Journal of Bioinformatics and Sequence Analysis Vol. 2(3), pp. 30-35, July 2010
Available online at http://www.academicjournals.org/JBSA
ISSN 2141-2464 ©2010 Academic Journals

*Full length Research Paper*

# SEALI: A sequence alignment tool

**Manoj Giri[1]*, Dipti Jindal[2], Savita Kumari[2], Sarla Kumari[3], Devender Singh[4], Jawahar Lal[5] and Neena Jaggi[6]**

[1]Department of Applied Sciences, Haryana College of Technology and Management, Kaithal-136 027, India.
[2]Department of Bioinformatics, Chaudhary Charan Singh Haryana Agricultural University, Hisar- 125 004, India.
[3]Deparmtnet of Electronics and Communication Engineering, Haryana College of Technology and Management, Kaithal-136 027, India.
[4]Department of Applied Sciences, Dayal Singh College, Karnal-132 001, India.
[5]Department of Applied Sciences, Markanda National College, Sahabad (M) 136 135, India.
[6]Department of Applied Sciences, National Institute of Technology, Kurukshetra-136 199, India.

Accepted 21 June, 2010

**In this paper we propose a novel program for sequence alignment. This program has been developed in PERL. The web interface uses CGI and front end for data input and viewing the result has also been developed. This program counts the length of two sequences, aligns the two sequences, counts the number of matches, mismatches, gaps and score and displays the possible alignment.**

**Key words:** Sequnce alignment, SEALI tool, PERL, dynamic programming, DNA, adenine (A), guanine (G), cytosine (C), thymine (T).

## INTRODUCTION

Sequence alignment is by far the most common task in Bioinformatics. A sequence alignment is a way of arranging the primary sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences [Mount, 2004]. Sequence alignments are useful in bioinformatics for identifying sequence similarity, producing phylogenetic trees, and developing homology models of protein structures. Alignments are often assumed to reflect a degree of evolutionary change between sequences descended from a common ancestor [HTTP//www.wikipedia. org/sequence_alignment//.].

There are three primary methods of producing pairwise alignments namely dynamic programming, dotplot and word method. The Needleman and Wunsch algorithm was the first rapid method in the biological literature for determining sequence homology (Needleman and Wunsch, 1970). It was based on dynamic programming and was used to align sequences globally. Smith and Waterman algorithm yields local alignment (Smith and Waterman, 1981). There are various tools for the sequence analysis available in the public domain such as FASTA, LALIGN and PRSS, BLAST, PipMaker, LAGAN, ParAlign, BLAT, YASS and Nigila, etc (Lipman and Pearson, 1985; Pearson, 1990; Altschul et al., 1990; Schwartz et al., 2000; Brudno et al., 2003; Rognes, 2001; Kent, 2002; Noe and Kucherov, 2004; Noe and Kucherov, 2005; Cartwright, 2007; Wang and Jiang, 1994).

The present research has been carried out to develop a simple sequence alignment tool using dynamic programming which counts the length of two sequences, aligns the two sequences, counts the number of matches, mismatches, gaps and score and displays the possible alignment.

## METHODS AND IMPLEMENTATION

The SEALI program has been implemented in Perl script. The web interface uses common gateway interfaces (CGI) and a front end for data input and viewing the results has also been developed. The script for web interfaces has been written in html. The hypertext

*Corresponding author. E-mail: manojgiri1@rediffmail.com. Tel: 097295-73419. Fax: (01746) 280711.

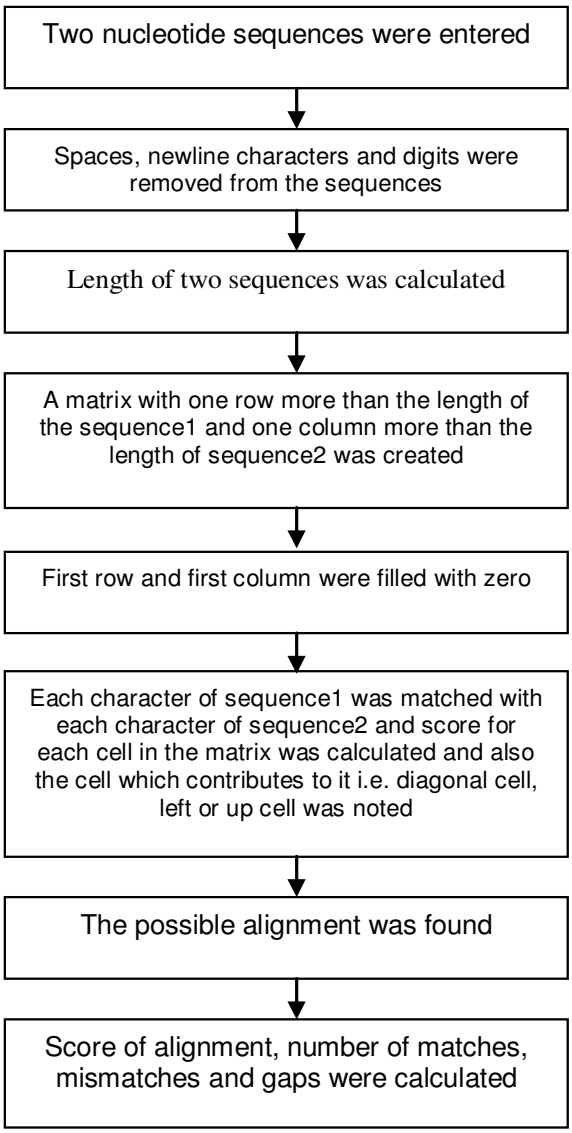


**Figure 1.** Pictorial representation of the steps followed during workflow of SEALI.

transport protocol (HTTP) is used for data transport. The main function of the program developed in the present work is to count the length of two sequences, align the two sequences, count the number of matches, mismatches, gaps, and score and display the possible alignment. The nucleotide or gene sequence (DNA) contains four nucleotide bases adenine (A), guanine (G), cytosine (C) and thymine (T). The sequence of these bases comprises genetic information. The complete nucleotide sequence is taken as a single string. Two nucleotide sequences are taken as two strings. The sequence of characters was given to computer program as input for manipulating for the required purpose. The methods followed for the functioning of the program is summarized as:

1. In the first step, two nucleotide sequences were given as input to the program.
2. The program for alignment either global or local was chosen.
3. It was checked whether the entered sequences were correct or not, that is the correct sequence should not contain any spaces or letter other than the four (A, G, C, and T).
4. The length of the two sequences was counted.
5. Alignment of two sequences was done.
6. Number of matches, mismatches, gaps and score were counted.
7. Results were displayed.

**Pictorial presentation**

The pictorial presentation for the SEALI has been shown in the Figure 1.

**Program code**

The program code has been written in PERL language. With the help of the coding, the SEALI tool runs and it takes the two nucleotide sequences as input through web interface and counts the length of two sequences, align the two sequences, counts the number of matches, mismatches, gaps and score and display the possible alignment. In this program there are two library files namely;

global.lib and local.lib.

```
#!/usr/bin/perl
require "global.lib";
require "local.lib";
print"Content-type: text/html\n\n";
print"<html>";
print"<body>";
read(STDIN,$buffer,$ENV{'CONTENT_LENGTH'});
$a = $buffer;
#get the name and value for each form input
@array= split(/&/,$a);
foreach $array(@array)
{
#separate the name and value
($key,$value) =split(/=/,$array);
#convert +signs to space
$value =~tr/+/ /;
#convert hexadecimal to ASCII characters
$value=~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
#store values in a hash called %form
$form{$key}=$value;
}
#print the algo  you have selected
print"$form{'algo'}\t";
$b =$form{'algo'};
if($b eq 'needle')
{
 &global($form{'text1'} ,$form{'text2'});
}
else
{
&local($form{'text1'} ,$form{'text2'});
}
print"</body>";
print"</html>";
```

Coding of library files

global.lib

```
sub need1
{
$seq1 = $_[0];
$seq2 =$_[1];
my $match =2;
```

```
my $mismatch =-1;
my $gap =-2;
print"(global alignment)<br>";
my @best_score;
$seq1 =~s/\n|\s*|\d*//gi;
print"$seq1<br>";
$seq2 =~s/\n|\s*|\d*//gi;
 print"$seq2<br>";
my $size1=length($seq1);
print"length of seq1 is : $size1<br>";
my $size2 = length($seq2);
print"length of seq2 is : $size2<br>";
print"<br>";
# initialization
my @matrix;
$matrix[0][0] = 0;
for(my $j = 1; $j <= $size1; $j++) {
$matrix[0][$j]   =0;
}
for(my $i = 1; $i <=$size2; $i++) {
$matrix[$i][0]   = 0;
 for(my $j = 1; $j <=$size1; $j++) {
my ($diagonal_score, $left_score, $up_score);
# calculate match score
my $letter1 = substr($seq1, $j-1, 1);
my $letter2 = substr($seq2, $i-1, 1);
if ($letter1 eq $letter2) {
$diagonal_score = $matrix[$i-1][$j-1] + $match;
}
else {
$diagonal_score = $matrix[$i-1][$j-1] + $mismatch;
}
# calculate gap scores
$up_score   = $matrix[$i-1][$j] + $gap;
$left_score = $matrix[$i][$j-1] + $gap;
# choose best score
if ($diagonal_score >= $up_score)
{
if ($diagonal_score >= $left_score)
{
$matrix[$i][$j]  = $diagonal_score;
$best_score[$i][$j] = "diagonal";
}
else
{
$matrix[$i][$j]   = $left_score;
$best_score[$i][$j] = "left";
}
}
else {
if ($up_score >= $left_score) {
$matrix[$i][$j]   = $up_score;
$matrix[$i][$j]   = $up_score;
$best_score[$i][$j] = "up";
}
else {
$matrix[$i][$j]   = $left_score;
            $best_score[$i][$j] = "left";
}
}

}
}

# trace-back
my $align1 = "";
my $align2 = "";
# start at last cell of matrix
my $j = $size1;

my $i = $size2;
while($i!=0 || $j!=0) {
if ($best_score[$i][$j] eq "diagonal") {
$align1 .= substr($seq1, $j-1, 1);
$align2 .= substr($seq2, $i-1, 1);
$i--;
$j--;
}
elsif ($best_score[$i][$j] eq "left") {
$align1 .= substr($seq1, $j-1, 1);
$align2 .= "_";
$j--;
}
elsif ($best_score[$i][$j] eq "up") {
$align1 .= "_";
$align2 .= substr($seq2, $i-1, 1);
$i--;
}
elsif($j==0)
{
$align1.="_";
$align2 .= substr($seq2,$i-1,1);
$i--;
}
elsif($i == 0)
{
$align1 .=substr($seq1,$j-1,1);
$align2 .="_";
$j--;
}
}
$align1 = reverse $align1;
$align2 = reverse $align2;
#score
@a =split(//,$align1);
@b = split(//,$align2);
$len= @a;
$sum=0;
$j=0;
$nomatch = 0;
$nomismatch = 0;
$nogap = 0;
for($i=0;$i<$len;$i++)
{
while($i==$j)
{
if($a[$i] eq $b[$i])
{
$sum =$sum+ $match;
$nomatch = $nomatch + 1;
}

elsif(($a[$i] eq "_")||($b[$i] eq "_"))
{
$sum = $sum + $gap;
$nogap = $nogap +1;
}
else
{
$sum = $sum+ $mismatch;
$nomismatch = $nomismatch +1;
}
$j++;
}
}
print" score: $sum<br>";
```

```
print" matches: $nomatch<br>";
print" mismatches: $nomismatch<br>";
print" gaps: $nogap<br>";
$l1 = length($align1);
$l2 = length($align2);
print"<b>The possible alignment is:</b><br>";
$i=1;
for($i=1;$i<=$l1;$i++)
{
if($l1 >60)
{
$align1=~m/.{60}/;
print"$&<br>";
$align1= $';
$l1 = length($align1);
$align2=~m/.{60}/;
print"$&<br><br>";
$align2= $';
}
if($l1<=60)
{
print"$align1<br>";
print"$align2<br><br>";
last
}
}
}
1;
```

Coding of local.lib

```
sub water1
{
print"(local alignment)<br>";
$seq1= $_[0];
$seq2= $_[1];
my $match    =  2;
my $mismatch = -1;
my $gap      = -2;
# initialization
my @matrix;
my @best_score;
$seq1 =~s/\n|\s*|\d*//gi;
print"$seq1<br>";
$seq2 =~s/\n|\s*|\d*//gi;
print"$seq2<br>";
$len1= length($seq1);
$len2= length($seq2);
print"length of sequence1 is $len1 <br>";
print"length of sequence2 is $len2 <br>";
$matrix[0][0]   = 0;

for(my $j = 1; $j <= length($seq1); $j++) {
$matrix[0][$j]   = 0;
}
for (my $i = 1; $i <= length($seq2); $i++) {
$matrix[$i][0]  = 0;
# fill
for(my $j = 1; $j <= length($seq1); $j++) {
my ($diagonal_score, $left_score, $up_score);
# calculate match score

my $letter1 = substr($seq1, $j-1, 1);
my $letter2 = substr($seq2, $i-1, 1);
if ($letter1 eq $letter2) {
$diagonal_score = $matrix[$i-1][$j-1] + $match;
}
else {
$diagonal_score = $matrix[$i-1][$j-1]+ $mismatch;

}
# calculate gap scores
$up_score   = $matrix[$i-1][$j] + $gap;
$left_score = $matrix[$i][$j-1] + $gap;
if ($diagonal_score <= 0 and $up_score <= 0 and $left_score <= 0)
{
$matrix[$i][$j]   = 0;
}
# choose best score
else{
if($diagonal_score >= $up_score) {
if ($diagonal_score >= $left_score) {
$matrix[$i][$j]  = $diagonal_score;
$best_score[$i][$j]="diagonal";
}
else {
$matrix[$i][$j]   = $left_score;
$best_score[$i][$j]="left";
}
} else {
if ($up_score >= $left_score) {
$matrix[$i][$j] = $up_score;
$best_score[$i][$j]= "up";
}
else{
$matrix[$i][$j]= $left_score;
$best_score[$i][$j]= "left";
}
}
}
}
}
my $max_score = 0;
my $min_score = 99;
my $s=0;
my $t=0;
my $u=0;
my @index1;
my @index2;
for(my $i=1;$i<=length($seq2);$i++)
{
for(my $j=1;$j<=length($seq1);$j++)
{
if($matrix[$i][$j]>= $max_score){
$max_score = $matrix[$i][$j];

$index1[$s] =$i;
$index2[$t] = $j;
$s++;
$t++;
}
if($matrix[$i][$j]< $min_score){
$min_score= $matrix[$i][$j] ;
}
$maxscore[$u] = $matrix[$i][$j];
$u++;
}
}
print"<br>";
for(my $i=1;$i<=length($seq2);$i++)
{
for(my $j=1;$j<=length($seq1);$j++)
{
if($matrix[$i][$j]== $max_score){
$ind1[$r] = $i;
```

```
$ind2[$s] = $j;
$r++;
$s++;
}
}
}

#trace back
my $m=0;
my $n=0;
foreach $i(@ind1)
{
foreach$j(@ind2)
{
while(!($matrix[$i][$j] == 0))
{
 if($best_score[$i][$j] eq "diagonal")
{
$align1[$m] .= substr($seq1, $j-1,1);
$align2[$n] .= substr($seq2, $i-1,1);
$i--;
$j--;
}
 elsif($best_score[$i][$j] eq "left")
{
$align1[$m] .= substr($seq1,$j-1,1);
$align2[$n] .= "_";
$j--;
}
 elsif($best_score[$i][$j] eq "up")
 {
$align1[$m] .="_";
$align2[$n] .=substr($seq2 ,$i-1,1);
$i--;
}
}
}
$align1[$m]= reverse $align1[$m];
$align2[$n] = reverse $align2[$n];
$m++;
$n++;
}
#score
$p=0;
$o=0;
foreach $a(@align1)
{
@c =split(//,$a);
$len=@c;
foreach $b(@align2)
{
while($o==$p)
{
@d=split(//,$b);
$j=0;
$sum=0;
$nomatch=0;
$nomismatch=0;
$nogap=0;
for($i=0;$i<$len;$i++)
{
while($j==$i)
{
if($c[$j] eq $d[$j])
{
$sum =$sum+ $match;
$nomatch = $nomatch +1;
}
elsif(($c[$i] eq'_')||($d[$i] eq'_'))
{
$sum += $gap;
$nogap= $nogap+1;
}
else
{
$sum += $mismatch;
$nomismatch = $nomismatch+1;
}
$j++;
}
}
print"score:  $sum<br>";
print"matches:  $nomatch<br>";
print"mismatches:  $nomismatch<br>";
print"gaps:  $nogap<br>";
last;
}
$p++;
}
$p=0;
$o=$o+1;
}
$length = @align1;
for($j=0;$j<$length;$j++)
{
$align1[$j]=$align1[$j];
$align2[$j]=$align2[$j];
$l1= length($align1[$j]);
for($i=1;$i<=$l1;$i++)
{
if($l1 >60)
{
$align1[$j]=~m/.{60}/;
print"$&<br>";
$align1[$j]= $';
$l1 = length($align1[$j]);
$align2[$j]=~m/.{60}/;
print"$&<br><br>";
$align2[$j]= $';
}
if($l1<60)
{
print"$align1[$j]<br>";
print"$align2[$j]<br><br>";
last;
}
}
}
}
1;
```

## DISCUSSION

A sequence alignment is a way of arranging the primary sequences of DNA, RNA, or protein. Alignment provides a powerful tool to compare related sequences, and the alignment of two residues could reflect a common evolutionary origin, or could represent common structural and/or catalytic roles, not always reflecting an evolutionary process. If two sequences in an alignment share a common ancestor, mismatches can be interpreted as point mutations and gaps as indels introduced in one or both lineages in the time since they diverged from one

another [HTTP//www.wikipedia.org/sequence_alignment//.].

Pairwise sequence alignment methods are used to find the best-matching piecewise (local) or global alignments of two query sequences. Pairwise alignments can only be used between two sequences at a time. Multiple sequence alignment is an extension of pairwise alignment to incorporate more than two sequences at a time. Multiple alignment methods try to align all of the sequences in a given query set. Multiple alignments are often used in identifying conserved sequence regions across a group of sequences hypothesized to be evolutionarily related (Wang and Jiang, 1994). Computational approaches to sequence alignment generally fall into two categories: global alignments and local alignments. So SEALI is a tool which aligns the two sequences globally and locally. For the tool development the code was written in perl and script for front end was written in html. Two nucleotide sequences were given as input to the program. In this tool, sequences can be aligned either locally or globally by selecting one option. It was checked whether the entered sequences were correct or not, that is the correct sequence should not contain any spaces or letter other than the four (A, G, C, and T). The length of the two sequences was counted. Alignment of two sequences was done. Number of matches, mismatches, gaps and score were counted. Results were found.

## ACKNOWLEDGMENTS

The authors are thankful to Dr. Sudhir Kumar, Associate Professor and Head of the Bioinformatics Section, at CCSHAU, Hisar for his inspiration to the present work.

## REFERENCES

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). Basic local alignment search tool. J. Mol. Biol., 215: 403-410.

Brudno M, Do CB, Cooper GM, Kim MF, Davydov E, Green ED, Sidow A, Batzoglou S (2003). LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA. Genome Res., 13: 721-731.

Cartwright AR (2007). Ngila: global pairwise alignments with logarithmic and affine gap costs. Bioinform, 23: 1427-1428.

Kent WJ (2002). BLAT-The BLAST-like alignment tool. Genome Res., 12: 656-664.

Mount DW (2004). Bioinformatics: Sequence and Genome Analysis (2nd ed), Cold Spring Harbour Laboratory Press: Cold Spring Harbour,HTTP//www.wikipedia.org/sequence_alignment//.

Needleman SB, Wunsch CD (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol., 48: 443-453.

Noe L, Kucherov G (2004). Improved hit criteria for DNA local alignment. BMC Bioinform., 5: 149.

Noe L, Kucherov G (2005). YASS: enhancing the sensitivity of DNA similarity search. Nucleic Acid Res., 33: 540-543.

Pearson WR (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. Methods Enzymol., 183: 63–98.

Pearson WR, Lipman DJ (1985). Rapid and sensitive protein similarity searches. Science, 227(4693): 1435-1441.

Rognes T (2001). ParAlign: a parallel sequence alignment algorithm for rapid and sensitive database searches. Nucleic Acids Res., 29: 1647-1652.

Schwartz S, Zhangf Z, Frazer KA, Smit A, Riemer C, Bouck J, Gibbs R, Hardison R, Miller W (2000). PipMaker-A web server for aligning two genomic DNA sequences. Genome Res., 10: 577-586.

Smith T, Waterman M (1981). Identification of common molecular subsequences. J. Mol. Biol., 147: 195-197.

Wang L, Jiang T (1994). On the complexity of multiple sequence alignment. J. Comput. Biol., 1: 337-348.