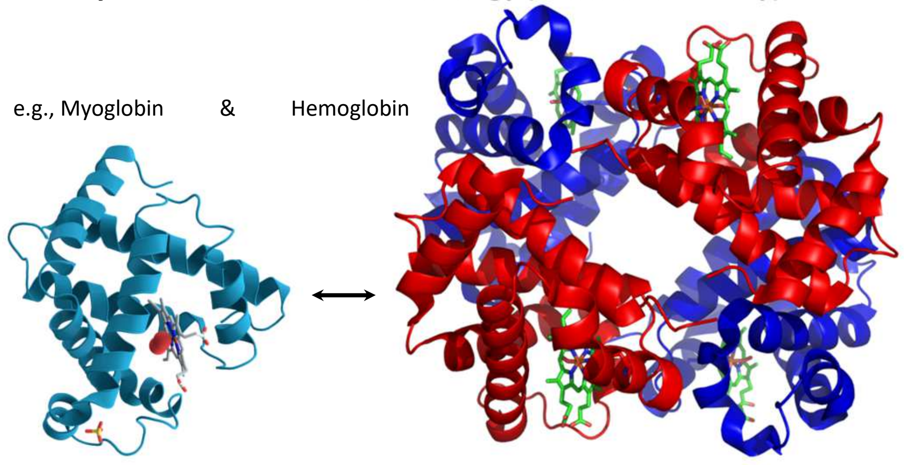
Typically, to be "biologically related" means to <u>share a common ancestor</u>. In biology, we call this *homologous*.
Two proteins sharing a common ancestor are said to be *homologs*.
Homology often implies structural similarity & sometimes (not always) sequence similarity. **A statistically significant sequence or structural similarity can be used to infer homology (common ancestry).**

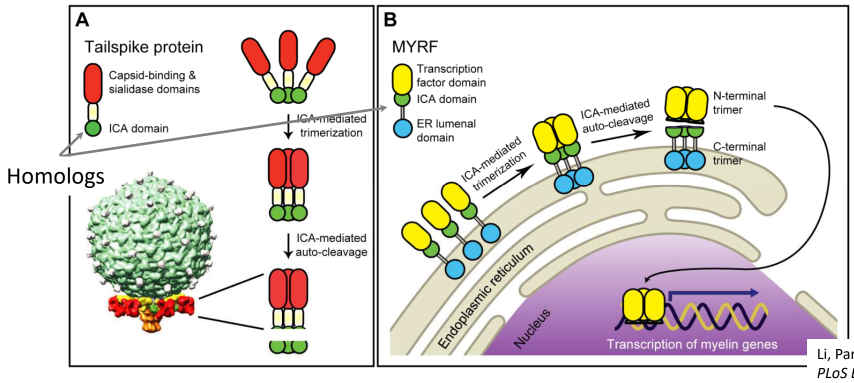
e.g., Myoglobin & Hemoglobin



http://en.wikipedia.org/wiki/File:Myoglobin.png & File:1GZX_Haemoglobin.png

In practice, searching for sequence or structural similarity is one of the most powerful computational approaches to discover a gene's function. We can often gain insight about a protein from its homologs.

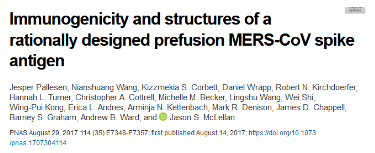
For example, my lab discovered that myelinating the neurons in your brain reuses the same biochemical mechanism that phage use to make capsids. The key breakthrough was recognizing that the human and phage proteins contained homologous domains.



A
Tailspike protein
Capsid-binding & sialidase domains
ICA domain
ICA-mediated trimerization
ICA-mediated auto-cleavage
Homologs
B
MYRF
Transcription factor domain
ICA domain
ER lumenal domain
ICA-mediated trimerization
ICA-mediated auto-cleavage
N-terminal trimer
C-terminal trimer
Endoplasmic reticulum
Nucleus
Transcription of myelin genes

Li, Park, Marcotte
PLoS Biology (2013)

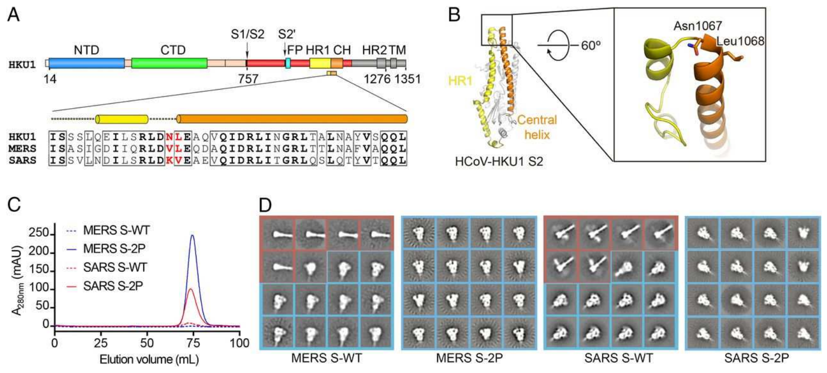
& for a very timely example, here’s the “trillion dollar” paper from the McLellan lab that the SARS-CoV-2 vaccines are designed from based on homology to MERS and SARS spike antigens



Immunogenicity and structures of a rationally designed prefusion MERS-CoV spike antigen

Jesper Pallesen, Nianshuang Wang, Kizzmekia S. Corbett, Daniel Wrapp, Robert N. Kirchdoerfer, Hannah L. Turner, Christopher A. Cottrell, Michelle M. Becker, Lingshu Wang, Wei Shi, Wing-Pui Kong, Erica L. Andres, Arminja N. Kettenbach, Mark R. Denison, James D. Chappell, Barney S. Graham, Andrew B. Ward, and Jason S. McLellan

PNAS August 29, 2017 114 (35) E7348-E7357; first published August 14, 2017; https://doi.org/10.1073/pnas.1707304114



Sequence alignment algorithms such as BLAST, PSI-BLAST, FASTA, and the Needleman–Wunsch & Smith-Waterman algorithms arguably comprise some of the most important driver technologies of modern biology and underlie the sequencing revolution.

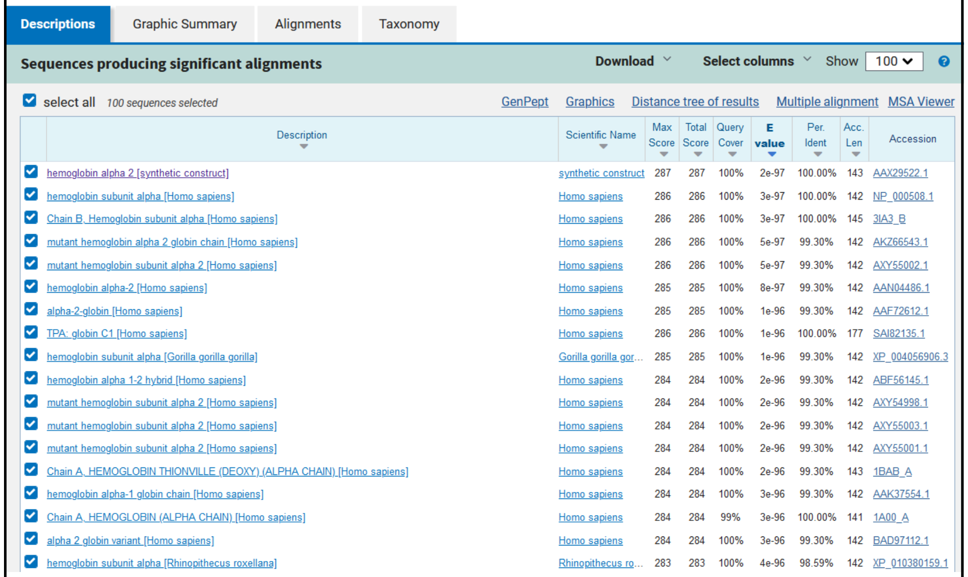
So, let’s start learning bioinformatics algorithms by learning how to align two protein sequences.

Live demo:

http://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&BLAST_PROGRAMS=blastp&PAGE_TYPE=BlastSearch&SHOW_DEFAULTS=on&LINK_LOC=blasthome

MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP
AVHASLDKFLASVSTVLTSKYR

The next few slides show the data from searching this dbase (#'s may be a bit different from the live version):
Title:All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF excluding environmental samples from WGS projects
Molecule Type:Protein
Update date:2024/01/11
Number of sequences:648450839



Descriptions | Graphic Summary | Alignments | Taxonomy

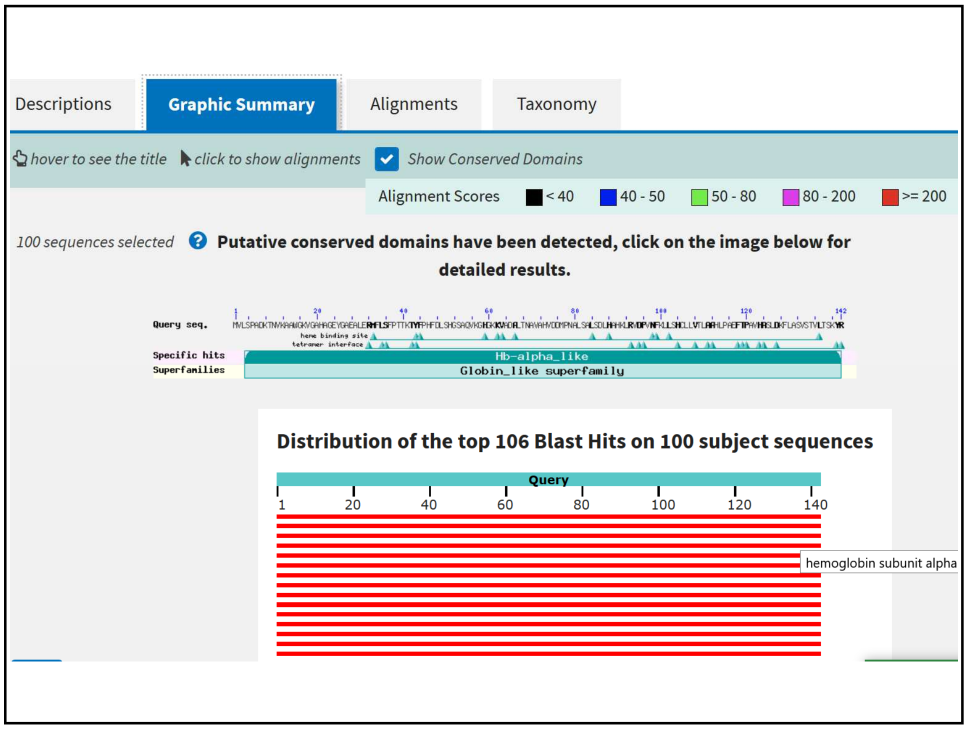
Sequences producing significant alignments

Download | Select columns | Show 100

select all *100 sequences selected*

GenPept | Graphics | Distance tree of results | Multiple alignment | MSA Viewer

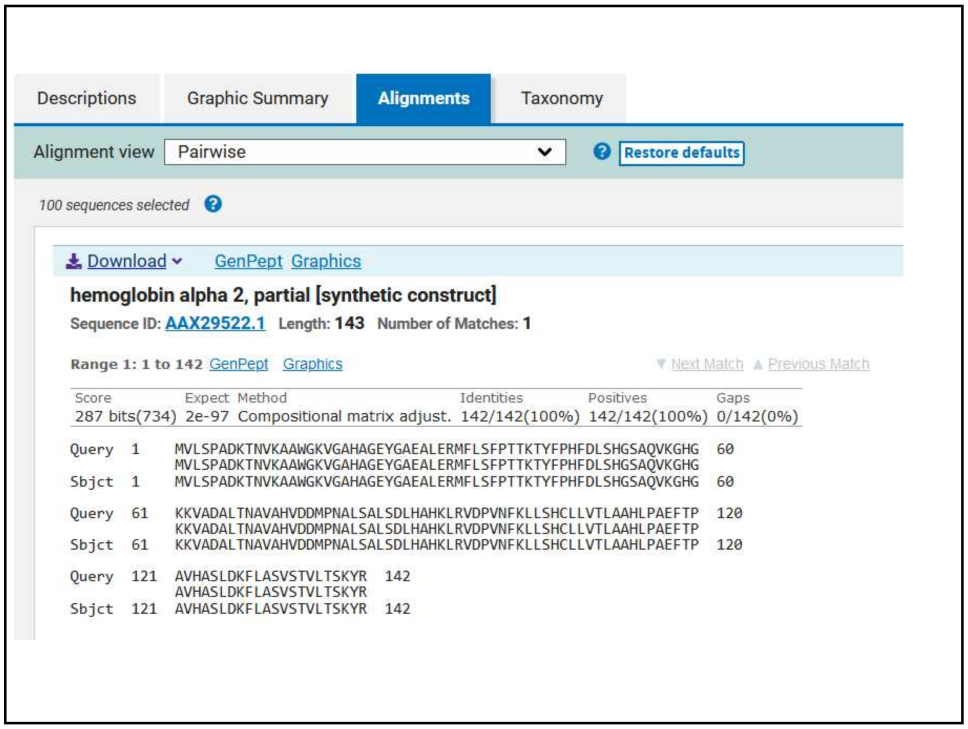
	Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
✓	hemoglobin alpha 2 [synthetic construct]	synthetic construct	287	287	100%	2e-97	100.00%	143	AAX29522.1
✓	hemoglobin subunit alpha [Homo sapiens]	Homo sapiens	286	286	100%	3e-97	100.00%	142	NP_000508.1
✓	Chain B, Hemoglobin subunit alpha [Homo sapiens]	Homo sapiens	286	286	100%	3e-97	100.00%	145	3IA3_B
✓	mutant hemoglobin alpha 2 globin chain [Homo sapiens]	Homo sapiens	286	286	100%	5e-97	99.30%	142	AKZ66543.1
✓	mutant hemoglobin subunit alpha 2 [Homo sapiens]	Homo sapiens	286	286	100%	5e-97	99.30%	142	AXY55002.1
✓	hemoglobin alpha-2 [Homo sapiens]	Homo sapiens	285	285	100%	8e-97	99.30%	142	AAN04486.1
✓	alpha-2-globin [Homo sapiens]	Homo sapiens	285	285	100%	1e-96	99.30%	142	AAF72612.1
✓	TPA: globin C1 [Homo sapiens]	Homo sapiens	286	286	100%	1e-96	100.00%	177	SAI82135.1
✓	hemoglobin subunit alpha [Gorilla gorilla gorilla]	Gorilla gorilla gor...	285	285	100%	1e-96	99.30%	142	XP_004056906.3
✓	hemoglobin alpha 1-2 hybrid [Homo sapiens]	Homo sapiens	284	284	100%	2e-96	99.30%	142	ABF56145.1
✓	mutant hemoglobin subunit alpha 2 [Homo sapiens]	Homo sapiens	284	284	100%	2e-96	99.30%	142	AXY54998.1
✓	mutant hemoglobin subunit alpha 2 [Homo sapiens]	Homo sapiens	284	284	100%	2e-96	99.30%	142	AXY55003.1
✓	mutant hemoglobin subunit alpha 2 [Homo sapiens]	Homo sapiens	284	284	100%	2e-96	99.30%	142	AXY55001.1
✓	Chain A, HEMOGLOBIN THIONVILLE (DEOXY) (ALPHA CHAIN) [Homo sapiens]	Homo sapiens	284	284	100%	2e-96	99.30%	143	1BAB_A
✓	hemoglobin alpha-1 globin chain [Homo sapiens]	Homo sapiens	284	284	100%	3e-96	99.30%	142	AAK37554.1
✓	Chain A, HEMOGLOBIN (ALPHA CHAIN) [Homo sapiens]	Homo sapiens	284	284	99%	3e-96	100.00%	141	1A00_A
✓	alpha 2 globin variant [Homo sapiens]	Homo sapiens	284	284	100%	3e-96	99.30%	142	BAD97112.1
✓	hemoglobin subunit alpha [Rhinopithecus roxellana]	Rhinopithecus ro...	283	283	100%	4e-96	98.59%	142	XP_010380159.1



100 sequences selected

Putative conserved domains have been detected, click on the image below for detailed results.

Distribution of the top 106 Blast Hits on 100 subject sequences



100 sequences selected

hemoglobin alpha 2, partial [synthetic construct]

Sequence ID: AAX29522.1 **Length:** 143 **Number of Matches:** 1

Range 1: 1 to 142 GenPept Graphics

```
Score           Expect  Method                         Identities     Positives      Gaps
287 bits(734)   2e-97   Compositional matrix adjust.   142/142(100%)  142/142(100%)  0/142(0%)

Query  1    MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG  60
            MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
Sbjct  1    MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG  60

Query  61   KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP  120
            KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP
Sbjct  61   KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP  120

Query  121  AVHASLDKFLASVSTVLTSKYR  142
            AVHASLDKFLASVSTVLTSKYR
Sbjct  121  AVHASLDKFLASVSTVLTSKYR  142
```

Protein sequence alignment

Two biologically related proteins with similar sequences:

```
FlgA1 EAGNVKLKRGRLDTLPPRTVLDINQLVDAISLRDLSPDQPIQLTQFRQAWRVKAGQRVNVIASGD
         ++K+K+GRLDTLPP  +L+ N    A+SLR ++  QP+     R+ W +KAGQ V V+A G+
FlgA2 TLQDIKMKQGRLDTLPPGALLEPNFAQGAVSLRQINAGQPLTRNMLRRLWIIKAGQDVQVLALGE
```

Also biologically related (& fold up into the same 3D protein structure):

```
FlgA1 EAGNVKLKRGRLDTLPPRTVLDINQLVDAISLRDLSPDQPIQLTQFRQAWRVKAGQRVNVIASGD
       A   +         P        +L   I+ R L P + I     R+AW V+ G  V V
FlgA3 LAALKQVTLIAGKHKPDAMATHAEELQGKIAKRTLLPGRYIPTAAIREAWLVEQGAAVQVFFIAG
```

But these are biologically unrelated (& fold up into unrelated structures):

```
FlgA1 AGNVKLKRGRLDTLPPRTVLDINQLVDAISLRDLSPDQPIQLTQFRQA-WRVKAGQRVNVIASGD
      AG+V  K G +  + PRT  ++      I+     P  PI    +++A WRV A + V V+  GD
HvcPP AGHV--KNGTMRIVGPRTCSNVWNGTFPINATTTGPSIPIPAPNYKKALWRVSATEYVEVVRVGD
```

(FYI, we'll draw examples from Durbin *et al*., *Biological Sequence Analysis*, Ch. 1 & 2).

To align two sequences, we need to perform 3 steps:

1. **We need some way to decide which alignments are better than others.**
   For this, we'll invent a way to give the alignments a "score" indicating their quality.

2. **Align the two proteins so that they get the best possible score.**

3. **Decide if the score is "good enough" for us to believe the alignment is biologically significant.**

To align two sequences, we need to perform 3 steps:

1. We need some way to decide which alignments are better than others.
   For this, **we'll invent a way to give the alignments a "score" indicating their quality.**

2. Align the two proteins so that they get the best possible score.

3. Decide if the score is "good enough" for us to believe the alignment is biologically significant.

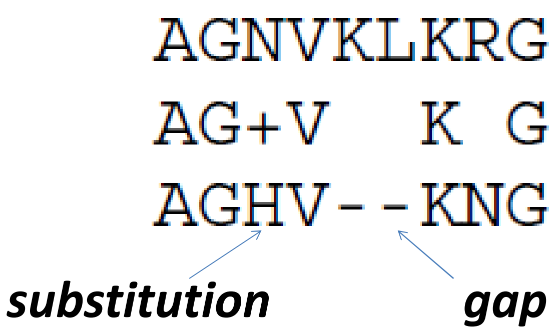
We'll treat mutations as independent events.

This allows us to create an ***additive scoring scheme.***
The score for a sequence alignment will be the sum of the scores for aligning each of the individual positions in two sequences.

What kind of mutations should we expect?

Substitutions, ***insertions*** and ***deletions***.

Insertions and deletions can be treated as equivalent events by considering one or the other sequence as the reference, and are usually called ***gaps***.



Let's consider two models:

First, a **random** model, where amino acids in the sequences occur independently at some given frequencies.

The probability of observing an alignment between *x* and *y* is just the product of the frequencies (q) with which we find each amino acid.

We can write this as:

What does the capital pi mean?

$$P(x, y \mid R) = \prod_i q_{x_i} \prod_j q_{y_j}$$

What's this mean?

What's this mean?

i is just a counter indicating the sequence position

Second, a **match** model, where amino acids at a given position in the alignment arise from some common ancestor with a probability given by the joint probability p_{ab}.

So, under this model, the probability of the alignment is the product of the probabilities of seeing the individual amino acids aligned.

We can write that as:

What does the capital pi mean again?

$$P(x,y \mid M) = \prod_i p_{x_i y_i}$$

What's this mean?

What's this mean?

To decide which model better describes an alignment, we'll take the ratio:

$$\frac{P(x,y \mid M)}{P(x,y \mid R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_j q_{y_j}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

What did these mean again?

Such a ratio of probabilities under 2 different models is called an ***odds ratio***.

Where else have you heard odds ratios used?

Basically: if the ratio > 1, model M is more probable
if < 1, model R is more probable.

Now, to convert this to an additive score *S*, we can simply take the logarithm of the odds ratio (called the ***log odds ratio***):

$$S = \sum_i s(x_i, y_i)$$

This is just the score for aligning one amino acid with another amino acid:

$$s(a,b) = \log\left(\frac{p_{ab}}{p_a p_b}\right)$$

Here written *a* and *b* rather than x_i and y_i to emphasize that this score reflects the *inherent preference* of the two amino acids (*a* and *b*) to be aligned.

Almost done with step 1…

The last trick:
Take a big set of pre-aligned protein sequence alignments (that are correct!) and measure all of the pairwise amino acid substitution scores (the *s*(*a,b*)'s). Put them in a 20x20 ***amino acid substitution matrix :***

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

This is the **BLOSUM50** matrix.
(The numbers are scaled & rounded off to the nearest integer):

What's the score for aspartate (D) aligning with itself?
How about aspartate with phenylalanine (F)? Why?

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

Using this matrix, **we can score any alignment as the sum of scores of individual pairs of amino acids**.

For example, the top alignment in our earlier example:

```
FlgA1 EAGNVKLKRGRLDTLPPRTVLDINQLVDAISLRDLSPDQPIQLTQFRQAWRVKAGQRVNVIASGD
         ++K+K+GRLDTLPP  +L+ N    A+SLR ++  QP+     R+ W +KAGQ V V+A G+
FlgA2 TLQDIKMKQGRLDTLPPGALLEPNFAQGAVSLRQINAGQPLTRNMLRRLWIIKAGQDVQVLALGE
```

gets the score:

S(FlgA1,FlgA2) = – 1 – 2 – 2 + 2 + 4 + 6 + ... = 186

We also need to penalize **gaps**. For now, let's just use a constant penalty ***d*** for each amino acid gap in an alignment, *i. e.*:

the penalty for a gap of length g = -g*d

PAM vs. BLOSUM

Margaret Dayhoff (1925-1983)
Developed point accepted mutation matrices (PAM matrices)

Steve and Jorja Henikoff
Developed BLOSUM matrices

<u>Calibrated for different evolutionary times</u>	<u>Calibrated for different % identity sequences</u>
PAM-*n* = *n* substitutions per 100 residues	BLOSUM-*n* = for sequences of about *n* % identity
e.g. matrices from PAM1 to PAM250	averages substitution probabilities over sequence clusters, gives better estimates for highly divergent cases
measure PAM1, calculate higher PAMs from that	
<u>Explicit model of evolution</u> (calculated using a phylogenetic tree)	<u>Implicit model of evolution</u> (calculated from blocks of aligned sequences)

To align two sequences, we need to perform 3 steps:

1. We need some way to decide which alignments are better than others.
   For this, we'll invent a way to give the alignments a "score" indicating their quality.

2. **Align the two proteins so that they get the best possible score.**

3. Decide if the score is "good enough" for us to believe the alignment is biologically significant.

A sense of scale:

There are $\binom{2n}{n} \approx \frac{2^{2n}}{\sqrt{\pi n}}$ possible global alignments between two sequences of length n if we use gaps

So, with 2 sequences of length 100, that's > 10^{60} possible alignments

We'll use something called ***dynamic programming.***

This is **mathematically guaranteed** to find the best scoring alignment, and uses ***recursion.*** This means problems are broken into sub-problems, which are in turn broken into sub-problems, etc, until the simplest sub-problems can be solved.

We're going to find the best ***local*** alignment—the best matching internal alignment—without forcing all of the amino acids to align (i.e. to match ***globally***).

i.e., this $\longrightarrow$

```
ATGCAT
ATGCAT
```

Not this $\longrightarrow$

```
ACGTTATGCATGACGTA
-C---ATGCAT----T-
```

Here's the main idea:
We'll make a ***path matrix***, showing the possible alignments and their scores. There are simple rules for how to fill in the matrix.
<u>This will test all possible alignments & give us the top-scoring alignment between the two sequences</u>.

```
                 i=0                         x                      i=n
                 H    E    A    G    A    W    G    H    E    E
             0
       P  <--j=0              The path matrix will be
       A                      filled from the top left
       W                      to the bottom right
     y H
       E
       A
       E  <--j=m
```

<u>Here are the rules:</u>
For a given square in the matrix $F(i,j)$, we look at the squares to its left $F(i\text{-}1,j)$, top $F(i,j\text{-}1)$, and top-left $F(i\text{-}1,j\text{-}1)$. Each should have a score.

We consider **<u>3 possible events</u>** & **<u>choose the one scoring the highest</u>**:

(1) x_i is aligned to y_j	$F(i-1,j-1) + s(x_i,y_j)$
(2) x_i is aligned to a gap	$F(i-1,j) - d$
(3) y_j is aligned to a gap	$F(i,j-1) - d$

For this example, we'll use $d = 8$. We also set the left-most & top-most entries to zero.

<u>Just two more rules:</u>

If the score is negative, set it equal to zero.

At each step, we also keep track of which event was chosen by **drawing an arrow from the cell we just filled back to the cell which contributed its score to this one.**

That's it! Just repeat this to fill the entire matrix.

Here we go! Start with the borders & the first entry.

$$\begin{array}{cccccccccccc}
 & & H & E & A & G & A & W & G & H & E & E \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
P & 0 & 0 & & & & & & & & & \\
A & 0 & & & & & & & & & & \\
W & 0 & & & & & & & & & & \\
H & 0 & & & & & & & & & & \\
E & 0 & & & & & & & & & & \\
A & 0 & & & & & & & & & & \\
E & 0 & & & & & & & & & &
\end{array}$$

Why is this zero?
What's the score from our BLOSSUM matrix for substituting H for P?

Next round!

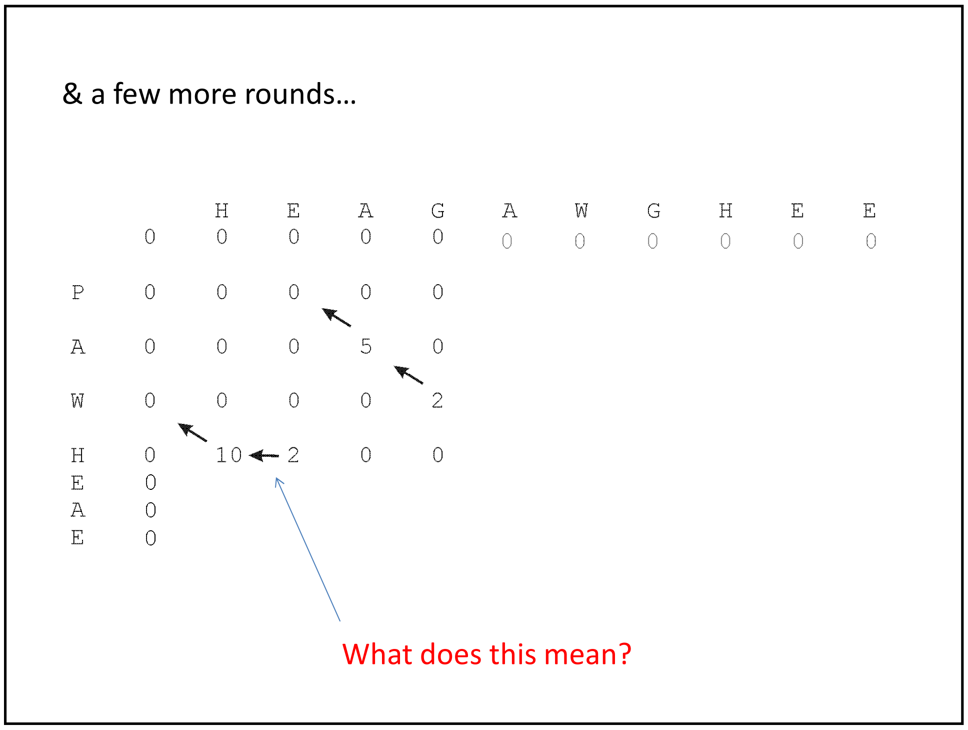
		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0								
A	0	0	0								
W	0										
H	0										
E	0										
A	0										
E	0										

Terrible! Again, none of the possible give positive scores.
We have to go a bit further in before we find a positive score…

A few more rounds, and a positive score at last!

		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							
A	0	0	0	$\nwarrow$ 5							
W	0	0	0								
H	0										
E	0										
A	0										
E	0										

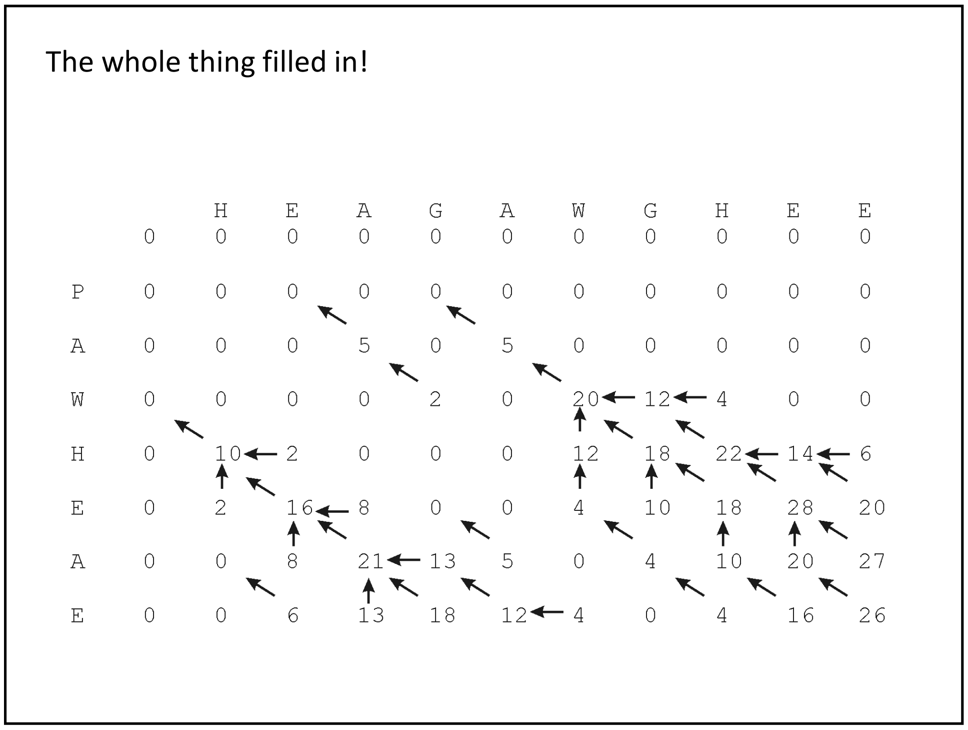
How did we get this one?



& a few more rounds…

		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						
A	0	0	0	↖5	0						
W	0	0	0	0	↖2						
H	0	↖10	←2	0	0						
E	0										
A	0										
E	0										

What does this mean?

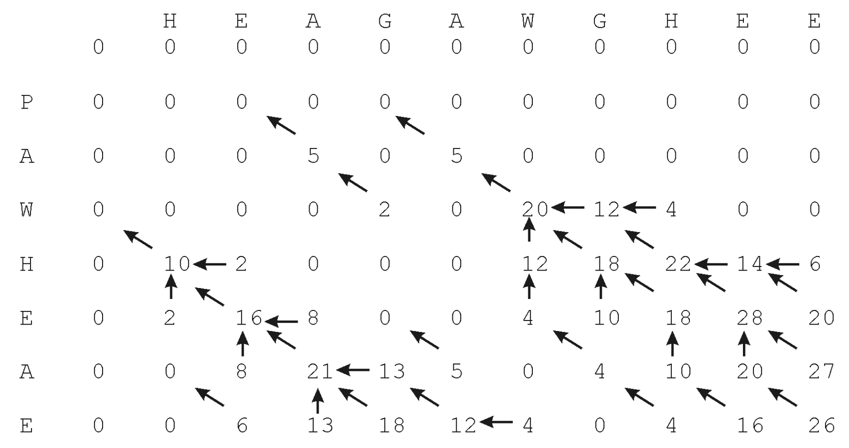


The whole thing filled in!

		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	↖5	0	↖5	0	0	0	0	0
W	0	0	0	0	↖2	0	↖20	←12	←4	0	0
H	0	↖10	←2	0	0	0	↑12	↖18	↖22	←14	←6
E	0	↑2	↖16	←8	0	0	↑4	↑10	↖18	↖28	↖20
A	0	0	↑8	↖21	←13	↖5	0	↖4	↑10	↑20	↖27
E	0	0	↖6	↑13	↖18	↖12	←4	0	↖4	↖16	↖26

Now, find the optimal alignment using a ***traceback*** process:

Look for the highest score, then follow the arrows back.
The alignment "grows" from right to left

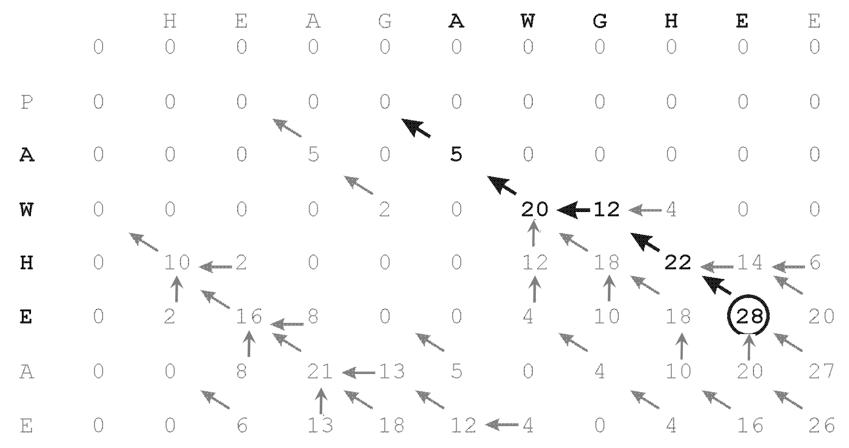


		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	5	0	0	0	0	0
W	0	0	0	0	2	0	20	12	4	0	0
H	0	10	2	0	0	0	12	18	22	14	6
E	0	2	16	8	0	0	4	10	18	28	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

This gives the following alignment:

```
AWGHE
AW-HE
```

(Note: for gaps, the arrow points to the sequence that gets the gap)



		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	5	0	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20**	**12**	4	0	0
H	0	10	2	0	0	0	12	18	**22**	14	6
E	0	2	16	8	0	0	4	10	18	**28**	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

To align two sequences, we need to perform 3 steps:

1. We need some way to decide which alignments are better than others.
   For this, we'll invent a way to give the alignments a "score" indicating their quality.

2. Align the two proteins so that they get the best possible score.

3. **Decide if the score is "good enough" for us to believe the alignment is biologically significant.**

This algorithm always gives the best alignment.

Every pair of sequences can be aligned in some fashion.

So, when is a score "good enough"?

How can we figure this out?

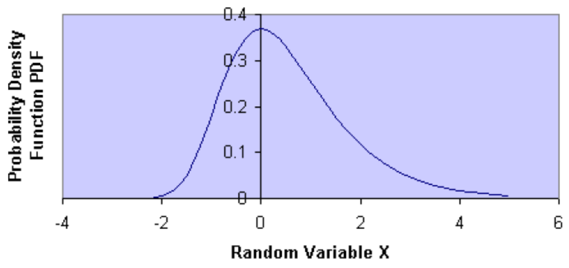
Here's one approach:

Shuffle one sequence. Calculate the best alignment & its score.
Repeat 1000 times.

If we never see a score as high as the real one, we say the real score has <1 in a 1000 chance of happening just by luck.

But if we want something that only occurs < 1 in a million, we'd have to shuffle 1,000,000 times…

Luckily, alignment scores follow a well-behaved distribution, the ***extreme value distribution***, so we can do a few trials & fit to this.



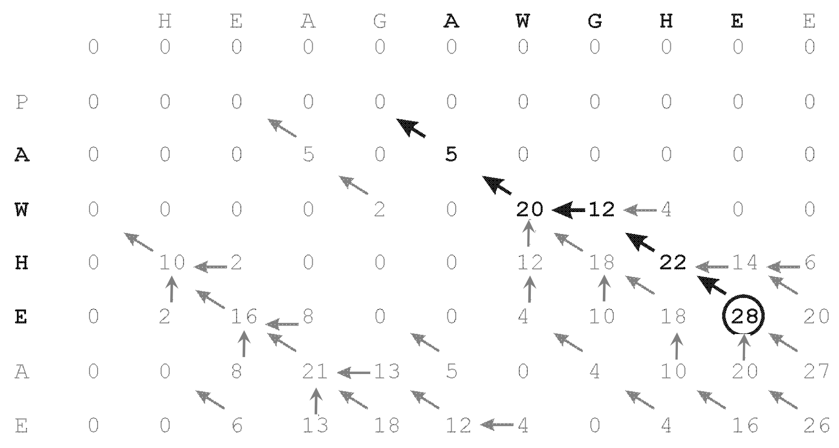
# random trials & their average score

$$p(\text{max score} \leq X) \approx e^{-\underset{\uparrow}{k}\underset{\uparrow}{N}e^{\underset{\uparrow}{\lambda}(X-\underset{\uparrow}{\mu})}}$$

This p-value gives the significance of your alignment. But, if we search a database and perform many alignments, we still need something more (next time).

Describe the shape & can be fit from a few trials

Some extensions: <u>Local</u> vs. <u>global</u> alignments
How might you force the full sequences to align?



		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	5	0	5	0	0	0	0	0
W	0	0	0	0	2	0	20	12	4	0	0
H	0	10	2	0	0	0	12	18	22	14	6
E	0	2	16	8	0	0	4	10	18	28	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

```
AWGHE
AW-HE
```

Some extensions: <u>Local</u> vs. <u>global</u> alignments
How might you force the full sequences to align?

A few tiny changes:

Initialize only the top left cell of the path matrix to zero
(not all top and left cells).

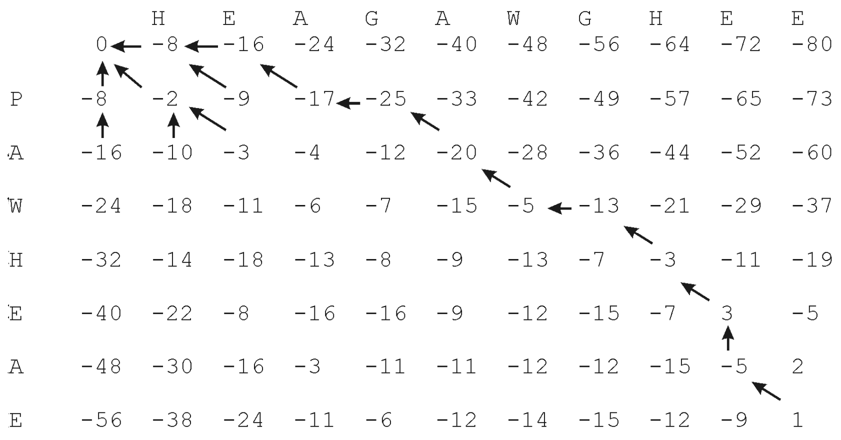
Leave the negative values (don’t set them to zero).

The optimal alignment should start at the <u>top left</u> cell and finish at the <u>bottom right</u> cell of the path matrix.

Start the trace-back at the <u>bottom right</u> cell

Some extensions: Local vs. global alignments

How might you force the full sequences to align?

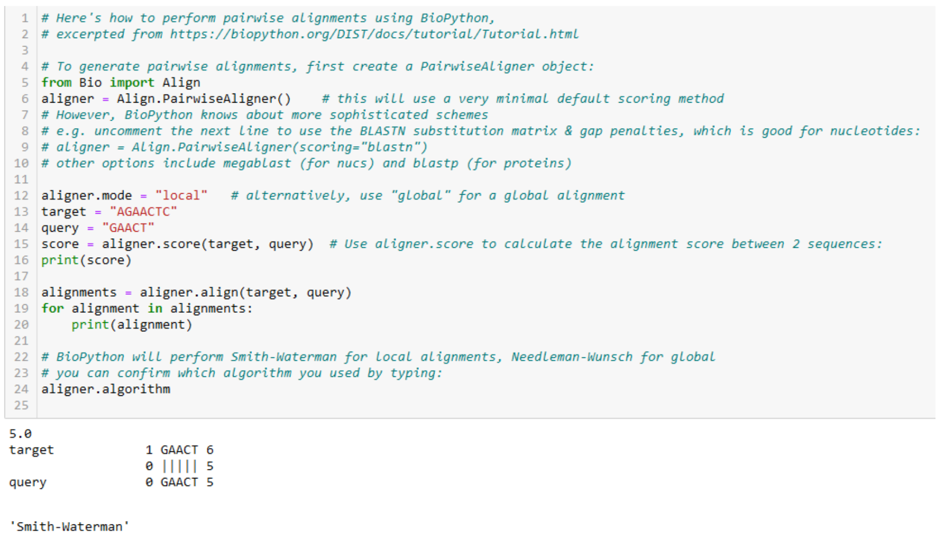


		H	E	A	G	A	W	G	H	E	E
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	-8	-2	-9	-17	-25	-33	-42	-49	-57	-65	-73
A	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
W	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
H	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
E	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
A	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

```
HEAGAWGHE-E
--P-AW-HEAE
```

How can you try this yourself using BioPython?

BioPython can perform a wide variety of sequence alignments, DNA/protein, local/global, dynamic programming, BLAST, different scoring schemes, etc, & is a great environment to learn and play with these approaches. Here's a minimal use case to start you off:



```python
# Here's how to perform pairwise alignments using BioPython,
# excerpted from https://biopython.org/DIST/docs/tutorial/Tutorial.html

# To generate pairwise alignments, first create a PairwiseAligner object:
from Bio import Align
aligner = Align.PairwiseAligner()    # this will use a very minimal default scoring method
# However, BioPython knows about more sophisticated schemes
# e.g. uncomment the next line to use the BLASTN substitution matrix & gap penalties, which is good for nucleotides:
# aligner = Align.PairwiseAligner(scoring="blastn")
# other options include megablast (for nucs) and blastp (for proteins)

aligner.mode = "local"   # alternatively, use "global" for a global alignment
target = "AGAACTC"
query = "GAACT"
score = aligner.score(target, query)  # Use aligner.score to calculate the alignment score between 2 sequences:
print(score)

alignments = aligner.align(target, query)
for alignment in alignments:
    print(alignment)

# BioPython will perform Smith-Waterman for local alignments, Needleman-Wunsch for global
# you can confirm which algorithm you used by typing:
aligner.algorithm
```

```
5.0
target            1 GAACT 6
                  0 ||||| 5
query             0 GAACT 5


'Smith-Waterman'
```

Using BioPython, you can change every aspect of the scoring & substitution matrices, as well as run BLAST locally or in the cloud.

e.g. here's the BLOSUM62 matrix, along w/ many others that BioPython knows about:

```python
from Bio.Align import substitution_matrices
substitution_matrices.load()
['BENNER22', 'BENNER6', 'BENNER74', 'BLASTN', 'BLASTP', 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', ..., 'TRANS']
matrix = substitution_matrices.load("BLOSUM62")
print(matrix)
```

```
#  Matrix made by matblas from blosum62.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 62
#  Entropy =   0.6979, Expected =  -0.5209
     A    R    N    D    C    Q    E    G    H    I    L    K    M    F    P    S    T    W    Y    V    B    Z    X    *
A  4.0 -1.0 -2.0 -2.0  0.0 -1.0 -1.0  0.0 -2.0 -1.0 -1.0 -1.0 -1.0 -2.0 -1.0  1.0  0.0 -3.0 -2.0  0.0 -2.0 -1.0  0.0 -4.0
R -1.0  5.0  0.0 -2.0 -3.0  1.0  0.0 -2.0  0.0 -3.0 -2.0  2.0 -1.0 -3.0 -2.0 -1.0 -1.0 -3.0 -2.0 -3.0 -1.0  0.0 -1.0 -4.0
N -2.0  0.0  6.0  1.0 -3.0  0.0  0.0  0.0  1.0 -3.0 -3.0  0.0 -2.0 -3.0 -2.0  1.0  0.0 -4.0 -2.0 -3.0  3.0  0.0 -1.0 -4.0
D -2.0 -2.0  1.0  6.0 -3.0  0.0  2.0 -1.0 -1.0 -3.0 -4.0 -1.0 -3.0 -3.0 -1.0  0.0 -1.0 -4.0 -3.0 -3.0  4.0  1.0 -1.0 -4.0
C  0.0 -3.0 -3.0 -3.0  9.0 -3.0 -4.0 -3.0 -3.0 -1.0 -1.0 -3.0 -1.0 -2.0 -3.0 -1.0 -1.0 -2.0 -2.0 -1.0 -3.0 -3.0 -2.0 -4.0
Q -1.0  1.0  0.0  0.0 -3.0  5.0  2.0 -2.0  0.0 -3.0 -2.0  1.0  0.0 -3.0 -1.0  0.0 -1.0 -2.0 -1.0 -2.0  0.0  3.0 -1.0 -4.0
E -1.0  0.0  0.0  2.0 -4.0  2.0  5.0 -2.0  0.0 -3.0 -3.0  1.0 -2.0 -3.0 -1.0  0.0 -1.0 -3.0 -2.0 -2.0  1.0  4.0 -1.0 -4.0
G  0.0 -2.0  0.0 -1.0 -3.0 -2.0 -2.0  6.0 -2.0 -4.0 -4.0 -2.0 -3.0 -3.0 -2.0  0.0 -2.0 -2.0 -3.0 -3.0 -1.0 -2.0 -1.0 -4.0
H -2.0  0.0  1.0 -1.0 -3.0  0.0  0.0 -2.0  8.0 -3.0 -3.0 -1.0 -2.0 -1.0 -2.0 -1.0 -2.0 -2.0  2.0 -3.0  0.0  0.0 -1.0 -4.0
I -1.0 -3.0 -3.0 -3.0 -1.0 -3.0 -3.0 -4.0 -3.0  4.0  2.0 -3.0  1.0  0.0 -3.0 -2.0 -1.0 -3.0 -1.0  3.0 -3.0 -3.0 -1.0 -4.0
L -1.0 -2.0 -3.0 -4.0 -1.0 -2.0 -3.0 -4.0 -3.0  2.0  4.0 -2.0  2.0  0.0 -3.0 -2.0 -1.0 -2.0 -1.0  1.0 -4.0 -3.0 -1.0 -4.0
K -1.0  2.0  0.0 -1.0 -3.0  1.0  1.0 -2.0 -1.0 -3.0 -2.0  5.0 -1.0 -3.0 -1.0  0.0 -1.0 -3.0 -2.0 -2.0  0.0  1.0 -1.0 -4.0
M -1.0 -1.0 -2.0 -3.0 -1.0  0.0 -2.0 -3.0 -2.0  1.0  2.0 -1.0  5.0  0.0 -2.0 -1.0 -1.0 -1.0 -1.0  1.0 -3.0 -1.0 -1.0 -4.0
F -2.0 -3.0 -3.0 -3.0 -2.0 -3.0 -3.0 -3.0 -1.0  0.0  0.0 -3.0  0.0  6.0 -4.0 -2.0 -2.0  1.0  3.0 -1.0 -3.0 -3.0 -1.0 -4.0
P -1.0 -2.0 -2.0 -1.0 -3.0 -1.0 -1.0 -2.0 -2.0 -3.0 -3.0 -1.0 -2.0 -4.0  7.0 -1.0 -1.0 -4.0 -3.0 -2.0 -2.0 -1.0 -2.0 -4.0
S  1.0 -1.0  1.0  0.0 -1.0  0.0  0.0  0.0 -1.0 -2.0 -2.0  0.0 -1.0 -2.0 -1.0  4.0  1.0 -3.0 -2.0 -2.0  0.0  0.0  0.0 -4.0
T  0.0 -1.0  0.0 -1.0 -1.0 -1.0 -1.0 -2.0 -2.0 -1.0 -1.0 -1.0 -1.0 -2.0 -1.0  1.0  5.0 -2.0 -2.0  0.0 -1.0 -1.0  0.0 -4.0
W -3.0 -3.0 -4.0 -4.0 -2.0 -2.0 -3.0 -2.0 -2.0 -3.0 -2.0 -3.0 -1.0  1.0 -4.0 -3.0 -2.0 11.0  2.0 -3.0 -4.0 -3.0 -2.0 -4.0
Y -2.0 -2.0 -2.0 -3.0 -2.0 -1.0 -2.0 -3.0  2.0 -1.0 -1.0 -2.0 -1.0  3.0 -3.0 -2.0 -2.0  2.0  7.0 -1.0 -3.0 -2.0 -1.0 -4.0
V  0.0 -3.0 -3.0 -3.0 -1.0 -2.0 -2.0 -3.0 -3.0  3.0  1.0 -2.0  1.0 -1.0 -2.0 -2.0  0.0 -3.0 -1.0  4.0 -3.0 -2.0 -1.0 -4.0
B -2.0 -1.0  3.0  4.0 -3.0  0.0  1.0 -1.0  0.0 -3.0 -4.0  0.0 -3.0 -3.0 -2.0  0.0 -1.0 -4.0 -3.0 -3.0  4.0  1.0 -1.0 -4.0
Z -1.0  0.0  0.0  1.0 -3.0  3.0  4.0 -2.0  0.0 -3.0 -3.0  1.0 -1.0 -3.0 -1.0  0.0 -1.0 -3.0 -2.0 -2.0  1.0  4.0 -1.0 -4.0
X  0.0 -1.0 -1.0 -1.0 -2.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -2.0  0.0  0.0 -2.0 -1.0 -1.0 -1.0 -1.0 -1.0 -4.0
* -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0 -4.0  1.0
```

Putting it all together, here's the example alignment we did manually

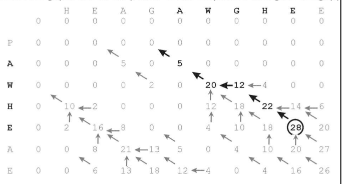
```python
from Bio import Align
from Bio.Align import substitution_matrices

aligner = Align.PairwiseAligner()
aligner.mode = "local"
matrix = substitution_matrices.load("BLOSUM50")

aligner.substitution_matrix = matrix
aligner.target_internal_open_gap_score = -8.000000
aligner.target_internal_extend_gap_score = -8.000000
aligner.target_left_open_gap_score = -8.000000
aligner.target_left_extend_gap_score = -8.000000
aligner.target_right_open_gap_score = -8.000000
aligner.target_right_extend_gap_score = -8.000000
aligner.query_internal_open_gap_score = -8.000000
aligner.query_internal_extend_gap_score = -8.000000
aligner.query_left_open_gap_score = -8.000000
aligner.query_left_extend_gap_score = -8.000000
aligner.query_right_open_gap_score = -8.000000
aligner.query_right_extend_gap_score = -8.000000

target = "HEAGAWGHEE"
query = "PAWHEAE"
score = aligner.score(target, query)
print(score)

alignments = aligner.align(target, query)
for alignment in alignments:
    print(alignment)
```

```
28.0
target            4 AWGHE 9
                  0 ||-|| 5
query             1 AW-HE 5
```

Here was our earlier version:

This gives the following alignment:
```
AWGHE
AW-HE
```

(Note: for gaps, the arrow points to the sequence that gets the gap)

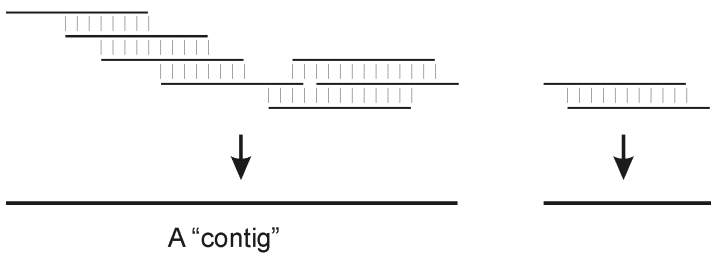


You can read more about using BioPython for sequence analyses & get example code at:

https://biopython.org/DIST/docs/tutorial/Tutorial.html

Chapter 7 is all about how to perform pairwise sequence alignments

Some extensions:
What about overlapping sequences?
e.g. as in 'shotgun sequencing' genomes where 'contigs' are built up from overlapping sequences



Some extensions:
What about overlapping sequences?

Modify global alignment to not penalize overhangs:

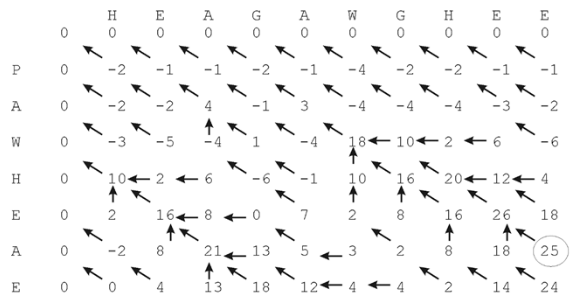
The optimal alignment should start at the top or left edge and finish at the bottom or right edge of the path matrix.

Set these boundary conditions :

$F(i,0) = 0$ for $i=1$ to n
$F(0,j) = 0$ for $j=1$ to m

Start the traceback at the cell with the highest score on the right or bottom border

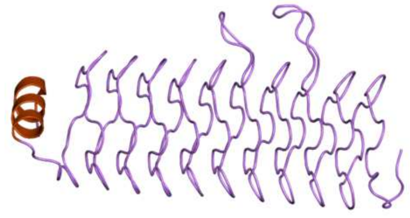
Some extensions:
What about overlapping sequences?
e.g. as in 'shotgun sequencing' genomes where 'contigs' are built up from overlapping sequences



		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	-2	-1	-1	-2	-1	-4	-2	-2	-1	-1
A	0	-2	-2	4	-1	3	-4	-4	-4	-3	-2
W	0	-3	-5	-4	1	-4	18	10	2	6	-6
H	0	10	2	6	-6	-1	10	16	20	12	4
E	0	2	16	8	0	7	2	8	16	26	18
A	0	-2	8	21	13	5	3	2	8	18	25
E	0	0	4	13	18	12	4	4	2	14	24

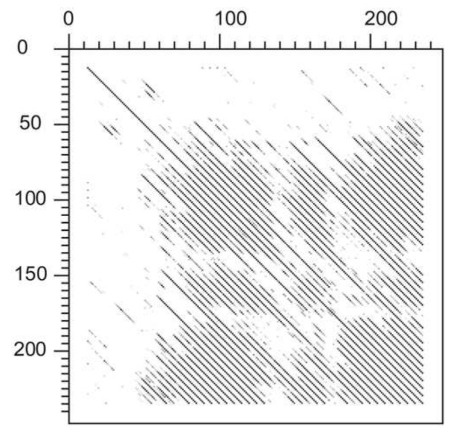
```
(overhang = HEA)    GAWGHEE
                    PAW-HEA   (overhang = E)
```

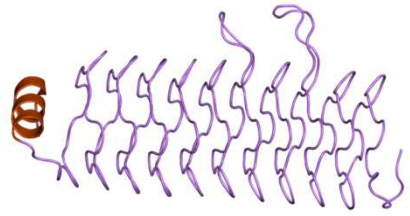
Some extensions:
How might you find repetitive sequences?



Structure of the pentapeptide repeat protein HetL
(from wiki, PMID18952182)

Align the sequence to itself and ignore the diagonal (optimal) alignment
→ High-scoring off-diagonal alignments will be repeats





Structure of the pentapeptide repeat protein HetL
(from wiki, PMID18952182)

Dot plot (quick visualization of sequence similarity) of the pentapeptide repeat protein HglK protein vs. itself
(http://en.wikipedia.org/wiki/Pentapeptide_repeat)