

Pairwise Sequence Comparison

Dotplots, scoring matrices, dynamic programming

Ian Hoffecker

ian.hoffecker@ki.se

Department of Medical Biochemistry &
Biophysics

Karolinska Institutet, Stockholm, Sweden

Background:

- ♦ Sequencing techniques from the 1970s
- ♦ Full genomes:
 - ♦ drosophila, human, caenorhabditis, saccharomyces, various bacteria, viruses...
- ♦ Databases
 - ♦ GenBank, UniProt, etc...
 - ♦ Many sequences available, much unknown about function of proteins
- ♦ Converting existing data into useful biological knowledge is today's challenge

OUTLINE – **analyzing matches** & **aligning sequences**

1. **Dotplots** – comparing sequence similarity visually
 - Window size
 - Tolerance
2. **Score Matrices** – quantifying sequence similarity
 - PAM
 - BLOSUM
 - Pairwise distance, hierarchy, and information theory
3. **Alignment types**
4. **Dynamic Programming** – break problem into small ones
 - Global optimal alignment - Needleman-Wunsch
 - Local optimal alignment - Smith-Waterman

INTRODUCTION

- ♦ Reasons for comparing 2 sequences:
 - ♦ Determine if a gene is protein coding
 - ♦ Determine if genes are of common descent – homology
 - ♦ Infer structure
 - ♦ Infer function
- ♦ Sequences:
 - ♦ DNA
 - ♦ RNA
 - ♦ Protein

INTRODUCTION

- ♦ What differences are we likely to encounter?
 - ♦ Point mutations
 - ♦ Insertions/deletions (indels)
 - ♦ Fusion of sequences
 - ♦ Duplication
- ♦ What processes are they?

2 Sequences

- are they different?
- how different?
- and in what way?

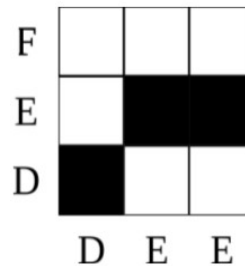
```
sequence_11 =
'MPIGSKERPFTFEIFKTRCNKADLGPISLNNFEELESAPPYNSPEAESEHKNNYNEPNLFTKPRKPSYQNLASTPIIFKEQGLTLPYQSPVKELDKFLDLGRNVPNSRHKSLRTVTKMKDQADVSCPLLNSLSESPVQLQCHTVTPQRKDSVCGSLFHTPKFVKGRQTPKHISESLGAEVDPMSWSSSLATPPTLSSSTVLIVRNEEASETVFPHDTHAVKSYFSNH
DESLKNDRIFASVDSENTNQREAAASHGFGKTSNGSFKVNSCKDHGKSMNVLDEVEYTVDTSEEDSFLCFKCRTKNLQKVRTSKTRKXIFHEANADECESKNQVKEKYSFVSEVEPNDDTDLSDNVAHQKPFESGSDKISKEVWPSLACEWSQLTSLGNGAQMEKIPLHISCCQDISEKDLDTENKRRKDFLTSENSLPRISLSPKSEKPLNEETVNNRDEEQH
LESHTDCLAVKQAIISGTSVAVSSFGQIKKSIIFRIRESPKETFNASFSGHMTDPNFKKETEASEESGLIEHTVCSQKEDSLCPNLIDNGSNPATTQNSVALKNAGLISTLKKKTKNFYIAIHDETFYGGKIPKQDKSELINCSAQFEANAFEAPLTFANADGSLHSSVKRSCQNDSEPTLSLSSFGTILRKCNRNETCSNNTVISQDLDYKAKCNKELQLFITPEADSL
CLQEGQCNDPKSKKVSIDKEEVLAAACHPVQHSKVEYSDDTDFQSQKXNLDYHENASTLILTPTSKDVLSNLMVSRGKESYKMSDKLGNMYESDVELTKNIPMEKNQDVCALNENYKINVELLPEKYMRVAVSPSRKVVQFNQNTLNVIQNKQEEITSKITVNPDSLEELFSDNENNFFVQVANERNLALGNTKELHETDLTCVNEPIFKNISTMVLVYDGTGDKQATQVSIKDDL
VYVLAENKNSVKQHKMTLGGDLKSDLSLNDIKPEKINDYMNKAGLGPISNHSFGSFRATSNKEIKLSEHNIKSKMFKDIEEQYPTSLACVENVTLALDNQKLSKPQSIINTVSAHLQSSVWSDCKNSHITPQMLFSKQDFNSNHLTPSQKAEITELSTLIEESGQFEFTQFRKPSYILQKSTFEVPENQMTLKTTSSEECRDLHVMNAPSIGQVDSKQFEG
TVEIKRRFAGLLKNDCKNSASGVLTDENEVGRFYSAGHTKLNWSTEALQAVKLFSDIENISEETSAEVHPISLSSKCHDSVSMFKIEHNHDKTVSEKNNKQLLQNNIEMTTGTVEEITENYKRNTEENEDNKYTAASRNSHLEFDGSDSKNDTVIHKDETDLFTDQHNIKCLLGGQFMKEGNTQIKEDLSDLTFLVAKAQEACHGNTSNKEQLTATKTEQNTKDF
ETSDDTFFQTAGSKNIVSAKESFNKINWFQDKPEELHNFSLNSELHSDIRKNKMDIISYEETDVIKWKILKESVVPVGTGQNLVTFQGGPERDEKIKEPTLLGHFTASGKVKYIAKESLDKVKNLFDEKEQSTSEITFSHQWAKTKYREACKDLELACETIETAAPKCEKQNSLNDKXNLSIETVPPKLLSDNLCRQENLTKSIFLKVYKHENVEKETAQSPATCYTNQ
SPYSVIENSALAFYTSRKTSSVSQTSLLLEAKKMLREGIFDQGERINTADYVGNLYENNSNISTIAENDKIHLEKQDITYLSNHSMSNSYSYHSDVEYNDSTRILSKNMLDSGIEPVLKNVEDQKNTSFVKVSNVMDANAYPQTVNEDIACEELVTSSSPCKNKNAIKLSISNSNFFVGPAPFIRASGKIVCVSHETIKKVKDIFDTSFKVTKENNEKSKICQTKIMAGYEL
ALDDSEDLHNSLNDDECSHSHKVFADIQSEELQHNNQMSGLKVKISIPDCVLETSIDICCKSLGKLKHSVSSANTCGIFSTASGKSVQVSDASLQNRQVSEIEDTSKQVSKVLFKSNHSDQLTRENTAIRTEPELISQKGFYNNVNSAFSGFSTASGKQVSLLESSLHKVGLVEEFDLIRTEHSLHYSPTSRQNSKILPRVDKRNPECHVNSEMEKTKSKEFKL
SNHLNVEGGSSENNHSIKVSPYLSQFQDQKQLVLGTVKSVLENHVLGKEQASPKNKVMEIGKTEFTSDVPVKNTIIEVCSYSDSEINYEFEAVEAKAFMEDDELTDKLSHATHSLFTCPENIEHMLSNRSRIGKRRGEPLILVGEPSIKRNLNEFDRIIEHQEQLKASKSPDGTIKDORLFFMHVSLLEPITCVPPFRTRKERQEQNPNIFAPGQEFLSKSHLYEHLTLE
KSSNLAVSGHPYQVASTRNEKMRHLITTRGPTKVFVPPFKTSHFRVQECVRNINLEENRQKQINDGHGSDSKNINDNEIHQFNKNSNQAAAVTFCKEEPELDTLITLQNRADQMRKIKKQRQVFPQPSGLYLAKTSTLPRISLKAAGVQVPSACSHKQLYTYGSKVCHIKINSKNAESFQHTEDYFGKESLWTKGKQLADGGWLPISNDGKAGKEEYFRALCD
TPGVDPKLISRIVNHYRWIILWLAAMECAFPEFANRCLSPERVLQLKYRYDTEIDRSRRAIKIMERDDTAKTLLVCSVDSIISLANISETSSNKTSADTQKVAIEITDGYAVKADLPPPLAVLKNRGLTVGQKILHGAELVGSPPACTLPEAPESLMLKISANSTRPARWYTKLGGFPDRPFPPLPSSLSDGGHVCVDVLIQRAYPIQMEKTSGLYIFRN
EREKKEAAKYVEAQKRLREALFTKIQEIEEHEENTTKPYLPSRALTRQQVRLQDGAELYEAVKNAADPAYLEGYFSEEQRALNHNHRQMLNDKQAOIQLEIRKAMESAQEQGLSRDVTVMKLRIVSYSKKEKDSVLSIWRPSSDLYSLTEGKRYRIYHLATSKSKSSEKSERANIQAATKTYQQLPVSDIELFQTYQPRELHFSKFLDPDFQPSCEVDLIGFVVS
VVKTKGLAPFVYLSDECYNLLAIKFIWDLNEDIKPHMLIAASNQWRPESKSGLLTLFAGDFSVFASAPKEGHFQETFNKMKNTVENIDILCNEAENKLMHLLHANDPKWSTPKDCTSGPYTAQIIPGTGNKLMSSPNCIEYQSPSLCMAKRKSVESTVPSAQMTSKCKSCKEKEIDDOKNCKRRALDFLRLPLPPVSPICITFVSPAQAQAFQPPRSCTGYETPIKKEEL
NSPQMTFFKFNIESLLESNIADEELALINTQALLSGSTGEKQFISVSESTRAPTSSSEYDLRLKRRCTSLIKEQESSQASTEECEKKNQDITITTKKYI'
```

```
sequence_12 =
'MPIGSKERPFTFEIFKTRCNKADLGPISLNNFEELESAPPYNSPEAESEHKNNYNEPNLFTKPRKPSYQNLASTPIIFKEQGLTLPYQSPVKELDKFLDLGRNVPNSRHKSLRTVTKMKDQADVSCPLLNSLSESPVQLQCHTVTPQRKDSVCGSLFHTPKFVKGRQTPKHISESLGAEVDPMSWSSSLATPPTLSSSTVLIVRNEEASETVFPHDTHAVKSYFSNH
DESLKNDRIFASVDSENTNQREAAASHGFGKTSNGSFKVNSCKDHGKSMNVLDEVEYTVDTSEEDSFLCFKCRTKNLQKVRTSKTRKXIFHEANADECESKNQVKEKYSFVSEVEPNDDTDLSDNVAHQKPFESGSDKISKEVWPSLACEWSQLTSLGNGAQMEKIPLHISCCQDISEKDLDTENKRRKDFLTSENSLPRISLSPKSEKPLNEETVNNRDEEQH
LESHTDCLAVKQAIISGTSVAVSSFGQIKKSIIFRIRESPKETFNASFSGHMTDPNFKKETEASEESGLIEHTVCSQKEDSLCPNLIDNGSNPATTQNSVALKNAGLISTLKKKTKNFYIAIHDETFYGGKIPKQDKSELINCSAQFEANAFEAPLTFANADGSLHSSVKRSCQNDSEPTLSLSSFGTILRKCNRNETCSNNTVISQDLDYKAKCNKELQLFITPEADSL
CLQEGQCNDPKSKKVSIDKEEVLAAACHPVQHSKVEYSDDTDFQSQKXNLDYHENASTLILTPTSKDVLSNLMVSRGKESYKMSDKLGNMYESDVELTKNIPMEKNQDVCALNENYKINVELLPEKYMRVAVSPSRKVVQFNQNTLNVIQNKQEEITSKITVNPDSLEELFSDNENNFFVQVANERNLALGNTKELHETDLTCVNEPIFKNISTMVLVYDGTGDKQATQVSIKDDL
VYVLAENKNSVKQHKMTLGGDLKSDLSLNDIKPEKINDYMNKAGLGPISNHSFGSFRATSNKEIKLSEHNIKSKMFKDIEEQYPTSLACVENVTLALDNQKLSKPQSIINTVSAHLQSSVWSDCKNSHITPQMLFSKQDFNSNHLTPSQKAEITELSTLIEESGQFEFTQFRKPSYILQKSTFEVPENQMTLKTTSSEECRDLHVMNAPSIGQVDSKQFEG
TVEIKRRFAGLLKNDCKNSASGVLTDENEVGRFYSAGHTKLNWSTEALQAVKLFSDIENISEETSAEVHPISLSSKCHDSVSMFKIEHNHDKTVSEKNNKQLLQNNIEMTTGTVEEITENYKRNTEENEDNKYTAASRNSHLEFDGSDSKNDTVIHKDETDLFTDQHNIKCLLGGQFMKEGNTQIKEDLSDLTFLVAKAQEACHGNTSNKEQLTATKTEQNTKDF
ETSDDTFFQTAGSKNIVSAKESFNKINWFQDKPEELHNFSLNSELHSDIRKNKMDIISYEETDVIKWKILKESVVPVGTGQNLVTFQGGPERDEKIKEPTLLGHFTASGKVKYIAKESLDKVKNLFDEKEQSTSEITFSHQWAKTKYREACKDLELACETIETAAPKCEKQNSLNDKXNLSIETVPPKLLSDNLCRQENLTKSIFLKVYKHENVEKETAQSPATCYTNQ
SPYSVIENSALAFYTSRKTSSVSQTSLLLEAKKMLREGIFDQGERINTADYVGNLYENNSNISTIAENDKIHLEKQDITYLSNHSMSNSYSYHSDVEYNDSTRILSKNMLDSGIEPVLKNVEDQKNTSFVKVSNVMDANAYPQTVNEDIACEELVTSSSPCKNKNAIKLSISNSNFFVGPAPFIRASGKIVCVSHETIKKVKDIFDTSFKVTKENNEKSKICQTKIMAGYEL
ALDDSEDLHNSLNDDECSHSHKVFADIQSEELQHNNQMSGLKVKISIPDCVLETSIDICCKSLGKLKHSVSSANTCGIFSTASGKSVQVSDASLQNRQVSEIEDTSKQVSKVLFKSNHSDQLTRENTAIRTEPELISQKGFYNNVNSAFSGFSTASGKQVSLLESSLHKVGLVEEFDLIRTEHSLHYSPTSRQNSKILPRVDKRNPECHVNSEMEKTKSKEFKL
SNHLNVEGGSSENNHSIKVSPYLSQFQDQKQLVLGTVKSVLENHVLGKEQASPKNKVMEIGKTEFTSDVPVKNTIIEVCSYSDSEINYEFEAVEAKAFMEDDELTDKLSHATHSLFTCPENIEHMLSNRSRIGKRRGEPLILVGEPSIKRNLNEFDRIIEHQEQLKASKSPDGTIKDORLFFMHVSLLEPITCVPPFRTRKERQEQNPNIFAPGQEFLSKSHLYEHLTLE
KSSNLAVSGHPYQVASTRNEKMRHLITTRGPTKVFVPPFKTSHFRVQECVRNINLEENRQKQINDGHGSDSKNINDNEIHQFNKNSNQAAAVTFCKEEPELDTLITLQNRADQMRKIKKQRQVFPQPSGLYLAKTSTLPRISLKAAGVQVPSACSHKQLYTYGSKVCHIKINSKNAESFQHTEDYFGKESLWTKGKQLADGGWLPISNDGKAGKEEYFRALCD
TPGVDPKLISRIVNHYRWIILWLAAMECAFPEFANRCLSPERVLQLKYRYDTEIDRSRRAIKIMERDDTAKTLLVCSVDSIISLANISETSSNKTSADTQKVAIEITDGYAVKADLPPPLAVLKNRGLTVGQKILHGAELVGSPPACTLPEAPESLMLKISANSTRPARWYTKLGGFPDRPFPPLPSSLSDGGHVCVDVLIQRAYPIQMEKTSGLYIFRN
EREKKEAAKYVEAQKRLREALFTKIQEIEEHEENTTKPYLPSRALTRQQVRLQDGAELYEAVKNAADPAYLEGYFSEEQRALNHNHRQMLNDKQAOIQLEIRKAMESAQEQGLSRDVTVMKLRIVSYSKKEKDSVLSIWRPSSDLYSLTEGKRYRIYHLATSKSKSSEKSERANIQAATKTYQQLPVSDIELFQTYQPRELHFSKFLDPDFQPSCEVDLIGFVVS
VVKTKGLAPFVYLSDECYNLLAIKFIWDLNEDIKPHMLIAASNQWRPESKSGLLTLFAGDFSVFASAPKEGHFQETFNKMKNTVENIDILCNEAENKLMHLLHANDPKWSTPKDCTSGPYTAQIIPGTGNKLMSSPNCIEYQSPSLCMAKRKSVESTVPSAQMTSKCKSCKEKEIDDOKNCKRRALDFLRLPLPPVSPICITFVSPAQAQAFQPPRSCTGYETPIKKEEL
NSPQMTFFKFNIESLLESNIADEELALINTQALLSGSTGEKQFISVSESTRAPTSSSEYDLRLKRRCTSLIKEQESSQASTEECEKKNQDITITTKKYI'
```

Dotplots

Dotplots

- 2D visualization of sequence similarity
- No convention for the order of sequence labeling
- If agreement between two elements/bases/amino acids exists – place a mark “dot”



Dotplots

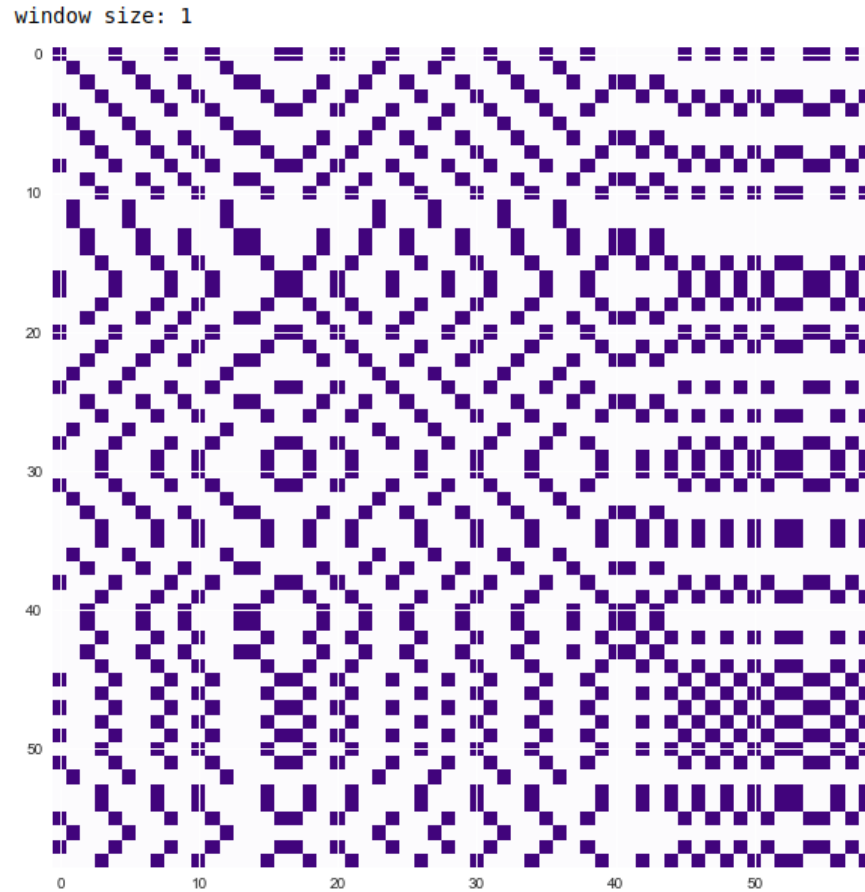
Dotplot versus linear inspection

- `sequence_1 = 'gctagctagtagcttaggatgatcgtacgtagctagctgattatagagagagaaggagaa'`
- `sequence_2 = 'gctagctagtaccttaggatgatcgtacgaagctaactgattatagagagagcaagcgaa'`

Dotplots

Dotplot versus linear inspection

- `sequence_1 = 'gctagctagtagcttaggatgatcgtagctagctgattatagagagagaaggagaa'`
- `sequence_2 = 'gctagctagctaccttaggatgatcgtagcgaagctaactgattatagagagagcaagcgaa'`



Dotplots

Noise depends on alphabet size and redundancy of bases

- DNA bases: 4
- Amino acids: 20
 - Codons – triplets of DNA bases
- Sliding window size as a filtering approach:

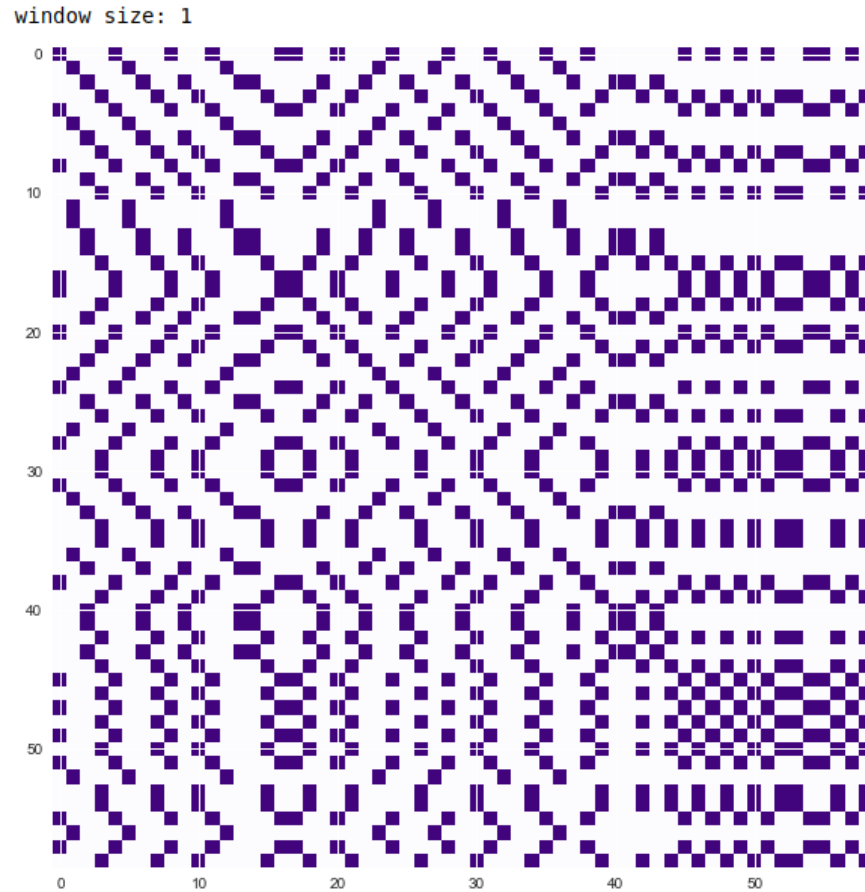


S	E	Q	V	E	N	C	E	S
S	E	K	V	E	N	S	E	R

Dotplots

Dotplot versus linear inspection

- `sequence_1 = 'gctagctagtagcttaggatgatcgtagctagctgattatagagagagaaggagaa'`
- `sequence_2 = 'gctagctagctaccttaggatgatcgtagcgaagctaactgattatagagagagcaagcgaa'`

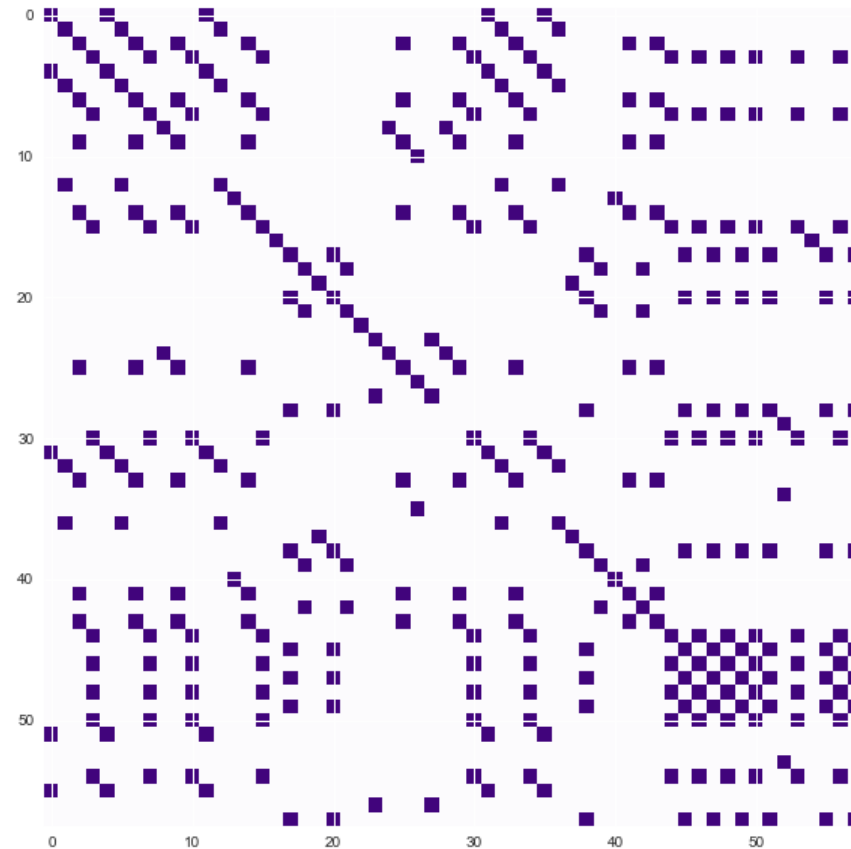


Dotplots

Dotplot versus linear inspection

- `sequence_1 = 'gctagctagtagcttaggatgatcgtagctagctgattatagagagagaaggagaa'`
- `sequence_2 = 'gctagctagctaccttaggatgatcgtagcgaagctaactgattatagagagagcaagcgaa'`

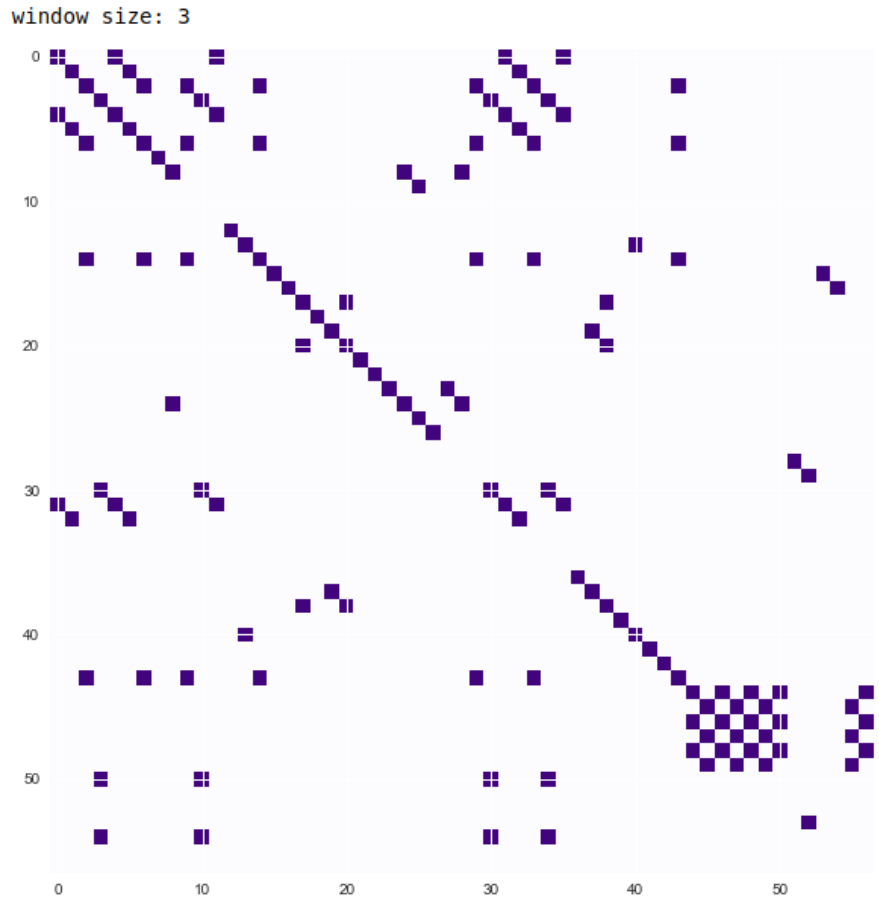
window size: 2



Dotplots

Dotplot versus linear inspection

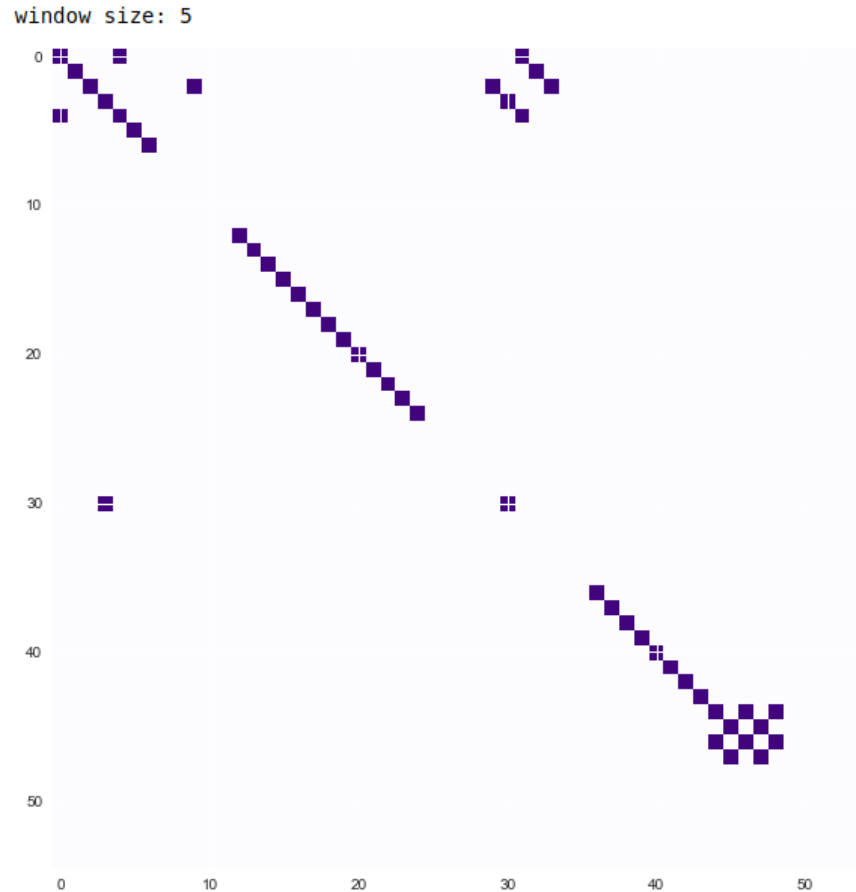
- sequence_1 = 'gctagctagtagcttaggatgatcgtagctagctgattatagagagagaaggagaa'
- sequence_2 = 'gctagctagtaccttaggatgatcgtagcgaagctaactgattatagagagagcaagcgaa'



Dotplots

Dotplot versus linear inspection

- sequence_1 = 'gctagctagtagcttaggatgatcgtagctagctgattatagagagagaaggagaa'
- sequence_2 = 'gctagctagtagcttaggatgatcgtagcgaagctaactgattatagagagagcaagcgaa'



Dotplots

Comparisons with different sizes – scanning

- sequence_4 = 'gatcgcgc'
- sequence_3 = 'gctagctagtgatcgcgcaccttaggatgatcgtgatcgcgcacgaagctaagatcgatcctgattatagaggatcgatcagagatcgatcgatcgatcgatcgcaagcgaa'

window size: 1



window size: 2



window size: 3



window size: 4



window size: 5



window size: 6



window size: 7



Dotplots

Tolerating mismatch

- Count the number of mismatches in the window:
- Max score is the window size
- Thresholding

```

window size: 8
[[8 0 1 ..., 0 0 1]
 [0 8 0 ..., 2 0 0]
 [1 0 8 ..., 0 2 0]
 ...
 [0 2 0 ..., 8 0 0]
 [0 0 2 ..., 0 8 0]
 [1 0 0 ..., 0 0 8]]

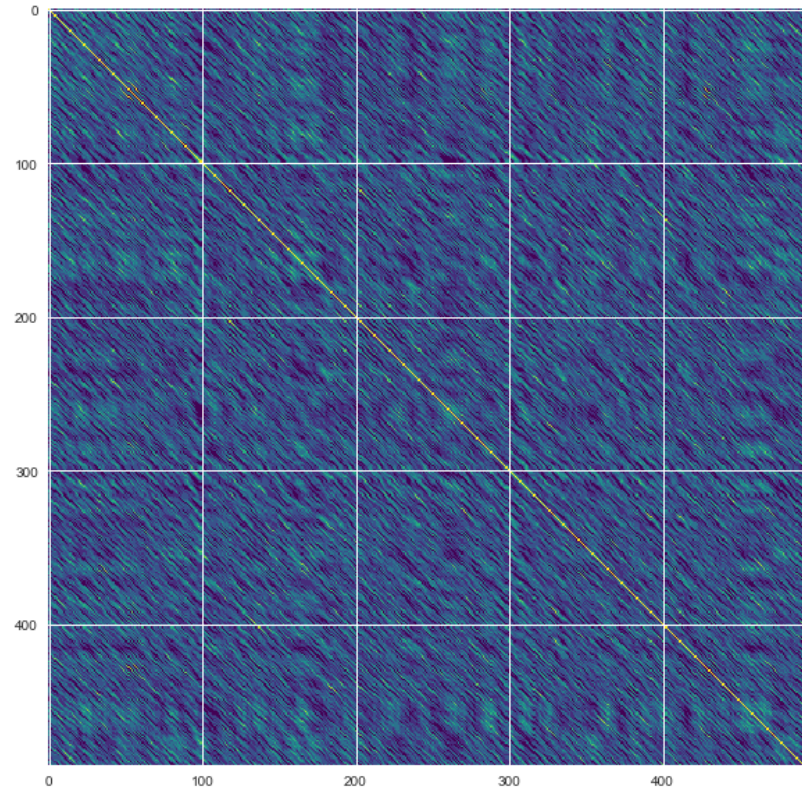
```



```

' CCCAGAAGCCGA ACTGGGCCAGACAACCCGGCGCTAACGCACTCAAAGCCGGGACGCGACGC
GACATATCGGCTAAGAGTAGGCCGGGAGTGTAGACCTTTGGGGTTGAATAAATCTGTCGTAGT
AACCGGCTTCAACGACCCGTACAGGTGGCACTTCAGGAGGGGCCCGCAGGGAGGAAGTTTTCT
GCTATTCGTGGCCGTTTCGTGGTAAGTGTGCGTTCCCTAGCCACTACAATTGTTTCTAAGCCG
TGTAATGAGAACAACCAACCATAGCGAATTGATGCGCCGCTCGGAATACCGTTTTGGCAAC
CCCTTACTAAGGCCATCGCGATTTTCAGGTATCGTGCATGTAGGGTTGGACCGCACGCATGTT
AAACTGCTGGCGAACCGCGATTCCACGACCGGTGCACGATTTAATTACGCCGACGTGACGACA
TTCCTGCTAATGCCTCACCCGCCGGACCGCCCTCGTGATGGGGTAGCTGGGCATGACCTT '

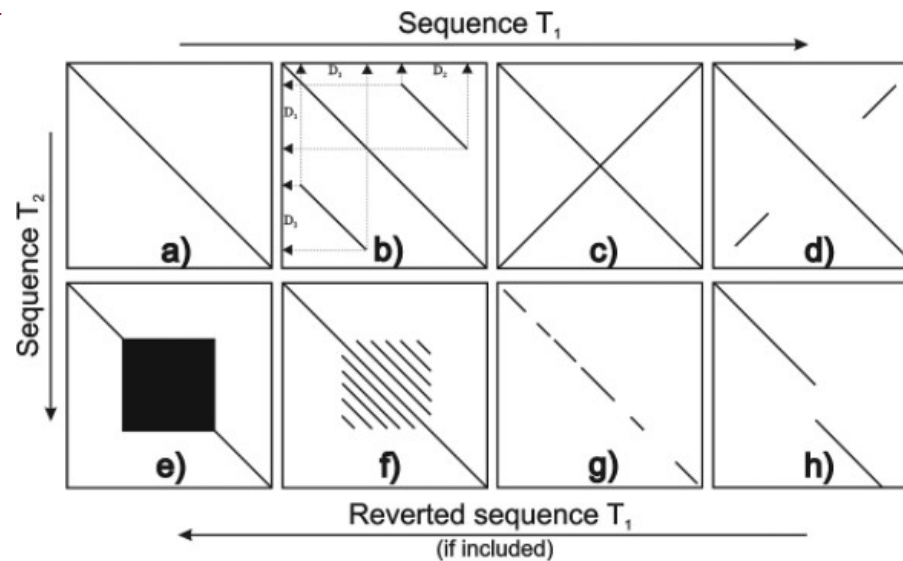
```



Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

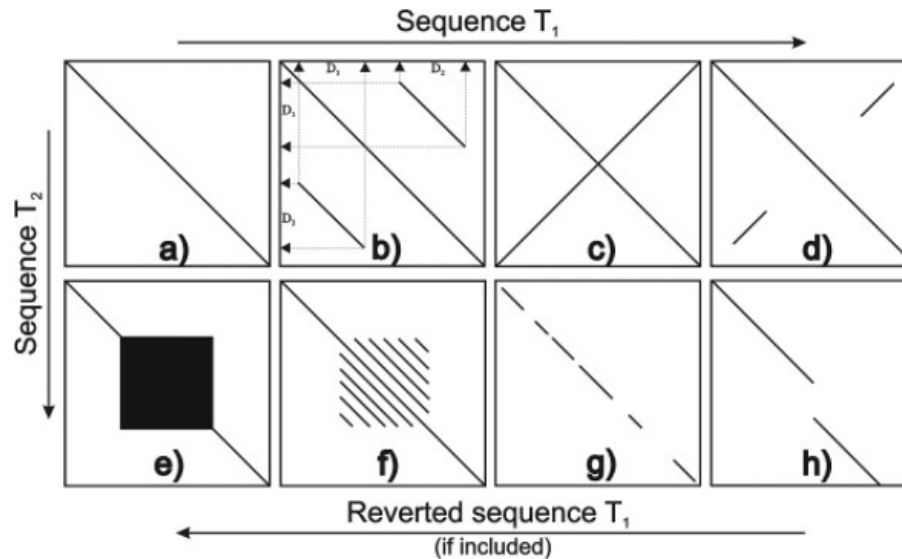


a. identity

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

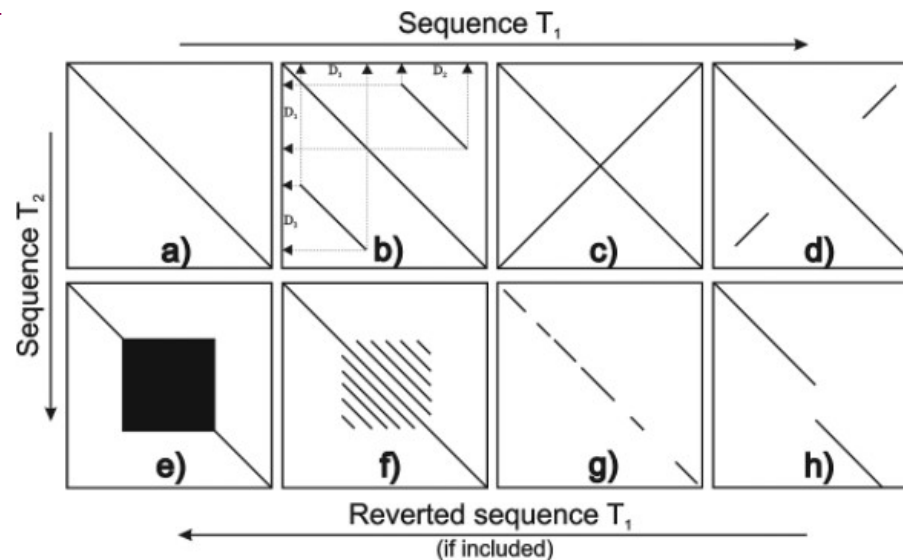


- a. identity
- b. duplication

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

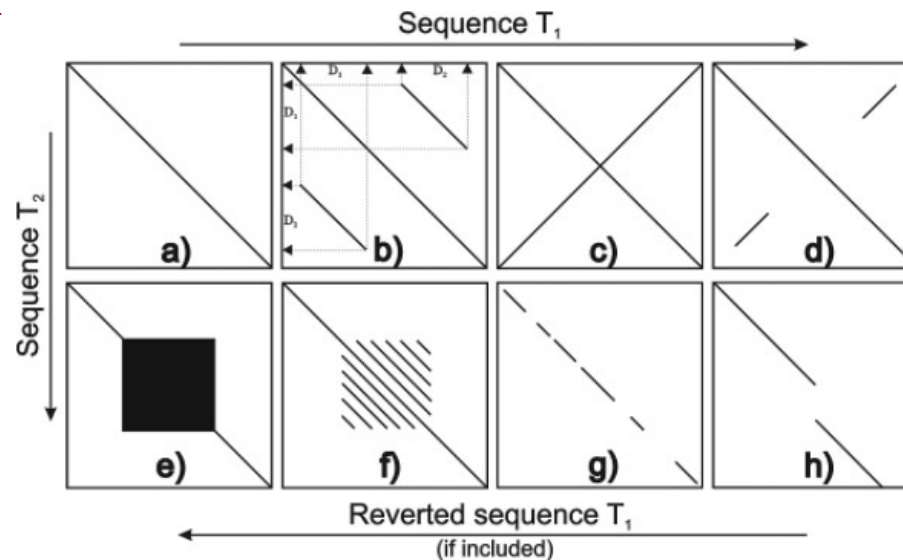


- a. identity
- b. duplication
- c. palindrome

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

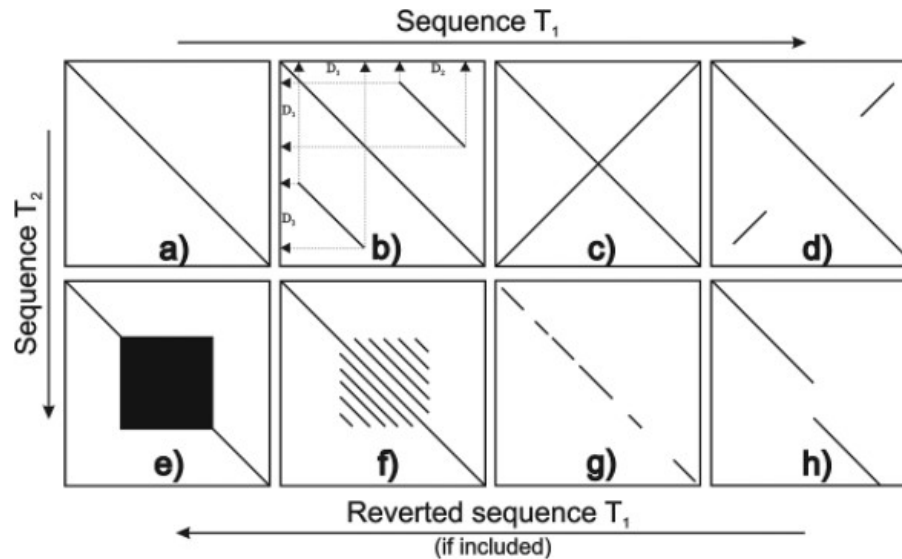


- a. identity
- b. duplication
- c. palindrome
- d. partial palindrome

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

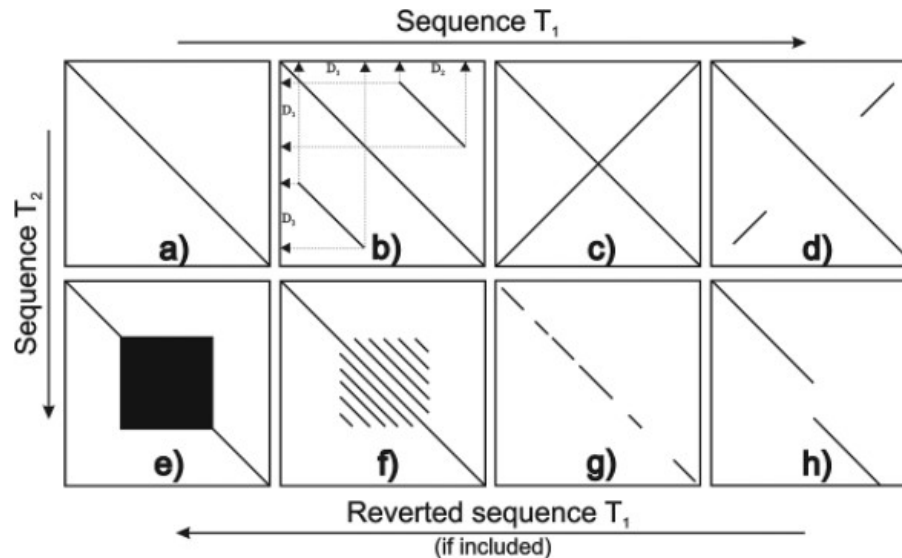


- a. identity
- b. duplication
- c. palindrome
- d. partial palindrome
- e. microsatellite repeats

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

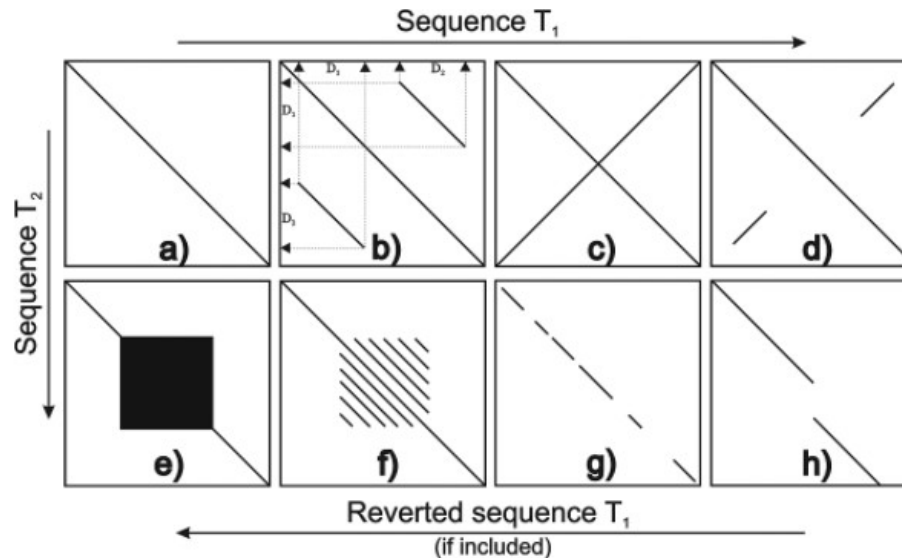


- a. identity
- b. duplication
- c. palindrome
- d. partial palindrome
- e. microsatellite repeats
- f. minisatellite repeats (patterns)

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

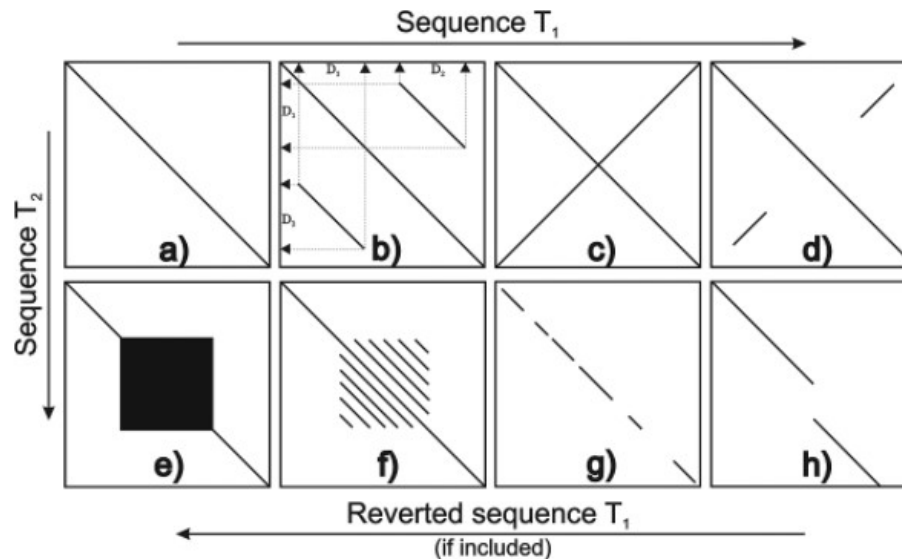


- a. identity
- b. duplication
- c. palindrome
- d. partial palindrome
- e. microsatellite repeats
- f. minisatellite repeats (patterns)
- g. homology

Dotplots

Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)

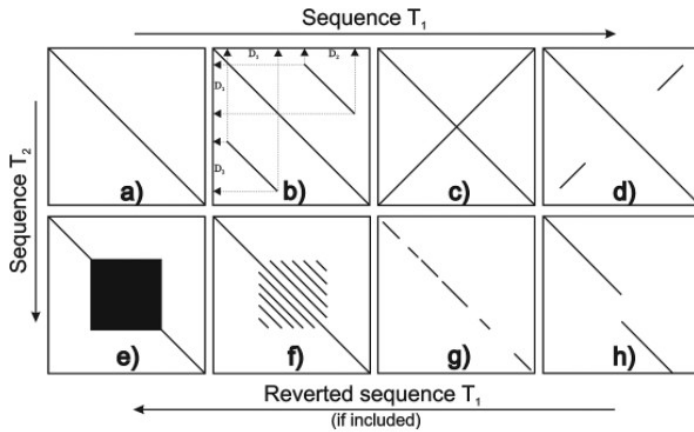


- a. identity
- b. duplication
- c. palindrome
- d. partial palindrome
- e. microsatellite repeats
- f. minisatellite repeats (patterns)
- g. homology
- h. insertion seq2, deletion seq1

Dotplots

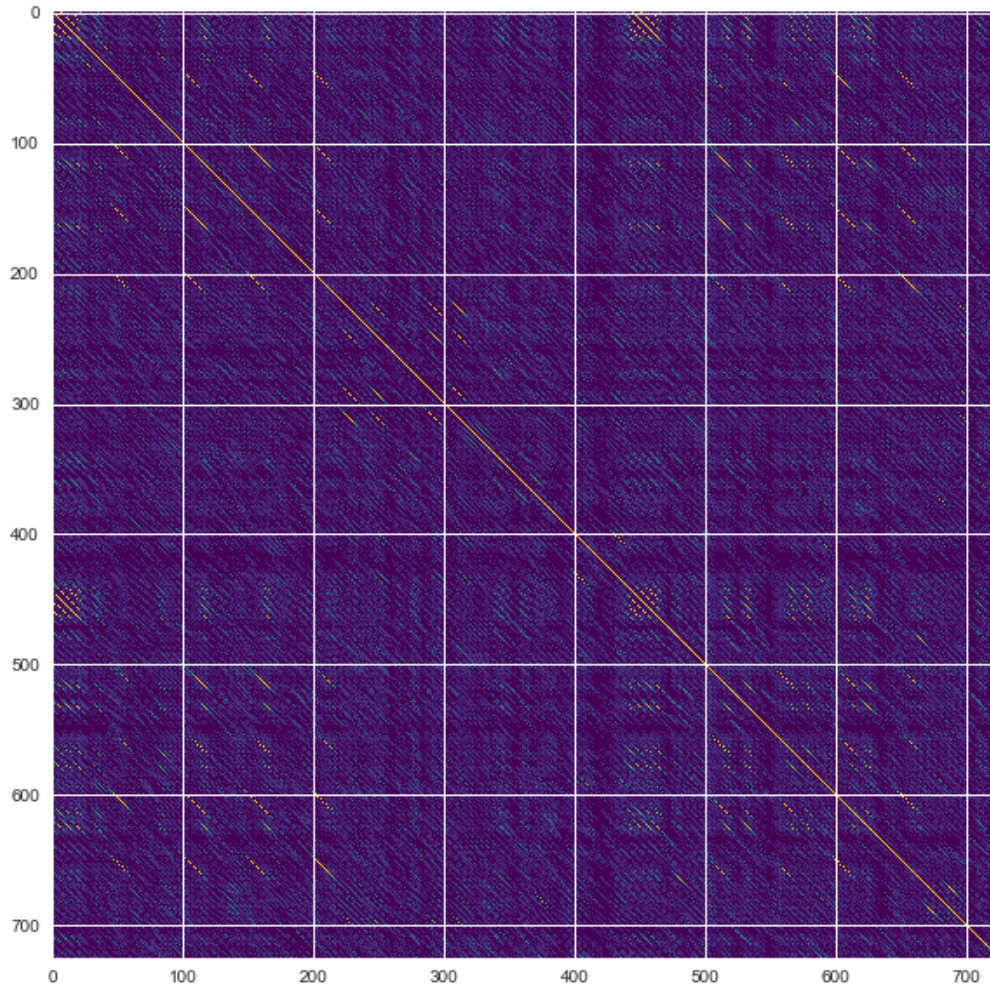
Dotplot zoo – interpretation by recognition of visual features

[HTTP://TINYURL.COM/6XYWQ34](http://tinyurl.com/6xywq34)



- a. Identity
- b. Duplication
- c. Palindrome
- d. Partial palindrome
- e. Microsatellite repeats
- f. Minisatellite repeats (patterns)
- g. Homology
- h. Insertion/deletion

sequence = 'I got, I got, I got, I got Loyalty, got royalty inside my DNA Cocaine quarter piece, got war and peace inside my DNA I got power, poison, pain and joy inside my DNA I got hustle though, ambition, flow, inside my DNA I was born like this, since one like this Immaculate conception I transform like this, perform like this Was Yeshuas new weapon I dont contemplate, I meditate, then off your fucking head This that put-the-kids-to-bed This that I got, I got, I got, I got Realness, I just kill shit cause its in my DNA I got millions, I got riches buildin' in my DNA I got dark, I got evil, that rot inside my DNA I got off, I got troublesome, heart inside my DNA I just win again, then win again like Wimbledon, I serve Yeah, thats'

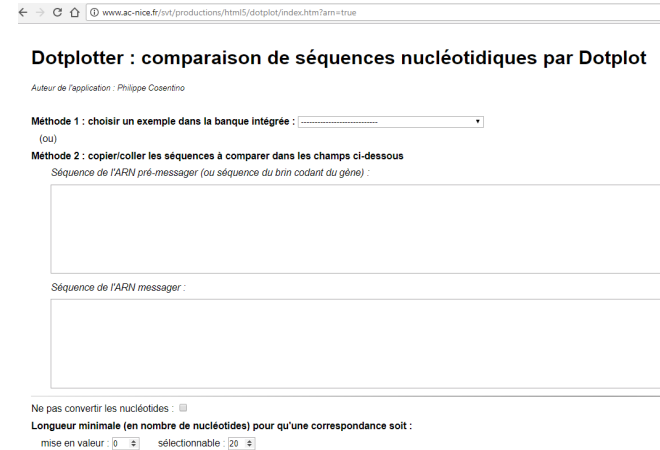


♦ Software options

- ♦ R packages:
 - ♦ Seqinr
 - ♦ Dotplot
- ♦ Python
 - ♦ [Github.com/kn-bibs/dotplot](https://github.com/kn-bibs/dotplot)
- ♦ Browser based
 - ♦ Dotplotter
 - ♦ Dotmatcher
- ♦ Downloadable
 - ♦ Synmap
 - ♦ Gepard

♦ Making your own

- ♦ ian's github:
 - ♦ [Github.com/Intertangler/bioinformatics_ith](https://github.com/Intertangler/bioinformatics_ith)



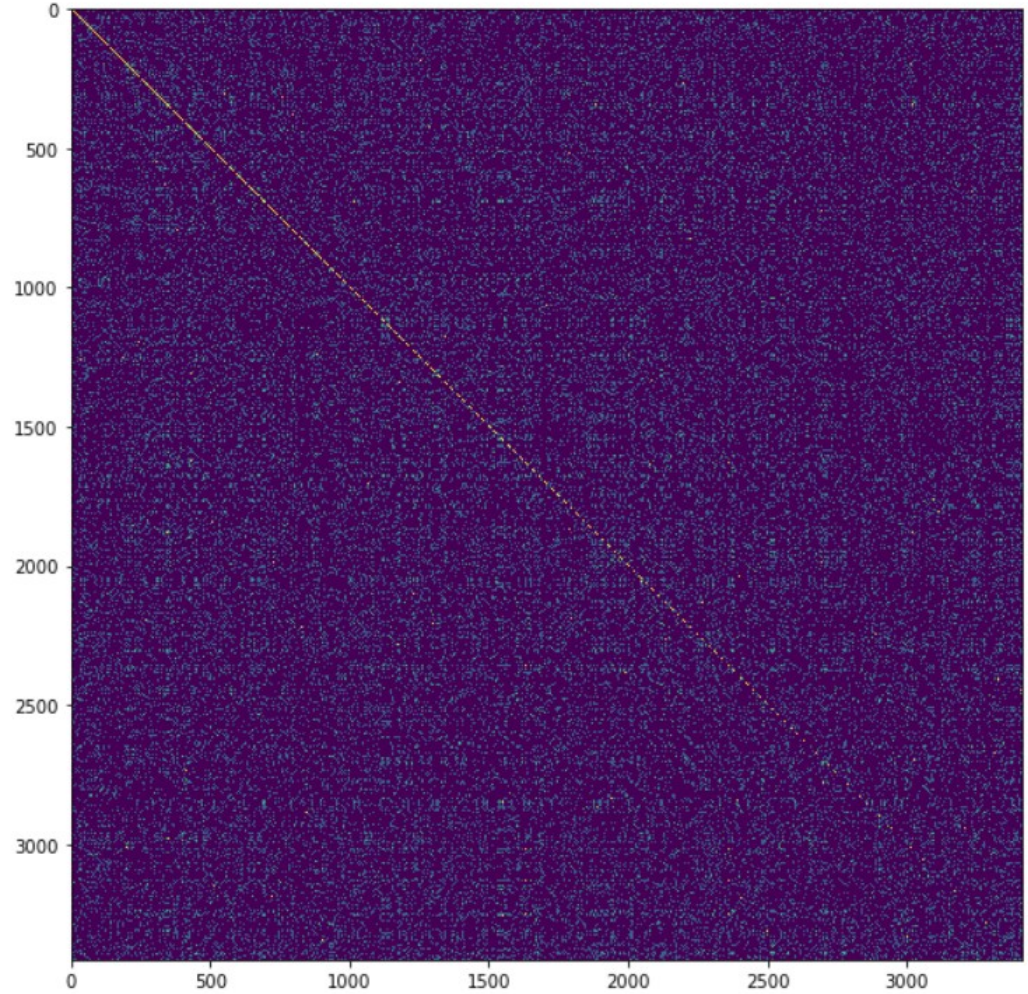
The screenshot shows a web browser window with the URL www.ac-nice.fr/irt/productions/html5/dotplot/index.htm?am=true. The page title is "Dotplotter : comparaison de séquences nucléotidiques par Dotplot". Below the title, it says "Auteur de l'application : Philippe Cosentino". There are two methods for inputting sequences: "Méthode 1 : choisir un exemple dans la banque intégrée" with a dropdown menu, and "Méthode 2 : copier/coller les séquences à comparer dans les champs ci-dessous". Under Method 2, there are two text input fields labeled "Séquence de l'ARN pré-messager (ou séquence du brin codant du gène) :" and "Séquence de l'ARN messager :". At the bottom, there are checkboxes for "Ne pas convertir les nucléotides" and "Longueur minimale (en nombre de nucléotides) pour qu'une correspondance soit :", with input fields for "mise en valeur" (set to 0) and "sélectionnable" (set to 20).

OUTLINE – **analyzing matches** & **aligning sequences**

1. **Dotplots** – comparing sequence similarity visually
 - Window size
 - Tolerance
2. **Score Matrices** – quantifying sequence similarity
 - PAM
 - BLOSUM
 - Pairwise distance, hierarchy, and information theory
3. **Alignment types**
4. **Dynamic Programming** – break problem into small ones
 - Global optimal alignment - Needleman-Wunsch
 - Local optimal alignment - Smith-Waterman

Scoring a match

Figure 23
 Figure 24



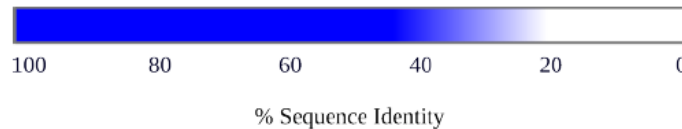
Assessing multiple candidates for an alignment

Example (Alignments)

1 2 3 4 5 6 7 8 9 10 11		1 2 3 4 5 6 7 8 9 10 11
T H I S L I N E - - -	or	T H I S - L I - N E - -
- - I S A L I G N E D		- - I S A L I G N E D

Scoring Matrices

- ▶ Percent identity (Sequence identity)
 - ▶ % identity = $\frac{\text{no. identical positions}}{\text{alignment length}} \times 100$
- ▶ Percent similarity (Sequence similarity)
 - ▶ % similarity = $\frac{\text{no. similar or identical positions}}{\text{alignment length}} \times 100$



- ▶ > 40% Sequence identity — Homology
- ▶ 20–40% Sequence identity — Homology probable
- ▶ < 20 % Sequence identity — Homology possible, but unlikely
- ▶ N.B. these limits are not absolute.

SCORE – How good is an alignment

- ▶ Two 100 aa proteins can be aligned in 10^{75} ways if gaps are allowed
- ▶ Use **alignment score** to measure the quality of alignments
- ▶ Calculate numerical value (score) for overall similarity and rank alignments

Optimal alignment best scoring alignment

Suboptimal alignment score slightly worse than best

The optimal alignment is not necessarily the **correct** alignment

Scoring Matrices

The substitution matrix – a linguistic analogy: biology vs biologi

$$\begin{bmatrix} S(a_1, a_1) & S(a_2, a_1) & \dots & S(a_n, a_1) \\ S(a_1, a_2) & S(a_2, a_2) & \dots & S(a_n, a_2) \\ \vdots & & & \\ S(a_1, a_n) & S(a_2, a_n) & \dots & S(a_n, a_n) \end{bmatrix}$$

identity-score matrix

Let's build the simplest type of score matrix

$$\begin{bmatrix} b & 1 & & & & & \\ g & 0 & 1 & & & & \\ i & 0 & 0 & 1 & & & \\ l & 0 & 0 & 0 & 1 & & \\ o & 0 & 0 & 0 & 0 & 1 & \\ y & 0 & 0 & 0 & 0 & 0 & 1 \\ & b & g & i & l & o & y \end{bmatrix}$$

Scoring Matrices

The substitution matrix – a linguistic analogy

now let's add a penalty for consant-related mismatches

$$\begin{bmatrix} b & 1 & & & & & \\ g & -1 & 1 & & & & \\ i & -1 & -1 & 1 & & & \\ l & -1 & -1 & -1 & 1 & & \\ o & -1 & -1 & 0 & -1 & 1 & \\ y & -1 & -1 & 0 & -1 & 0 & 1 \\ & b & g & i & l & o & y \end{bmatrix}$$

The score matrix

Now try scoring some different alignments.

-- biology

biologl--

$$\text{score} = S(b,o) + S(i,l) + S(o,o) + S(l,g) + S(o,i) = -1 + -1 + 1 + -1 + 0 = -2$$

biology

biologi

$$\text{score} = S(b,b) + S(i,i) + S(o,o) + S(l,l) + S(o,o) + S(g,g) + S(y,i) = 6$$

Note that maximum possible score with our matrix is 7.

The Meaning Behind Scores

- The random model assumes that mutations occur depending only on the amino acid frequency
 - e.g. $p(I \rightarrow L)$ is the same as $p(E \rightarrow L)$
- Nonrandom models assume a strong correlation between the residues in the alignment in question
- Odds ratio – the ratio of probability of the nonrandom vs the random models producing the alignment
- Logarithms are used to generate better behaved mathematics
 - Logarithms usually indicative of hierarchy or information (e.g. bits representing forks in a decision tree) and here we have a structural connection to phylogenetics or evolutionary changes

$$S = \sum_i \log \left(\frac{q_{a,b}}{p_a p_b} \right)_i = \sum_i (s_{a,b})_i$$

Types of scoring systems:

- ♦ Identity
 - ♦ Positive scores for identical residues only
 - ♦ Still used for nucleotides
- ♦ Similarity
 - ♦ Positive scores for similar residues
 - ♦ Based on chemical or mutational similarity
- ♦ Deriving scores
 - ♦ Theoretical, count mutations
 - ♦ Physicochemical properties
 - ♦ Empirical – evidence from evolution

PAM is an empirically calibrated model of evolution

- Multiple alignments
- Calculate the exposure of each amino acid type to mutation
 - Fraction of the residue type in the alignment multiplied by total number of mutations of any kind per 100 alignment positions
- Calculate the mutability of a residue type
 - Ratio of the total number of mutations in the data involving that residue divided by the total exposure of that residue
 - Reported relative to alanine – the relative mutability m_a
- Make a phylogenetic tree (family tree)
 - Mutations determine most likely positions in the tree
- Tabulate observed substitutions in matrix A, accepted point mutation matrix – a tally of mutations basically over a decided length of evolutionary time
- Assemble a mutation probability matrix M
- Create scoring matrix based on odds ratio of the model vs random chance

PAM in action



```
def compare_pam120(string1,string2,fulloutput=True):
    import numpy as np
    import matplotlib.pyplot as plt

    aa_index = {'C': 0, 'S': 1, 'T': 2, 'P': 3, 'A': 4, 'G': 5, 'N': 6, 'D': 7
                , 'E': 8, 'Q': 9, 'H': 10, 'R': 11, 'K': 12, 'M': 13, 'I': 14, 'L': 15
                , 'V': 16, 'F': 17, 'Y': 18, 'W': 19}

    PAM120 = np.array([[ 9]
                       ,[-1, 3]
                       ,[-3, 2, 4]

                       ,[-3, 1,-1, 6]
                       ,[-3, 1, 1, 3]
                       ,[-5, 1,-1,-2, 1, 5]

                       ,[-5, 1, 0,-2, 0, 0, 4]
                       ,[-7, 0,-1,-2, 0, 0, 2, 5]
                       ,[-7,-1,-2,-1, 0,-1, 1, 3, 5]
                       ,[-7,-2,-2, 0,-1,-3, 0, 1, 2, 6]

                       ,[-4,-2,-3,-1,-3,-4, 2, 0,-1, 3, 7]
                       ,[-4,-1,-2,-1,-3,-4,-1,-3,-3, 1, 1, 6]
                       ,[-7,-1,-1,-2,-2,-3, 1,-1,-1, 0,-2, 2, 5]

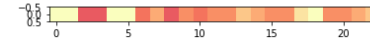
                       ,[-6,-2,-1,-3,-2,-4,-3,-4,-4,-1,-4,-1, 0, 8]
                       ,[-3,-2, 0,-3,-1,-4,-2,-3,-3,-3,-4,-2,-2, 1, 6]
                       ,[-7,-4,-3,-3,-3,-5,-4,-5,-4,-2,-3,-4,-4, 3, 1, 5]
                       ,[-2,-2, 0,-2, 0,-2,-3,-3,-3,-3,-3,-3,-4, 1, 3, 1, 5]

                       ,[-6,-3,-4,-5,-4,-5,-4,-7,-6,-6,-2,-4,-6,-1, 0, 0,-3, 8]
                       ,[-1,-3,-3,-6,-4,-6,-2,-5,-4,-5,-1,-6,-6,-4,-2,-3,-3, 4, 8]
                       ,[-8,-2,-6,-7,-7,-8,-5,-8,-8,-6,-5, 1,-5,-7,-7,-5,-8,-1,-1,12]])

    score = 0
    visual = np.zeros((1,len(string1)))
    for i in range(0,len(string1)):
        previous_score = score
        if aa_index[string1[i]] > aa_index[string2[i]]:
            score += int(PAM120[int(aa_index[string1[i]])][int(aa_index[string2[i]])])
        else:
            score += int(PAM120[int(aa_index[string2[i]])][int(aa_index[string1[i]])])
        if fulloutput == True:
            print score-previous_score, score, aa_index[string1[i]], aa_index[string2[i]], i, string1[i], string1[i]
        visual[0,i] = score-previous_score
    print "final score: ", score
    plt.imshow(visual,cmap="magma",vmin=int(np.min(np.min(PAM120))), vmax=int(np.max(np.max(PAM120))),interpolation='none')
    return score
```

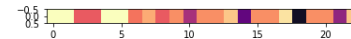
```
example1 = 'CCSSCCTPAGNDEHRVLYWDEQH'
perfect_match = compare_pam120(example1, example1)
```

```
9 9 0 0 0 C C
9 18 0 0 1 C C
3 21 1 1 2 S S
3 24 1 1 3 S S
9 33 0 0 4 C C
9 42 0 0 5 C C
4 46 2 2 6 T T
6 52 3 3 7 P P
3 55 4 4 8 A A
5 60 5 5 9 G G
4 64 6 6 10 N N
5 69 7 7 11 D D
5 74 8 8 12 E E
7 81 10 10 13 H H
6 87 11 11 14 R R
5 92 16 16 15 V V
5 97 15 15 16 L L
8 105 18 18 17 Y Y
12 117 19 19 18 W W
5 122 7 7 19 D D
5 127 8 8 20 E E
6 133 9 9 21 Q Q
7 140 10 10 22 H H
final score: 140
```

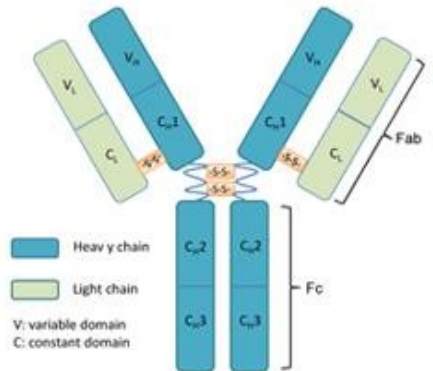


```
example2 = 'CCSSCCTPAGADEHAVLYADEAH'
a_few_substitutions = compare_pam120(example1, example2)
```

```
9 9 0 0 0 C C
9 18 0 0 1 C C
3 21 1 1 2 S S
3 24 1 1 3 S S
9 33 0 0 4 C C
9 42 0 0 5 C C
4 46 2 2 6 T T
6 52 3 3 7 P P
3 55 4 4 8 A A
5 60 5 5 9 G G
0 60 6 4 10 N N
5 65 7 7 11 D D
5 70 8 8 12 E E
7 77 10 10 13 H H
-3 74 11 4 14 R R
5 79 16 16 15 V V
5 84 15 15 16 L L
8 92 18 18 17 Y Y
-7 85 19 4 18 W W
5 90 7 7 19 D D
5 95 8 8 20 E E
-1 94 9 4 21 Q Q
7 101 10 10 22 H H
final score: 101
```



PAM in action



```
#heavy chain, constant region, Fc, Igg1, human
#https://www.ncbi.nlm.nih.gov/protein/AEV43323.1
```

```
# versus
```

```
human_heavychain_fragment = 'PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPEVKFNWYVDGVEVHNAKTKPREEQYNSTYRVVSVLTVLHQDWLNGKEYKCKVSNKALPAPIEKTISI
mouse_heavychain_fragment = 'SSVFIFPPKPKDVLITLTPKVTCCVVVDISKDDPEVQFSWFVDDVEVHTAQTQPREEQFNSTFRSVSELPIMHQDWLNGKEFKCRVNSAAFPAPIEKTISI
```

```
plt.rcParams['figure.figsize'] = 100,1
human_versus_mouse = compare_pam120(human_heavychain_fragment,mouse_heavychain_fragment, fulloutput=False)
```

```
final score: 815
```



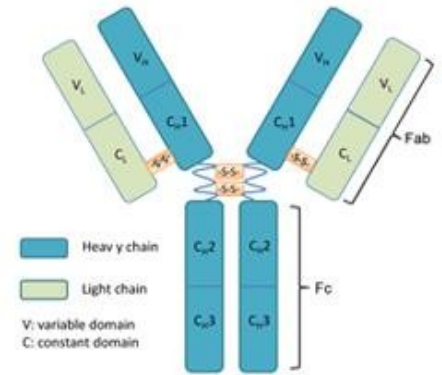
```
human_heavychain_fragment = 'PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPEVKFNWYVDGVEVHNAKTKPREEQYNSTYRVVSVLTVLHQDWLNGKEYKCKVSNKALPAPIEKTISI
macac_heavychain_fragment = 'PSVFLFPPKPKDTLMISRTPEVTCVVVDVSDQEDPDVKFNWYVNGAEVHHAQTQPRETQYNSTYRVVSVLTVTHQDWLNGKEYTCKVSNKALPAPIQKTSI
```

```
human_versus_monkey = compare_pam120(human_heavychain_fragment,macac_heavychain_fragment,fulloutput=False)
```

```
final score: 1047
```



BLOSUM62 in action



```
human_heavychain_fragment = 'PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPEVKFNWYVDGVEVHNAKTKPREEQYNSTYRVVSVLTVLHQDWLNGKEYKCKVSNKALPAPIEKTISI
macac_heavychain_fragment = 'PSVFLFPPKPKDTLMISRTPEVTCVVVDVSQEDPDVKFNWYVNGAEVHHAQTKPRETQYNSTYRVVSVLTVTHQDWLNGKEYTCKVSNKALPAPIQKTSI
human_versus_monkey = compare_BLOSUM62(human_heavychain_fragment,macac_heavychain_fragment,fulloutput=False)
```

final score: 1045



Choosing a Matrix

PAM models – evolutionary distance

BLOSUM – sequence similarity

Rules of thumb

- Choice depends on situation
- Distantly related → PAM-250, BLOSUM-50
- Closely related → PAM-120, BLOSUM-80
- Short sequences → PAM-40, BLOSUM-80
- BLOSUM62 standard for ungapped matching
- BLOSUM50 better for gapped

Scoring Matrices

Scoring Schemes for nucleotides

	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

Scores derived from alignments

(A)					(B)					(C)				
	A	C	G	T		A	C	G	T		A	C	G	T
A	67	-96	-20	-117	A	91	-114	-31	-123	A	100	-123	-28	-109
C	-96	100	-79	-20	C	-114	100	-125	-31	C	-123	91	-140	-28
G	-20	-79	100	-96	G	-31	-125	100	-114	G	-28	-140	91	-123
T	-117	-20	-96	67	T	-123	-31	-114	91	T	-109	-28	-123	100

► based on G+C content (37%, 47%, 53%)

Gaps

- ♦ Insertions and deletions
 - ♦ Homologous sequences have different lengths due to insertions and deletions
 - ♦ Treat by inserting gaps in the alignment
- ♦ Gap penalties
 - ♦ Gaps can be overused to create false alignments
 - ♦ - L - - - I N E
 - ♦ A L A N I N E
 - ♦ Gap penalty – gap opening
 - ♦ Introduction of a gap costs in score
 - ♦ Limits number of inserted gaps
 - ♦ Gap extension penalty
 - ♦ Smaller than opening penalty
 - ♦ Once a gap, easier to extend
 - ♦ Magnitude depends on setting
 - ♦ High for closely related sequences
 - ♦ Low for distantly related sequences

Gaps

Alignment B has higher similarity than A but is not necessarily correct

(A)

```

Bovine PI-3Kinase p110a      LNWENPDIMSELLFQNNEIFKNGDRLRQDMLTLQIIRIMENIWQNGQLDLRMLPYGCLSIGDCVGLIEVVRNSHTIMQIQCKGGLK GAL
cAMP-dependent protein kinase --WENPAQNTAHLDDQFERIKTLGTGSFGRVMLVKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDNSNLY

Bovine PI-3Kinase p110a      QFNSHTLHQWLKDNKNGEIIDAAIDLFTSRSCAGYCVATFILGIGDRHNSNIMVKDDGQLFHIDFGHFLDHKKKKFGYKRERVPFVLTQDF
cAMP-dependent protein kinase MVMEYVPGGEMFSLRRIIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGYIQVTDGFGFAKRVKGRGTWXL CGTPEYLAP

Bovine PI-3Kinase p110a      L I V I S K G A Q E C T K T R E F E R F Q E M C Y K A Y L A I R Q H A N L F I N L F S M M L G S G M P E L Q S F D D I A Y I R K T L A L D K T E Q E A L E Y F M K Q M N D A H H G G
cAMP-dependent protein kinase E I I L S K G Y N K A V D W W A L G V L I Y E M A A G Y P P F F A D Q P I Q I Y E K I V S G K V R F P S H F S S D L K D L L R N L L Q V D L T K R F G N L K N G V N D I K N H K W F

Bovine PI-3Kinase p110a      W T T K M D W I F H T I K Q H A L N -----
cAMP-dependent protein kinase A T T D W I A I Y Q R K V E A P F I P K F K G P G D T S N F D D Y E E E E I R V X I N E K C G K E F S E F

```

(B)

```

Bovine PI-3Kinase p110a      LNWENPDIMSELLFQNNEIFKNGDRLRQDMLTLQIIRIMENIWQNGQLDLRMLPYGCLSIGDCVGLIEVVRNSHTIMQIQCKGGLK GAL
cAMP-dependent protein kinase ?-WENPAQNTAHLDDQFERIKTLGTGSFGRVMLVKHM--ETGNHYAMKILDKQKV-VLKLQIEHTLNEKRILQAVNFPFLVKLEFSFKDN-

Bovine PI-3Kinase p110a      QFNSHTLHQWLKDNKNGEIIDAAIDLFTSRSCAGYCVATFILGIGDRHNSNIMVKD-DGQLFHIDFGHFLDHKKKKFGYKRERVPFVL--T
cAMP-dependent protein kinase -SNLYMVMEYVPGGEMFSLRRIIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGYIQVTDGFGFAKRVKGRGTWXL CGT

Bovine PI-3Kinase p110a      Q D F L --- I V I S K G A Q E C T K T R E F E R F - Q E M C -- Y K A Y L A I R Q H A N L F I N L F S M M L G S G M P E L Q S F D D I A Y I R K T L A L D K T E Q E A L E Y F M K
cAMP-dependent protein kinase P E Y L A P E I I L S K G Y N K A V D W W A L G V L I Y E M A A G Y P P F F A - D Q P I Q I Y E K I V S G K V R F -- P S H F S S D L K D L L R N L L Q V D L T K R -- F G N L K N

Bovine PI-3Kinase p110a      Q M N D A H H G G W T T K M D W I ----- F H T I K Q H A L --- N -----
cAMP-dependent protein kinase G V N D I K N H K W F A T T D W I A I Y Q R K V E A P F I P K F K G P G D T S N F D D Y E E E E I R V X I N E K C G K E F S E F

```

Heuristic scores

Simplest

Assign gap penalty $-d$
when aligning to gap

	b	a	b
	b	-	b
Gap penalty			-d

Definition (Linear gap penalty)

$$\gamma(g) = -gd \quad (1)$$

where g = length of
gap

	b	a	a	a	b
	b	-	-	-	b
Gap penalty			-d	-d	-d

Heuristic scores

Definition (Affine gap penalty)

$$\gamma(g) = -d - (g - 1)e \quad (2)$$

- ▶ Models observed preference for fewer and longer gaps
- ▶ Costs more to open than to extend ($d > e$)

	b	a	a	a	b
	b	-	-	-	b
Gap penalty		-d	-e	-e	

OUTLINE – **analyzing matches** & **aligning sequences**

1. **Dotplots** – comparing sequence similarity visually
 - Window size
 - Tolerance
2. **Score Matrices** – quantifying sequence similarity
 - PAM
 - BLOSUM
 - Pairwise distance, hierarchy, and information theory
3. **Alignment types**
4. **Dynamic Programming** – break problem into small ones
 - Global optimal alignment - Needleman-Wunsch
 - Local optimal alignment - Smith-Waterman

Alignment

Affine types

Global alignment

Alignment over full sequence

Local alignment

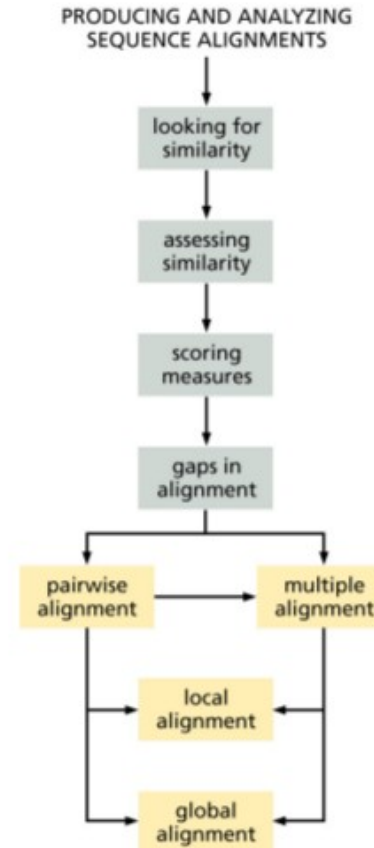
Alignment over partial sequence

Pairwise sequence alignment

Align two sequences

Multiple sequence alignment

Align multiple sequences



Flow diagram 4.2 in Understanding Bioinformatics (Zvelebil/Baum, 2008)

Alignment

Local alignment

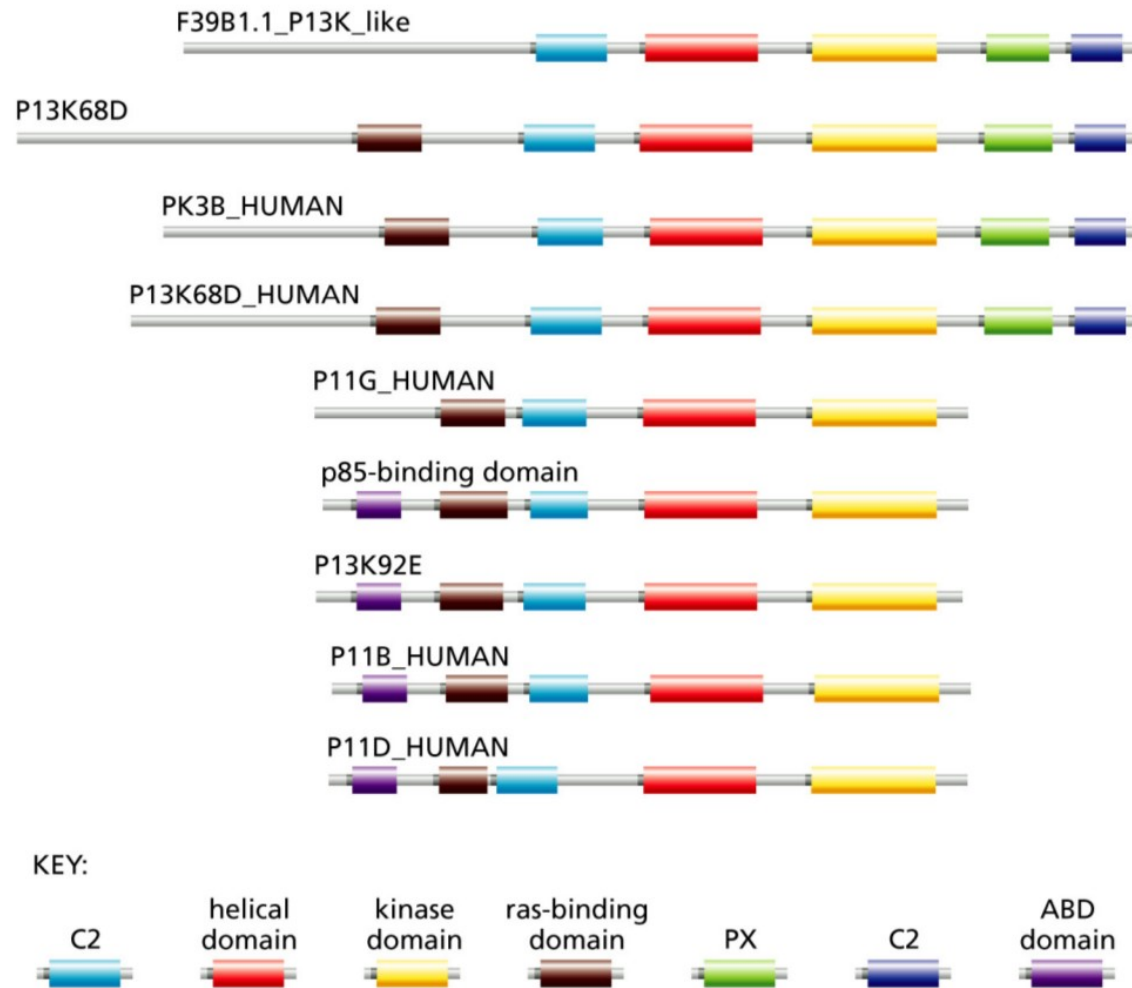


Figure 4.6 in Understanding Bioinformatics (Zvelebil/Baum, 2008)

Alignment

Local & Global alignment

(A) local

PI3-kinase DRHNSNIMVKDDGQLFHI DFG
 cAMP PK DLKPENLLIDQQGYIQVT DFG

(B) global

```

PI3-kinase  HQLGNLR--LEECRI---MSSAKRPLWLNWENPDIMSELLFQNNIIFKNGDDLRRQDMLT
cAMP PK     GNAAAARKKXGEQESVKEFLAKAKEDFLKKWENPAQNTAHL DQFERIKTLGTGSFGRVML-
              10      20      30      40      50

PI3-kinase  LQIIRIME--NIWQNGGLDLRMLPYGCLSIGDCVGLIEVVRNSHTIMQ-IQCKGGLK GAL
cAMP PK     ---VKHMETGNHYAMKILDKQKVVK-----LQIEHTLNEKRILQAVNFPFLVKLEF
              60      70      80      90     100     110

PI3-kinase  QFNSHT-LHQWLKDKNKGEIYDAA--IDL FTRSCAGYCVATFILGIGDRHNSNIMVKD-D
cAMP PK     SFKDNSNLYMMEYVPGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLK
              110     120     130     140     150     160

PI3-kinase  GQLFHI DFGHFLDHKKKKFGYKRERVP-----FVLTQDFL---IVISKGAQECTKTREFE
cAMP PK     PENLLIDQQGYI--QVTDFGFAK-RVKGRTWXL CGTPEYLAPEIILSKGYNKAVDWWALG
              170     180     190     200     210     220

PI3-kinase  RF-QEMC--YKAYLAIRQHANLFINLFSMMLGSGMPELQSFDDIAYIRKTLALDKTEQEA
cAMP PK     VLIYEMAAGYPPFFA-DQPIQIYEKIVSGKVR--FPSHFSSDLKDLLRNLLQVDLTKR--
              230     240     250     260     270     280

PI3-kinase  LEYEMKQMNDAHHGGWTTKMDWI-----FHTIKQHALN----
cAMP PK     FGNLKNQVNDIKNHKWFATTDWIAIYQRKVEAPFIPKFKGPGDTSNFDDEEEEEIRVXIN
              280     290     300     310     320     330     340
  
```

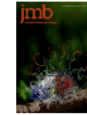
Figure 4.7 in Understanding Bioinformatics (Zvelebil/Baum, 2008)

Needleman Wunsch



Journal of Molecular Biology

Volume 48, Issue 3, 28 March 1970, Pages 443-453



A general method applicable to the search for similarities in the amino acid sequence of two proteins ☆

Saul B. Needleman, Christian D. Wunsch

[Show more](#)

[https://doi.org/10.1016/0022-2836\(70\)90057-4](https://doi.org/10.1016/0022-2836(70)90057-4)

[Get rights and content](#)

Abstract

A computer adaptable method for finding similarities in the amino acid sequences of two proteins has been developed. From these findings it is possible to determine whether significant homology exists between the proteins. This information is used to trace their possible evolutionary development.

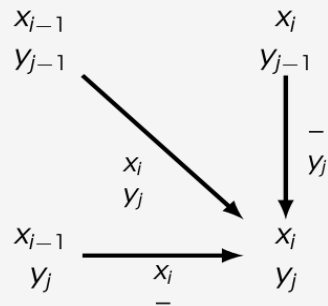
The maximum match is a number dependent upon the similarity of the sequences. One of its definitions is the largest number of amino acids of one protein that can be matched with those of a second protein allowing for all possible interruptions in either of the sequences. While the interruptions give rise to a very large number of comparisons, the method efficiently excludes from consideration those comparisons that cannot contribute to the maximum match.

Comparisons are made from the smallest unit of significance, a pair of amino acids, one from each protein. All possible pairs are represented by a two-dimensional array, and all possible comparisons are represented by pathways through the array. For this maximum match only certain of the possible pathways must be evaluated. A numerical value, one in this case, is assigned to every cell in the array representing like amino acids. The maximum match is the largest number that would result from summing the cell values of every pathway.

Needleman Wunsch

Observation: three ways to extend alignment

- ▶ Match (diagonal)
- ▶ Gap in top protein (vertical)
- ▶ Gap in left protein (horizontal)



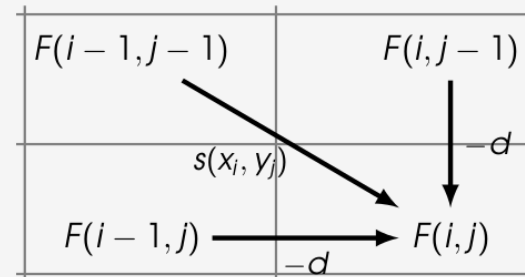
Score matrix F

- ▶ $F(i, j)$ = score up to x_i, y_j
- ▶ Build recursively

Iteration

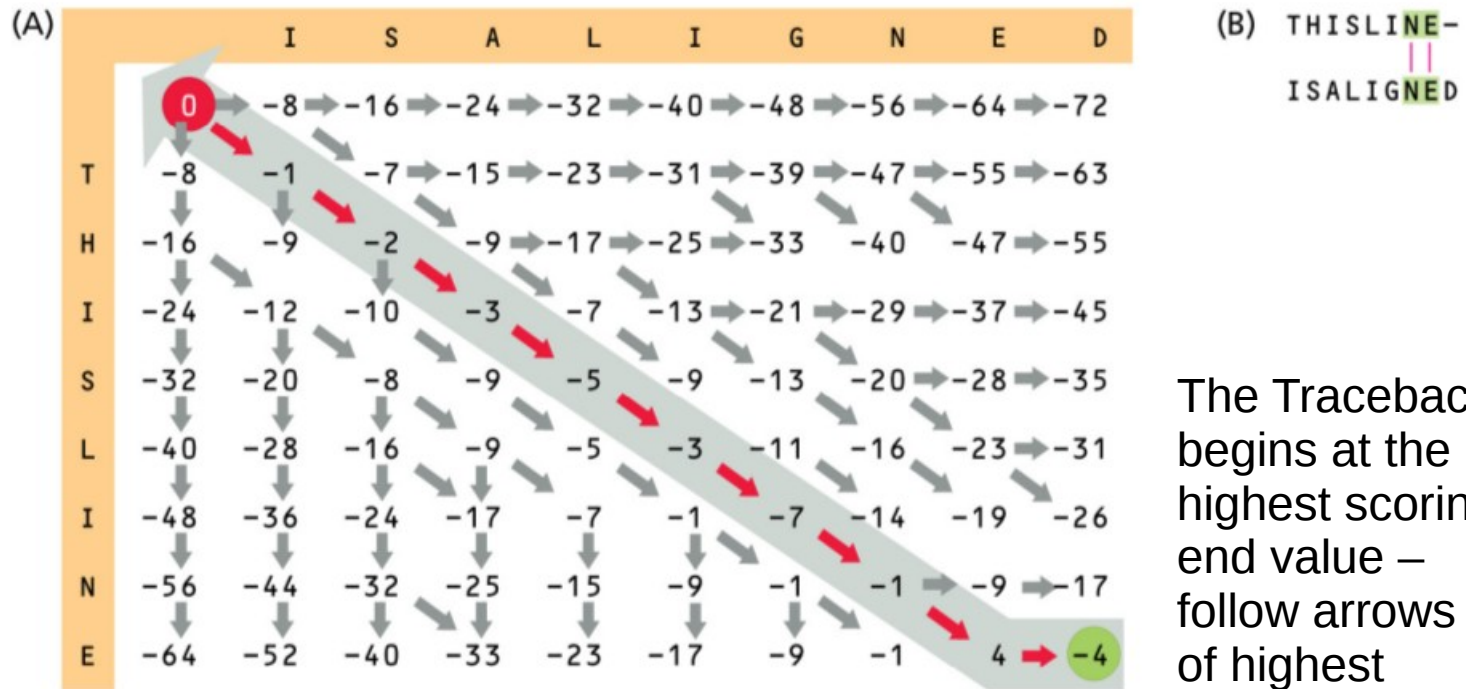
$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases} \quad (3)$$

Illustration



Dynamic Programming

Needleman-Wunsch Global alignment



The Traceback begins at the highest scoring end value – follow arrows of highest scores back to 0

Figure 5.9A in Understanding Bioinformatics (Zvelebil/Baum, 2008)

Dynamic Programming

Changing the gap penalty value – must be compatible with substitution matrix

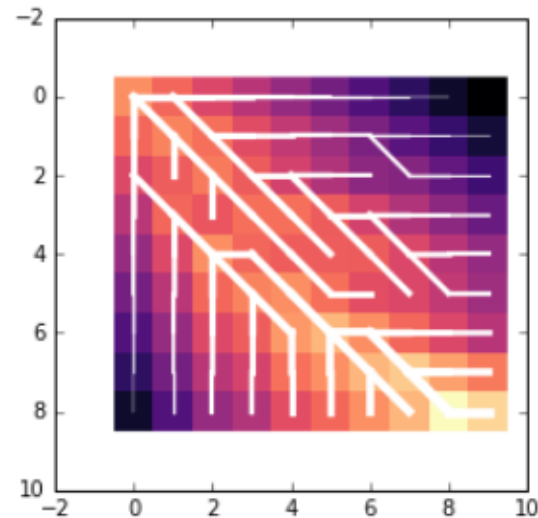
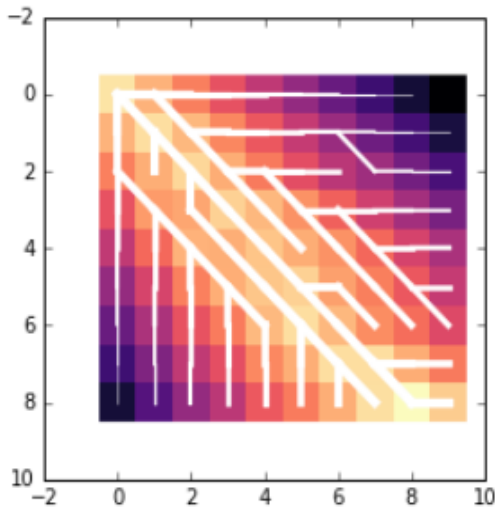
THISLINE-
||
ISALIGNED

Gap penalty = -8

THIS-LI-NE-
|| || ||
--ISALIGNED

Gap penalty = -4

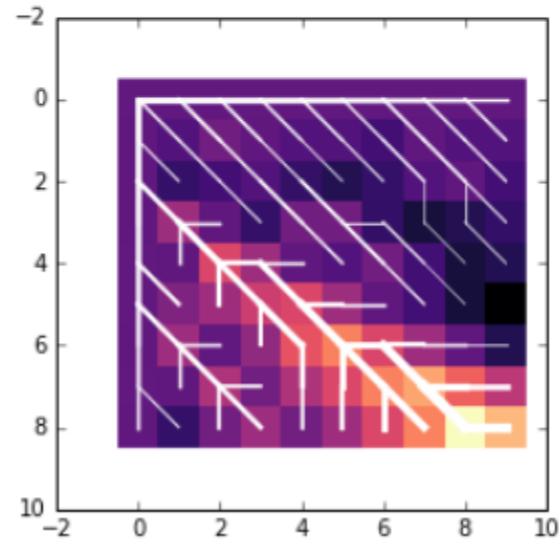
[[0. -8. -16. -24. -32. -40. -48. -56. -64. -72.]	[[0. -4. -8. -12. -16. -20. -24. -28. -32. -36.]
[-8. -1. -7. -15. -23. -31. -39. -47. -55. -63.]	[-4. -1. -3. -7. -11. -15. -19. -23. -27. -31.]
[-16. -9. -2. -9. -17. -25. -33. -38. -46. -54.]	[-8. -5. -2. -5. -9. -13. -17. -18. -22. -26.]
[-24. -12. -10. -3. -7. -13. -21. -29. -37. -45.]	[-12. -4. -6. -3. -3. -5. -9. -13. -17. -21.]
[-32. -20. -8. -9. -5. -9. -13. -20. -28. -36.]	[-16. -8. 0. -4. -5. -5. -5. -8. -12. -16.]
[-40. -28. -16. -9. -5. -3. -11. -16. -23. -31.]	[-20. -12. -4. -1. 0. -3. -7. -8. -11. -15.]
[-48. -36. -24. -17. -7. -1. -7. -14. -19. -26.]	[-24. -16. -8. -5. 1. 4. 0. -4. -8. -12.]
[-56. -44. -32. -25. -15. -9. -1. -1. -9. -17.]	[-28. -20. -12. -9. -3. 0. 4. 6. 2. -2.]
[-64. -52. -40. -33. -23. -17. -9. -1. 4. -4.]	[-32. -24. -16. -13. -7. -4. 0. 4. 11. 7.]



Dynamic Programming

Needleman-
Wunsch
No penalty for end
gaps

[0.	0.	0.	0.	0.	0.	0.	0.	0.	0.]
[0.	-1.	1.	0.	-1.	-1.	-2.	0.	-1.	-1.]
[0.	-3.	-2.	-1.	-3.	-4.	-3.	-1.	0.	-2.]
[0.	4.	0.	-3.	1.	1.	-3.	-5.	-4.	-3.]
[0.	0.	8.	4.	0.	-1.	1.	-2.	-5.	-4.]
[0.	2.	4.	7.	8.	4.	0.	-2.	-5.	-8.]
[0.	4.	0.	3.	9.	12.	8.	4.	0.	-4.]
[0.	0.	5.	1.	5.	8.	12.	14.	10.	6.]
[0.	-3.	1.	4.	1.	4.	8.	12.	19.	15.]]



Needleman-Wunsch

- Construct a matrix with scores
 - Begin by filling in end gap scores according to either
 - Fixed gap penalty $n*d$
 - Forgiving end gaps $\rightarrow 0$
 - Compute hypothetical alignment scores from boxes with 3 corner-forming neighbors that already have values beginning from the far corner
 - Accept highest scores and directions so that each box has a winner
 - Iterate until matrix is full
 - Perform a traceback
 - Start from highest scoring element – somewhere on the end axes
 - Follow arrows and highest scores backward to zero
 - Fill out the alignment transcript as you go

Smith-Waterman Local Alignment

Motivation

- ▶ Find similar domains and subsequences

Smith-Waterman

- ▶ First local alignment algorithm (1981)
- ▶ Requirements on scoring scheme:
 1. $E(\text{random}) < 0$
 2. $E(\text{non-random}) > 0$
- ▶ Most scoring matrices fulfil requirements
- ▶ Key differences to global alignment algorithm:
 1. If $F(i, j) < 0 \Rightarrow F(i, j) \rightarrow 0$ and alignment is rejected
 2. Start traceback from highest element

Smith-Waterman Local Alignment

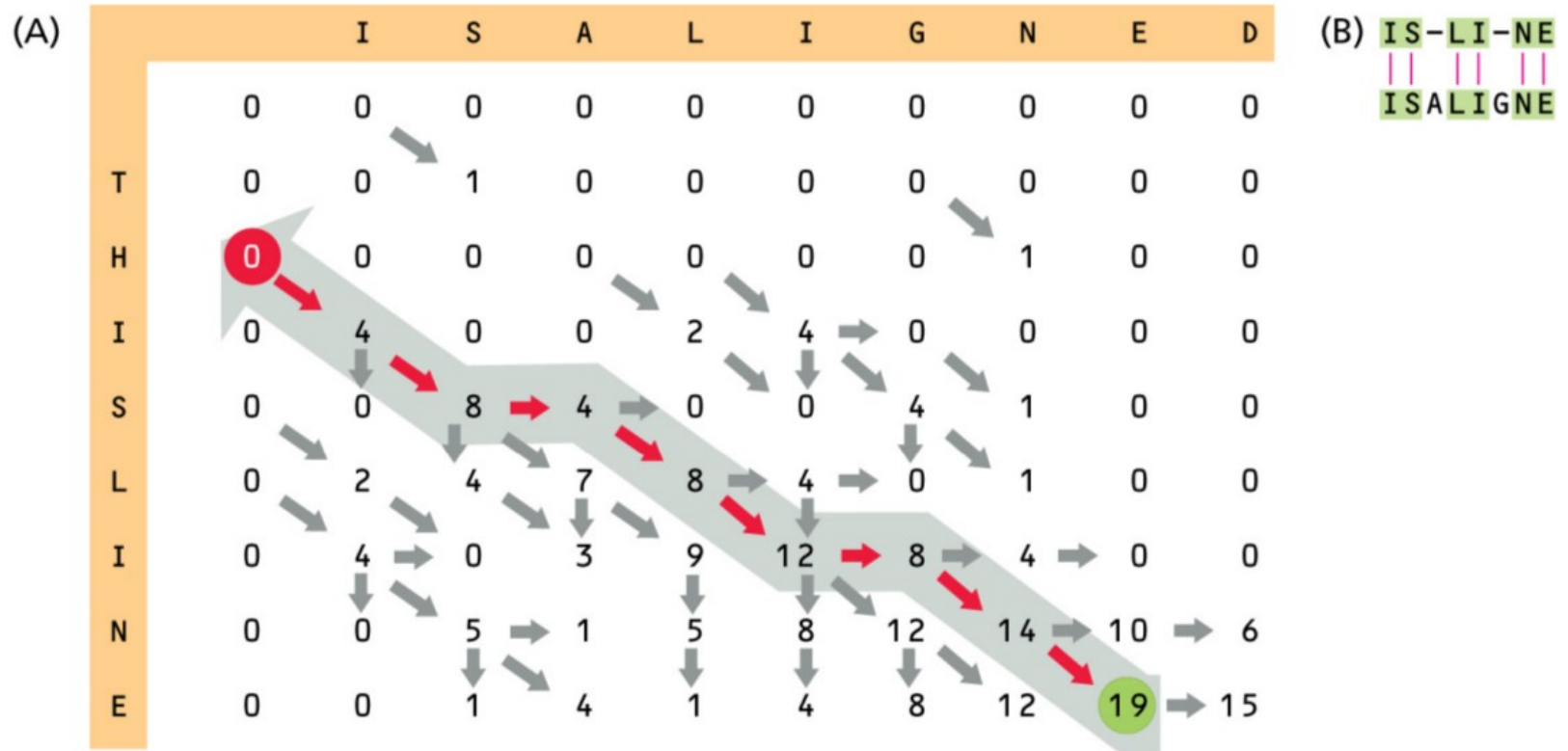


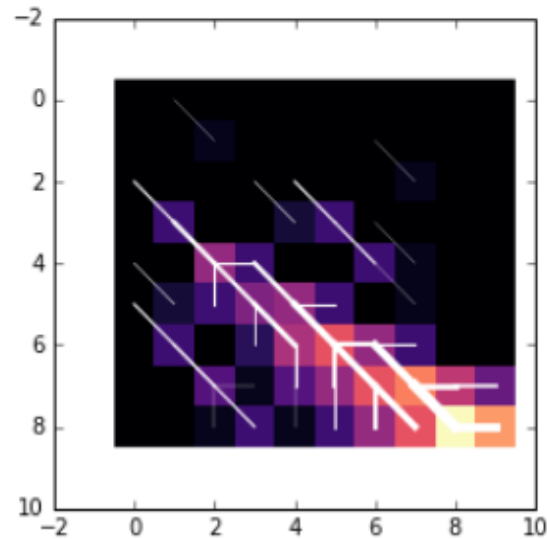
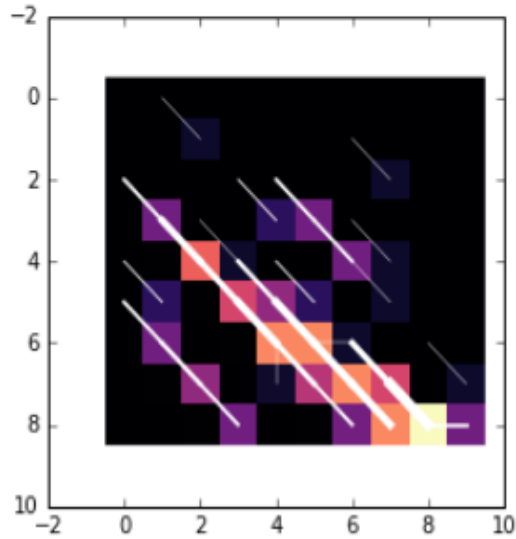
Figure 5.16 in Understanding Bioinformatics (Zvelebil/Baum, 2008)

Dynamic Programming

Gap penalty -8

Gap penalty -4

[0.	0.	0.	0.	0.	0.	0.	0.	0.	0.]	[0.	0.	0.	0.	0.	0.	0.	0.	0.]
[0.	0.	1.	0.	0.	0.	0.	0.	0.]	[0.	0.	1.	0.	0.	0.	0.	0.	0.]	
[0.	0.	0.	0.	0.	0.	0.	1.	0.]	[0.	0.	0.	0.	0.	0.	1.	0.	0.]	
[0.	4.	0.	0.	2.	4.	0.	0.	0.]	[0.	4.	0.	0.	2.	4.	0.	0.	0.]	
[0.	0.	8.	1.	0.	0.	4.	1.	0.]	[0.	0.	8.	4.	0.	0.	4.	1.	0.]	
[0.	2.	0.	7.	5.	2.	0.	1.	0.]	[0.	2.	4.	7.	8.	4.	0.	1.	0.]	
[0.	4.	0.	0.	9.	9.	1.	0.	0.]	[0.	4.	0.	3.	9.	12.	8.	4.	0.]	
[0.	0.	5.	0.	1.	6.	9.	7.	0.]	[0.	0.	5.	1.	5.	8.	12.	14.	10.]	
[0.	0.	0.	4.	0.	0.	4.	9.	12.]	[0.	0.	1.	4.	1.	4.	8.	12.	19.]	



Summary

- Comparisons
 - Dot plots are a visual way to compare a pair
 - Score matrices are built from empirical data about substitution frequencies
 - Gaps can be incorporated into score matrices heuristically
 - PAM is based on closely related evolutionary changes
 - PAM 1, 100, 250 refer to evolutionary distances, powers of PAM
 - BLOSUM is based on conserved domains in distantly related proteins
 - BLOSUM 50, 62 etc are % threshold identities to cluster sequences
- Pairwise alignments
 - Local alignments give optimal alignment even for multidomain proteins with unrelated segments
 - Smith waterman
 - Global alignments give optimal alignment for similar sized proteins with mostly corresponding positions
 - Needleman-Wunsch