to create my own way to align sequences, trying to work only with the sequences, but finally, it was decided to follow the clasic way to create alignments with substitutions and dinamics Matrix, and creating individual scores for each nucleotide match or missmatch, and analyzing the dinamic matrix to generate the final alignment.

Estructure

- To handle with the challenge I follow the next structure:
- **ParwiseAlign.py: main** script. It works like a module and contains all the functions necessary to do the align.
- **FinalPractise.py: i**n order to give a graphical support to the project a very simple flask interface was coded here.
  - Templates:
  - PairwiseAlign_Template.html: main template.
  - AlignResults.html: template of results.
  -

Modules

- Only two modules were used, **matplotlib.pyplot**, and **Flask**.
- At the begining, the idea was to include another module with Biopython tools, like work with sequences from databases only using like input the database ID. But due to problems with Python and Biopython versions as well as seq items, finally this feature is not included.

Difficulties

I think I learnt a lot of python with this practise, due to all the problems I had:

- First, I knew that complex algorithms are used to do the alignments, but working with short sequences I tried to generate alignments basing it in matches, missmatches and possibles gaps. But after a few hours I understood that the only way to do it efficiently was using matrix, substitutions and dinamic matrix.
- Matrix were a challenge at the begining, before I understood that in python matrix are done using list and sublist.
- As you can see, in my main script there are a lot of for loops, it was a challenge too generate all of them without missing data.
- One of the hardest points was the analysis of the dinamyc matrix. When the dinamyc matrix is done, you need to analyse it following the highest scores like a snake through the matrix, and, if you see the code now, I only select the higher score, and follow the way back to the (0,0) point jumping over the highest scores. It really wasn't easy. Actually, here I understood why the algorithms have problems with the repetitions, because my code even has problems with repetitions of 2 nucleotides in some cases.

Difficulties II

- The final difficulty with the alignment was the gap asumption, where add gaps, and how. This, in addition to repetitions problem, is the mainly problem of my code, because now I only be able to add gaps according to the matches or missmatches found in the align generated by the dinamic matrix, but I need to compare yet every gap addition with a score value in order to choose the better one.
- In addition, I had some probles with biopython, as i said before, and with Flask.
  I wanted to show the alingment in a better way that it's done now, and also I wanted to show the plot my code has.
- Actually, although the Flask tool is user-friendly, I really recomend to use the console in order to work with PairwiseAlign tool.

Example

- Given the secuences:

```
seq1=('TTCTA')

seq2=('TGCTCTA')
```

```
import PairwiseAlign
```

- Creation of the sequence Matrix:

```
In [9]: M=PairwiseAlign.MGenerator(seq1,seq2)

In [10]: M
Out[10]: [['T', 'T', 'C', 'T', 'A'], ['T', 'G', 'C', 'T', 'C', 'T', 'A']]

In [11]: print(M[0],M[1],sep='\n')
['T', 'T', 'C', 'T', 'A']
['T', 'G', 'C', 'T', 'C', 'T', 'A']
```

- Generation of points of matches and missmathces. Now there are a part of the object generated not created, this is due to the difference length of sequences, when the alingment is ended, the funtion can count also the gaps, and it will fully show gaps, matchs and missmathes:

```
In [12]: points=PairwiseAlign.M_count(M)

In [13]: print(M[0],M[1],points,sep='\n')
['T', 'T', 'C', 'T', 'A']
['T', 'G', 'C', 'T', 'C', 'T', 'A']
['*', ' ', '*', '*', ' ']
```

Example II

- Missmatches, matches and gaps counters show the positions of these into the sequence matrix. If there are no coincidence, a mensage is printed.

```
In [19]: PairwiseAlign.missmatchesCounter(M)
Out[19]: [1, 4]

In [20]: PairwiseAlign.matchesCounter(M)
Out[20]: [0, 2, 3]

In [21]: PairwiseAlign.gapCounter(M)
No gaps found
Out[21]: []
```

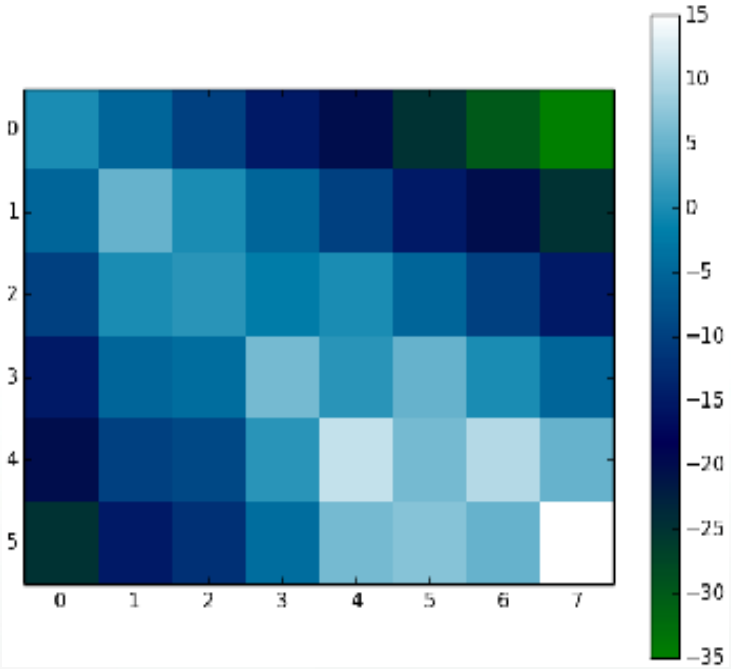
- Generation of substitution and dinamic Matrix:

```
In [24]: PairwiseAlign.substitutionM(M)
Out[24]:
[[5, -4, -2, 5, -2, 5, -4],
 [5, -4, -2, 5, -2, 5, -4],
 [-2, -4, 5, -2, 5, -2, -4],
 [5, -4, -2, 5, -2, 5, -4],
 [-4, -2, -4, -4, -4, -4, 5]]

In [25]: PairwiseAlign.dinamicM(M)
Out[25]:
[[0, -5, -10, -15, -20, -25, -30, -35],
 [-5, 5, 0, -5, -10, -15, -20, -25],
 [-10, 0, 1, -2, 0, -5, -10, -15],
 [-15, -5, -4, 6, 1, 5, 0, -5],
 [-20, -10, -9, 1, 11, 6, 10, 5],
 [-25, -15, -12, -4, 6, 7, 5, 15]]
```

- Plot of dinamic Matrix:

```
PairwiseAlign.dinamicPlot(PairwiseAlign.dinamicM(M))
```



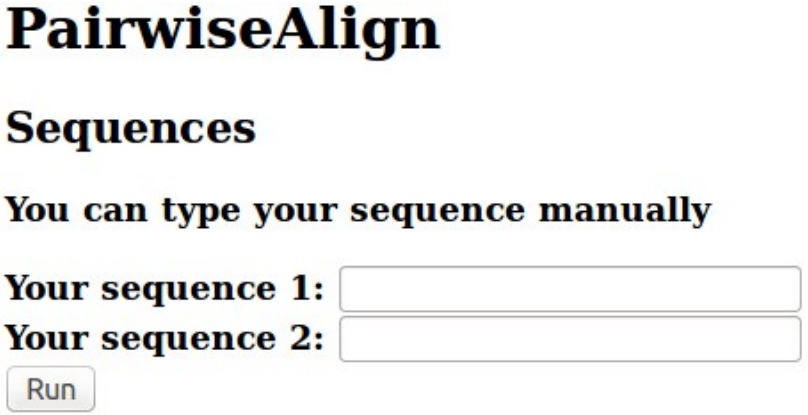
Example III

- Finally, the last function of PairwiseAlign.py create the final alinment of the sequences in sequence Matrix and returns the score of the alignment, the alingment, and a list with the sequence Matrix of the alignemt (list[0]) and the points generated for M_count function, which represent matches, missmatches and gaps.

```
In [7]: PairwiseAlign.dinamicAnalizer(M)
score: 39
['T', 'T', 'C', 'T', '-', '-', 'A']
['T', 'G', 'C', 'T', 'C', 'T', 'A']
['*', ' ', '*', '*', '-', '-', '*']
Out[7]:
([['T', 'T', 'C', 'T', '-', '-', 'A'], ['T', 'G', 'C', 'T', 'C', 'T', 'A']],
 ['*', ' ', '*', '*', '-', '-', '*'])
```

Example IV

- The flask interface looks like that:
  - Main flask template:



  - Result flask template: