Introduction to Bioinformatics
3rd practice
Dot plot, Pairwise Alignment: Needleman-Wunsch and Smith-Waterman algorithms

I. Dot Plot:

A dotplot provides a graphical comparison of two sequences (protein or DNA). First consider the sequences to be compared written out along Cartesian axes. These sequences will be compared in fixed length sections (word size/ window size for example 11). Each word from the horizontal sequence is compared with each word of the vertical sequence. Words are compared by considering corresponding residues. Using a scoring scheme (for example +1 for a matched pair of bases, 0 for a mismatched pair), a similarity score is computed for each pair of compared words. Significant similarity is defined by the selection of a cut-off score/ threshold (for example 8 indicating 8 or more matched bases between words of size 11 to be significant).

Significantly matching pairs of words are indicated by drawing a dot in a position representing the middles of the matched words. Once all possible words of the horizontal sequence have been compared to all possible words of the vertical sequence, a number of dots should have been plotted. Roughly diagonal runs of dots indicate roughly similar regions in the two sequences being compared. Dotplots provide a comprehensive overview, which is useful when we compare long sequences or entire genomes (http://last.cbrc.jp/); however, textual alignments are required to examine sequence similarities in detail.

Main parameters of dotplots:

1.: **Scoring schema**:

It is table that defines the values of matches and the penalties for mismatches between the sequences.

DNA:

In the simplest case the value of the matches is 1 and the penalty of mismatches is -1. It can be more complex if we take into account for example that some nucleotides are more similar to each other than others. You can find a more complex DNA substitution table here: http://rosalind.info/glossary/dnafull/ . The choice of the nucleotide matrix usually does not have a huge impact for the result of the comparison.

	A	T	G	C
A	1	0	0	0
T	0	1	0	0
G	0	0	1	0
C	0	0	0	1

Protein:

Protein substitution matrices (for example BLOSUM62) are bigger (20 amino acids) and more complex. Their scores must reflect far more than alphabetic identity. Matched amino acids that are not identical but are known to be similar cannot just be regarded as a “mismatch”. The numbers for the most commonly used protein scoring schemes are derived from studies of aligned protein families. They reflect the frequency with which amino acid pairs are observed to successfully substitute for each other and the expected abundance of particular amino acids. Choosing the correct matrix is crucial in the comparison of proteins.

2.: **Cut-off score**:

The higher the cut-off score the less dots will be plotted. But, each dot is more likely to be significant.

The lower the cut-off score the more dots will be plotted. But, dots are more likely to indicate a chance match (noise).

In general, there is no “correct” cut-off score for a dot-plot. It is often necessary to try several values.

3.: **Window size**:

Large word size gives a general overview of the sequences. The detail of individual features may be lost, but the main matching regions are picked out with optimal clarity. “Noise” is reduced, but it is possible that some small “real” features might be missed.

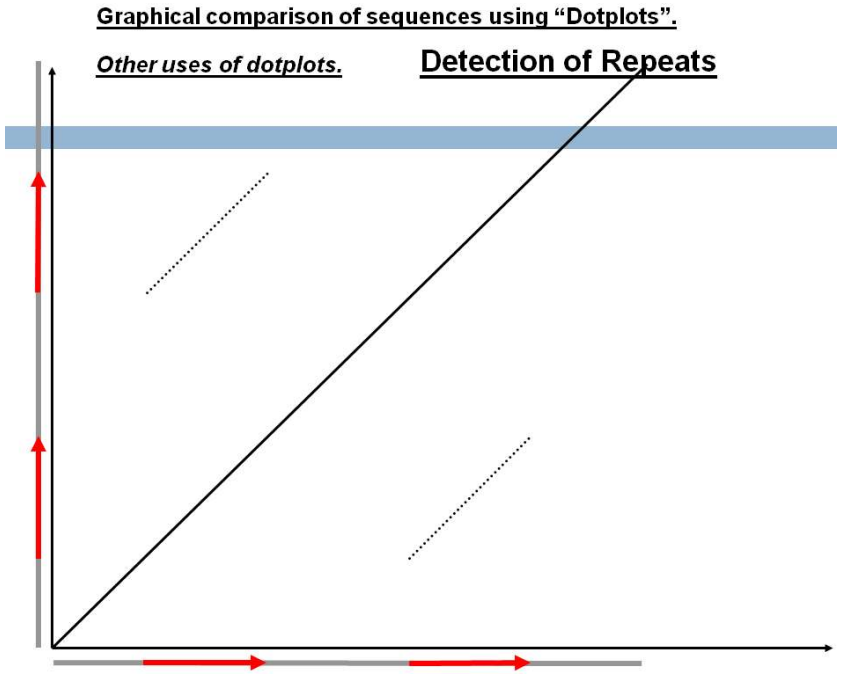
The smaller the word size, the smaller the feature that can be detected. Smaller word size gives more detailed plots. To try for too much detail by using a very small word size normally results in a very “noisy” picture. Very small “features” are usually not real (“noise”).

Appropriate choice of cut-off score for each size of word is important and can reduce unnecessary feature loss when using larger word sizes.

Other uses of dotplot:

1.: Detection of **repeats** in a sequence:

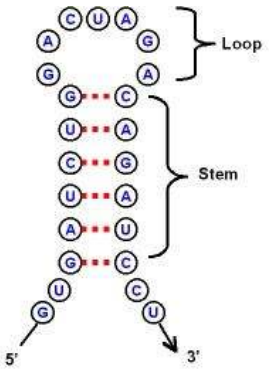
Repeat regions within a single sequence can be detected by computing a dotplot of the sequence compared to itself. A pair of matching repeat regions should generate a line of dots parallel to the leading diagonal indicating their positions. The leading diagonal itself will, of course, be drawn, indicating that the whole sequence is identical to itself! The whole plot will be symmetrical about the leading diagonal.

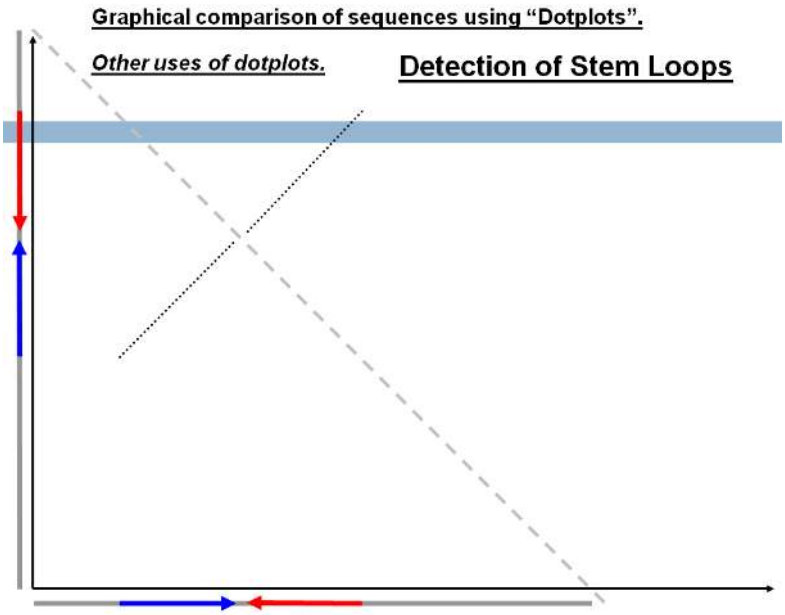


2.: Detection of **stem loops** in a sequence:

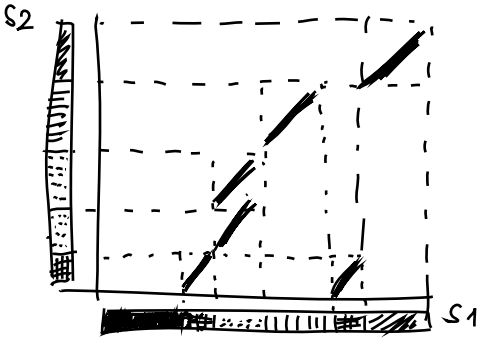
Stem loops can appear in RNA and 1 strained DNA sequences and can have for example regulatory roles. Stem loops are constructed from regions that are reverse complements of each other in the nucleotide sequence. These regions can connect and form two strained structures.

Stem loops can be found in a dotplot if we compare a sequence with its reverse complement (complement sequence of the original but in the reverse order). In this case the stem arm sequences of the stem loop appear around the trailing diagonal of the plot.





I.1.: Draw a dot-plot manually for the following two protein sequences! Different patterns mean different domains.



I.2.: Analyse the following DNA sequences with dot-plots!

sequence 1:
gtgacatcagagggtgggcacgctcctcccagctccctgtggcccagtcagaatctcagcctgaggacggtgttggcttcggcagccccgagata
catcagagggtgggcacgctcctccctccactcgcccctcaaacaaatgccccgcagcccatttctccaccctcatttgatgaccgcagattca
sequence 2:
cctcctgtgtccctctgcctcgccgctgttccggaacctgctctgcgcggcacgtcctggcagtggggcaggtggagctgggcgggggccctggt
gcaggcagcctgcagcccttggccctggaggggtccctgcagaagcgtggcattcatcagagggtgggcacgctctgcgaagagacgtccctgg

Use the following EMBOSS website: http://www.bioinformatics.nl/cgi-bin/emboss/dotmatcher

1. Paste the sequences in the boxes!
2. Use the default EDNAFULL scoring matrix (it handles all kinds of DNA substitutions)! http://rosalind.info/glossary/dnafull/
3. Set the window size and the threshold!
4. You can make titles and axis labels to your plot.
5. Push the "Run dotmatcher" button!

images of solution for I.2. included in subfolders

I.2.a.: What are the effects of the parameters to the plot? What happens in case of big and small window size and threshold (cut-off score) value and what is the connection between them?

In case of small window size short matches were discovered that were not discovered in the case of large window. Threshold value controls the number of matches

I.2.b.: Find the similar regions in the two sequences!

sequence 1: 145 - 190 ↔ sequence 2: 0-46
sequence 1: 145 - 190 ↔ sequence 2: 80 - 135

I.2.c.: Try to find repeats in the sequences!

sequence 1: 10 - 50 ↔ 60 - 100 ↔ 110 - 130 ↔ 120 - 150
sequence 2: eg. 0 - 30 ↔ 60 - 80 ↔ 70 - 110 etc

I.2.d.: Try to find stem loops in the sequences!
You can use the following website for generating the reverse complement sequence: http://reverse-complement.com/ (Usually nearly perfect match is required in stem loops in order to stabilise their structure.)

II. Pairwise Alignment:

Pairwise alignment algorithms are similar to dotplots, they show the similarities between two sequences based on a given scoring schema. But these methods operate with window size = 1 and compare each positions of the 1st sequence with each positions of the 2nd and assign a score to each comparison.
At first, we need a scoring scheme, substitution matrix, which stores the values of matches and mismatches. The values of this matrix are usually based on evolutionary relations. For example BLOSUM62 is widely used matrix for proteins. Its values were calculated from the occurrences of amino acid pairs in the BLOCKS database, clustered at 62% level.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

We also need gap penalty values. Gaps can be handled in multiple ways: in the simplest case every gaps get the same value. A more realistic assumption is differentiating higher gap opening and a lower gap extension value, because losing a longer sequence part in one step is more likely than losing it by independent deletions of each amino acid or nucleotide.
Global alignment is used when we wish to compare the entire sequences. Local alignment is useful if we are interested in finding only the most similar region. There are multiple algorithms; we will work with the most simple but still really useful ones.

Global alignment: Needleman-Wunsch algorithm:

We will construct an *F* matrix recursively where *F(i,j)* is the score of the best alignment between $x_{1\ldots i}$ and $y_{1\ldots j}$, $s(x_i,y_j)$ is the score value for aligning x_i and y_j

$$F(0,0) = 0$$

$$F(i,j) = \max\begin{cases} F(i-1,j-1) + s(x_i,y_j) & \leftarrow \text{alignment of residues} \\ F(i-1,j) - d & \leftarrow \text{gap in sequence y} \\ F(i,j-1) - d & \leftarrow \text{gap in sequence x} \end{cases}$$

The score of the optimal alignment between x and y is the lower right corner value of F matrix.

Example:
We have 2 protein sequences:
x = HEAGAWGHEE
y = PAWHEAE
We use linear gap penalty: d = 4
and the BLOSUM62 scoring schema for these examples (s).

1. The F matrix (scores):

	-	H	E	A	G	A	W	G	H	E	E
-											
P											
A											
W											
H											
E											
A											
E											

2. F(0,0)=0
3. Fill the 1st row and the 1st column with gap penalties:

	-	H	E	A	G	A	W	G	H	E	E
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40
P	-4										
A	-8										
W	-12										
H	-16										
E	-20										
A	-24										
E	-28										

4. Fill the matrix based on the rule until you get the final result:

	-	H	E	A	G	A	W	G	H	E	E
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40
P	-4	-2									
A	-8										
W	-12										
H	-16										
E	-20										
A	-24										
E	-28										

3 possible movements:

Vertical:

score + gap penalty = -4 + -4 = -8

Horizontal:

score + gap penalty = -4 + -4 = -8

Diagonal:

score + mismatch score = 0 + -2 = -2

The maximum is: -2

Value from the substitution matrix (H vs. P)

	-	H	E	A	G	A	W	G	H	E	E
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40
P	-4	-2	-5	-9	-13	-17	-21	-25	-29	-33	-37
A	-8	-6	-3	-1	-5	-9	-13	-17	-21	-25	-29
W	-12	-10	-7	-5	-3	-7	2	-2	-6	-10	-14
H	-16	-4	-8	-9	-7	-5	-2	0	6	2	-2
E	-20	-8	1	-3	-7	-8	-6	-4	2	11	7
A	-24	-12	-3	5	1	-3	-7	-6	-2	7	10
E	-28	-16	-7	1	3	0	-4	-8	-6	3	**12**

Score of the alignment

5. To get the alignment we have to backtrack with which path we have reached this value. We always have to store a pointer showing which one we have chosen from the three possibilities.

	-	H	E	A	G	A	W	G	H	E	E
-	0	← -4	← -8	← -12	← -16	← -20	← -24	← -28	← -32	← -36	← -40
P	↑ -4	↖ -2	↖ -5	← -9	← -13	← -17	← -21	← -25	← -29	← -33	← -37
A	↑ -8	↑ -6	↑ -3	↖ -1	↖ -5	↖ -9	← -13	← -17	← -21	← -25	← -29
W	↑ -12	↑ -10	↑ -7	↑ -5	↑ -3	↑ -7	↖ 2	↖ -2	← -6	← -10	← -14
H	↑ -16	↑ -4	↑ -8	↑ -9	↑ -7	↑ -5	↑ -2	↑ 0	↖ 6	← 2	← -2
E	↑ -20	↑ -8	↑ 1	↑ -3	↑ -7	↑ -8	↑ -6	↑ -4	↑ 2	↖ 11	← 7
A	↑ -24	↑ -12	↑ -3	↑ 5	↑ 1	↑ -3	↑ -7	↑ -6	↑ -2	↑ 7	↖ 10
E	↑ -28	↑ -16	↑ -7	↑ 1	↑ 3	↑ 0	↑ -4	↑ -8	↑ -6	↑ 3	↖ 12

↖ Align amino acids from sequences

← Insert gap in the vertical sequence

↑ Insert gap in the horizontal sequence

The result:

```
HEAGAWGHE_E
_PA__W_HEAE
```

Local alignment: Smith-Waterman algorithm:

The algorithm is really similar to the NW. The differences:

1. The score values have a lower bound = 0, so the new rule is:

$$F(0,0) = 0$$
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_i) \\ F(i-1,j) - d \\ F(i,j-1) - d \\ 0 \end{cases}$$

2. At backtracking we start from the largest value of F, it is the score of the alignment.
3. We backtrack until we reach a 0 value.

It gives the best single local match between the two sequences.

Example:

We have 2 protein sequences:

x = HEAGAWGHEE

y = PAWHEAE

We use linear gap penalty: d = 6

and the BLOSUM62 scoring schema for these examples (s).

	-	H	E	A	G	A	W	G	H	E	E
-	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	-0	0	0	0
A	0	0	0	4	0	4	0	0	0	0	0
W	0	0	0	0	2	0	15	9	3	0	0
H	0	8	2	0	0	0	9	13	17	11	5
E	0	2	13	7	1	0	3	7	13	22	16
A	0	0	7	17	11	5	0	3	7	16	21
E	0	0	5	11	15	10	4	0	3	12	21

The result:

AWGHE

AW_HE

These algorithms can be programmed efficiently using the concept of **dynamic programming**. It is a method of solving complex problems by breaking them down into simpler subproblems. We solve each subproblem only once, store the results and then combine them to get the overall solution. This method is useful because:

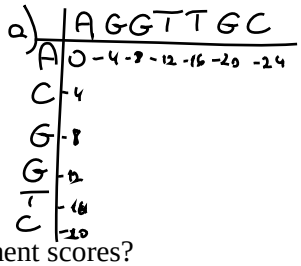
- can handle exponentially growing data
- easy to optimise
- fast
- usually can be parallelised

disadvantage:

- high memory usage

Here, we calculated the elements of the F matrix only once, stored the elements in the matrix and reused them when one of the neighbours had to be calculated, that why it was dynamic programming.

II.1.: Construct manually the **a) global** and the **b) local** pairwise alignment of the following 2 DNA sequences!
Gap penalty: -1; Scoring schema: mismatch: -1; match: +1



sequences: AGGTTGC
ACGGTC

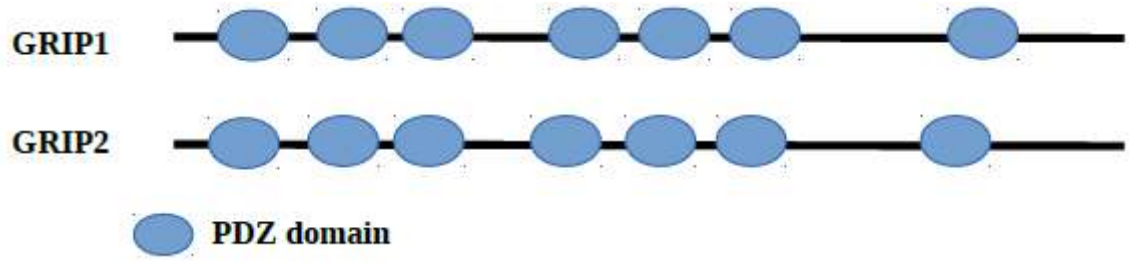
What are the results of the alignments and what are the alignment scores?

II.2.: Compare the following protein sequences with pairwise aligners (Needleman-Wunsch or Smith-Waterman) of the EMBOSS website: http://www.ebi.ac.uk/Tools/emboss/
1. Upload the sequence files.
2. You can see the details of the scoring under the "More options..." button. You do not need to change the default settings.
3. Push the "Submit" button and wait for the result of the alignment.

II.2.a.: Here are two virus protein sequences. Judging by the results of a pairwise alignment, can they be related? Is local or global pairwise alignment the better choice to compare the sequences?

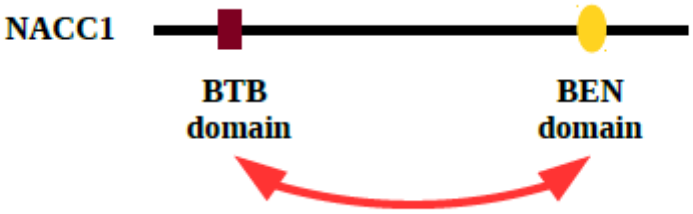
virus sequence 1:
PIFLSMFGRAGRNGAKGPRGRRRSPRPPGGSSMTPGDDGNQGPRGPGEQRDQPDQMDPLV
HPVTSVRSGPWERLGLRGRGESGVLRETLEMWAGQEGLISRVRDDPRESEVRPVTGVQTV
WPE
virus sequence 2:
MPFYIRSDLGPRRAVRGPRFTVPDPKPPPDPAHPLDGTDNVMMAFPKFKPYGFFAYNPWG
PIFLSMFGMAGRNGAKGPRGRRRSPRPLGESSMTPRGRRRPGVQGSRGAEGPAGPDGPAG
PFGPSGDVGPMGAAGSQGRRGIRGPQGDPGAVGRTGGVDLKGPGGPAGDMGPAGPRGPAG
EAGFVAPESVDMIPPVGFTSATVSAATLNSSKVGPVGDQGIRGPTGPSGAPGSQGPDRDV
GGMGPEDTKGDDGPVGPKGPQGATIF

II.2.b.: In the 2b1_grip.fasta and 2b2_grip.fasta files you can find 2 sequences in fasta format. One of them is the 3rd PDZ domain of the GRIP1 protein, and the second one is the middle part of the GRIP2 protein. We would like to find the 3rd PDZ domain of GRIP1 in GRIP2 and see how similar the two domains are. Which pairwise aligner would you use to find out? Try it!



https://en.wikipedia.org/wiki/PDZ_domain

II.2.c.: In the 2c1_nacc1.fasta and 2c2_nacc1.fasta files you can find 2 proteins with the same 2 domains but the domains are in switched position. Use pairwise aligners to compare the sequences. What do you experience? Does the alignment show that the domains are the same just in different places?


NACC1
BTB
domain
BEN
domain