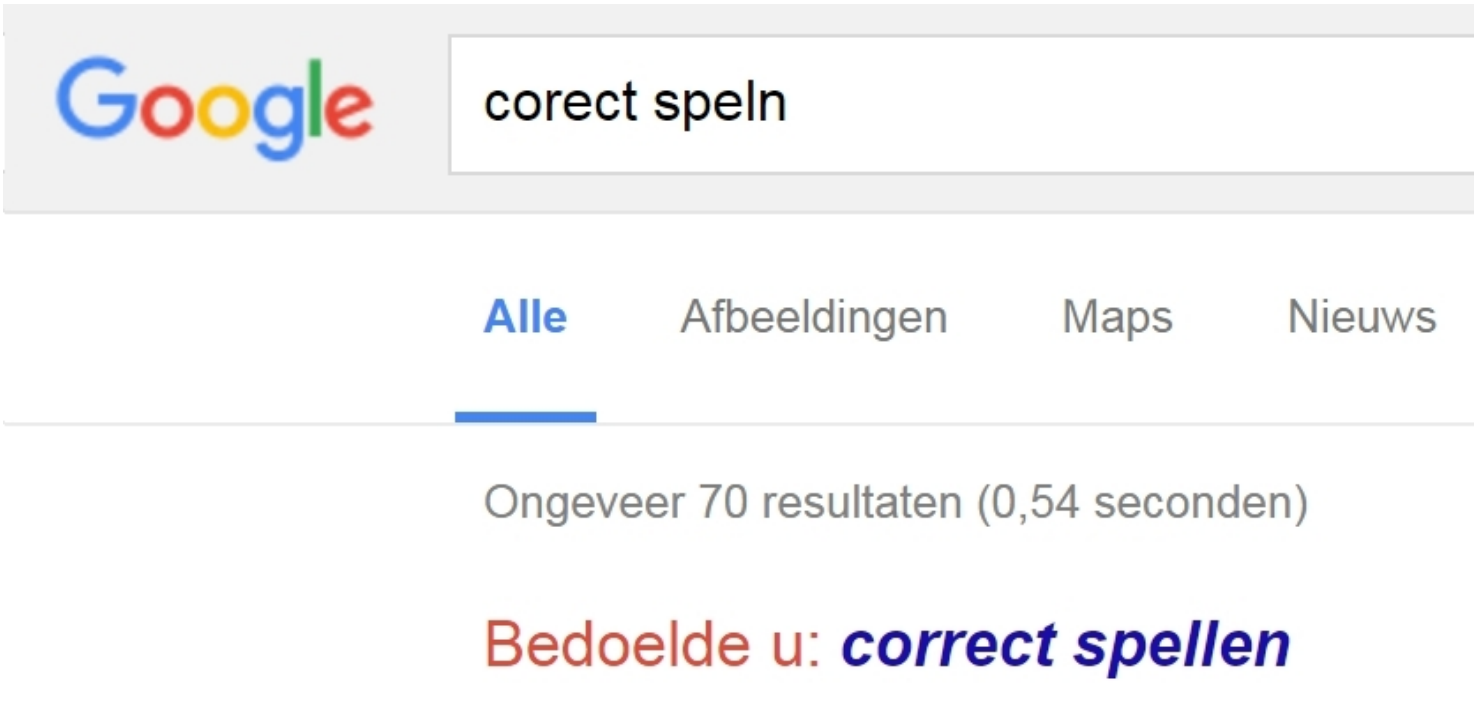
Data-analysis and Retrieval
Case study k-grams:
Biological sequence alignment

Hans Philippi

May 15, 2024

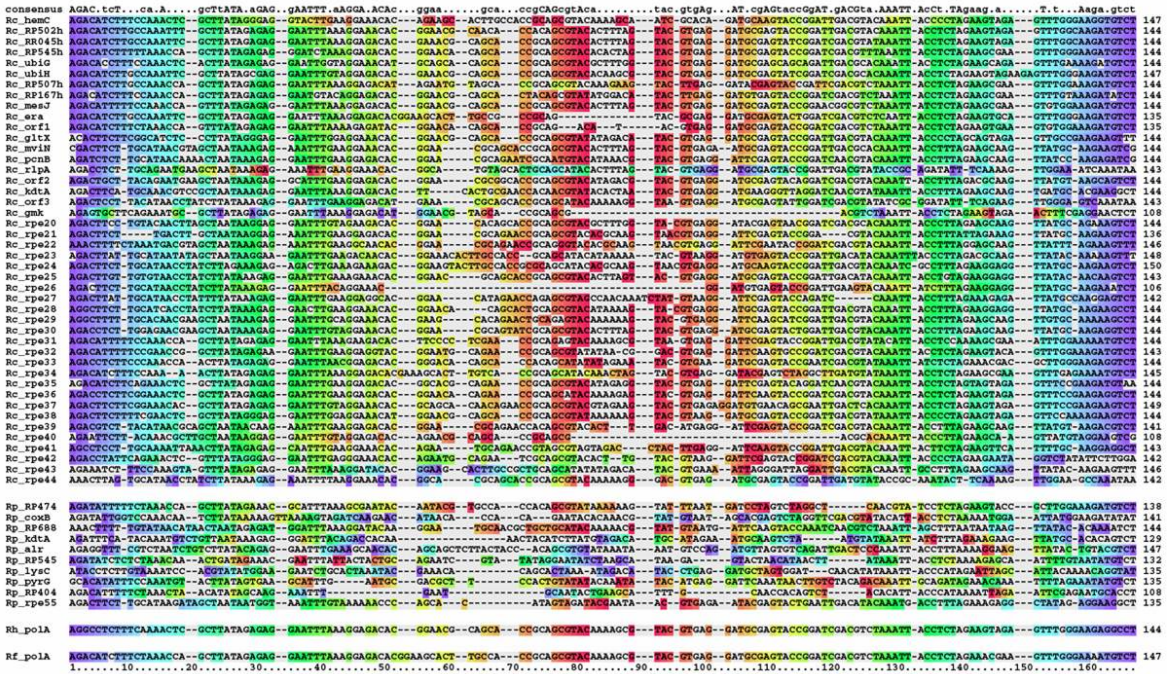
Text search

- So far, we considered exact text search ...
- ... supported by indexing techniques ...
- ... and possibly with wildcards
- But (almost) everyone knows this phenomenon:



Approximate string matching

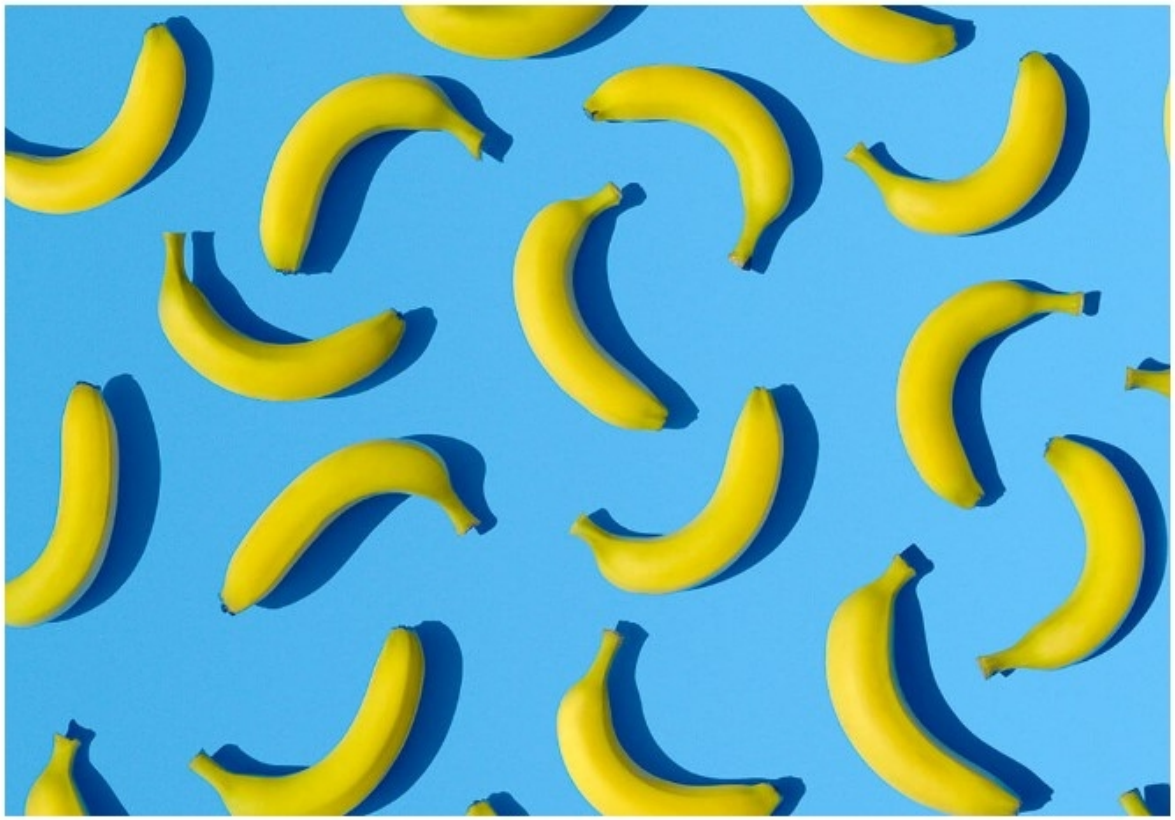
- Application: automatic spelling correction
- Can be solved using dynamic programming techniques
- But in large scale applications, this may be computationally (too) heavy
- Heuristic indexing techniques based on k-grams
- Application: biological sequence alignment



Sequence alignment: searching for homologies

How Genetically Related Are We to Bananas?

Share



https://www.pfizer.com/news/articles/how_genetically_related_are_we_to_bananas

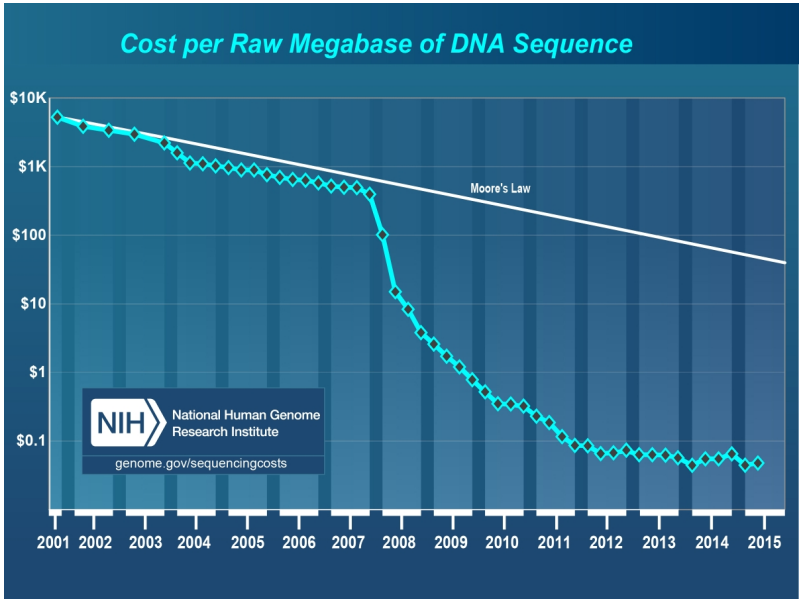
Sequence alignment: searching for homologies

Queries in context:

- *We have a number of patients with disease X. Can we find a sequence that is common in their genomes and that is different from corresponding sequences in genomes of non-patients?*
- *We have a new virus that has a lot properties in common with some known viruses. Can we find the differences in genetic properties? Can these differences be the result of a plausible sequence of spontaneous mutations, or is it likely to be engineered?*

DNA sequencing: milestones

- 1953 Crick & Watson discover the molecular structure of DNA (double helix)
- 1977 Sanger pioneers with sequencing techniques
- 2000-2003 human genome sequenced
- 2008-2015 dramatic decrease of sequencing cost
- Currently: e.g. individual DNA analysis

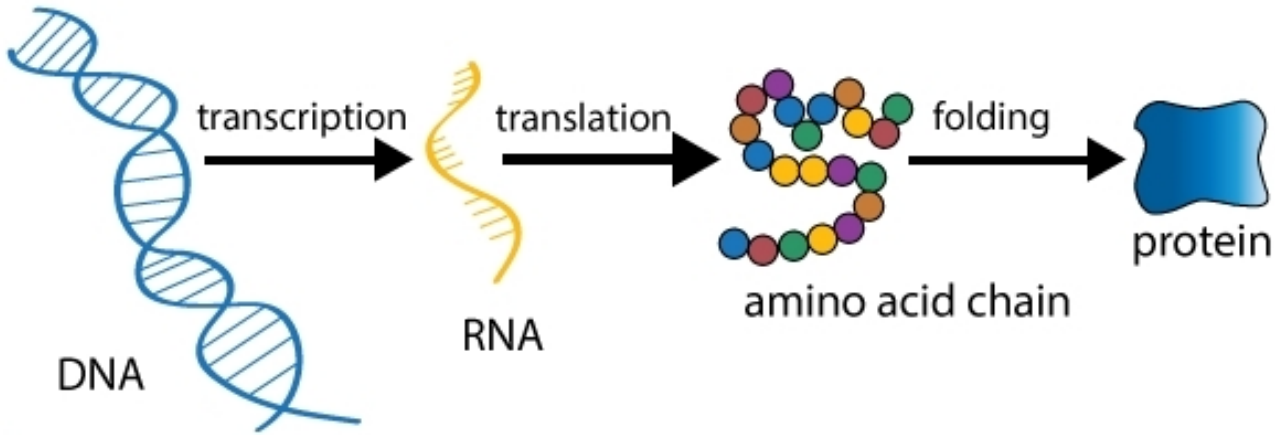


Some genetics

- The Model: a DNA sequence is a string over the alphabet {A,C,G,T}
- Each of the letters represents a *base*, in the chemical sense
- A = adenine, G = guanine, C = cytosine and T = thymine
- T in DNA corresponds to U (uracil) in RNA
- A gene is a part of the genome that codes for a specific protein
- The length of a gene varies from a few hundreds to several thousands characters
- Example: ATGGGCGTGATCAAGCCCGACATGAAGATC...
- Background reading: *Altman, Computer Applications in Molecular Biology*

Some genetics

- Gene expression: a part of the genome is copied as messenger-RNA (transcription)
- The ribosome translates the mRNA code into a protein



Genetics: protein coding

- A protein is represented by a sequence of amino acids. On earth, 20 different amino acids are known. Each of the amino acids is identified by a unique letter.
- A codon is a triplet of base characters. There are $4^3 = 64$ different codons
- Each codon determines an amino acid. Most amino acids are represented by more than codon
- A DNA/RNA string that encodes a protein is can be seen as a sequence of codons, for instance `ATGACCAGGATCTTTAAGTGA` ...
- ... can be read as `ATG-ACC-AGG-ATC-TTT-AAG-TGA`

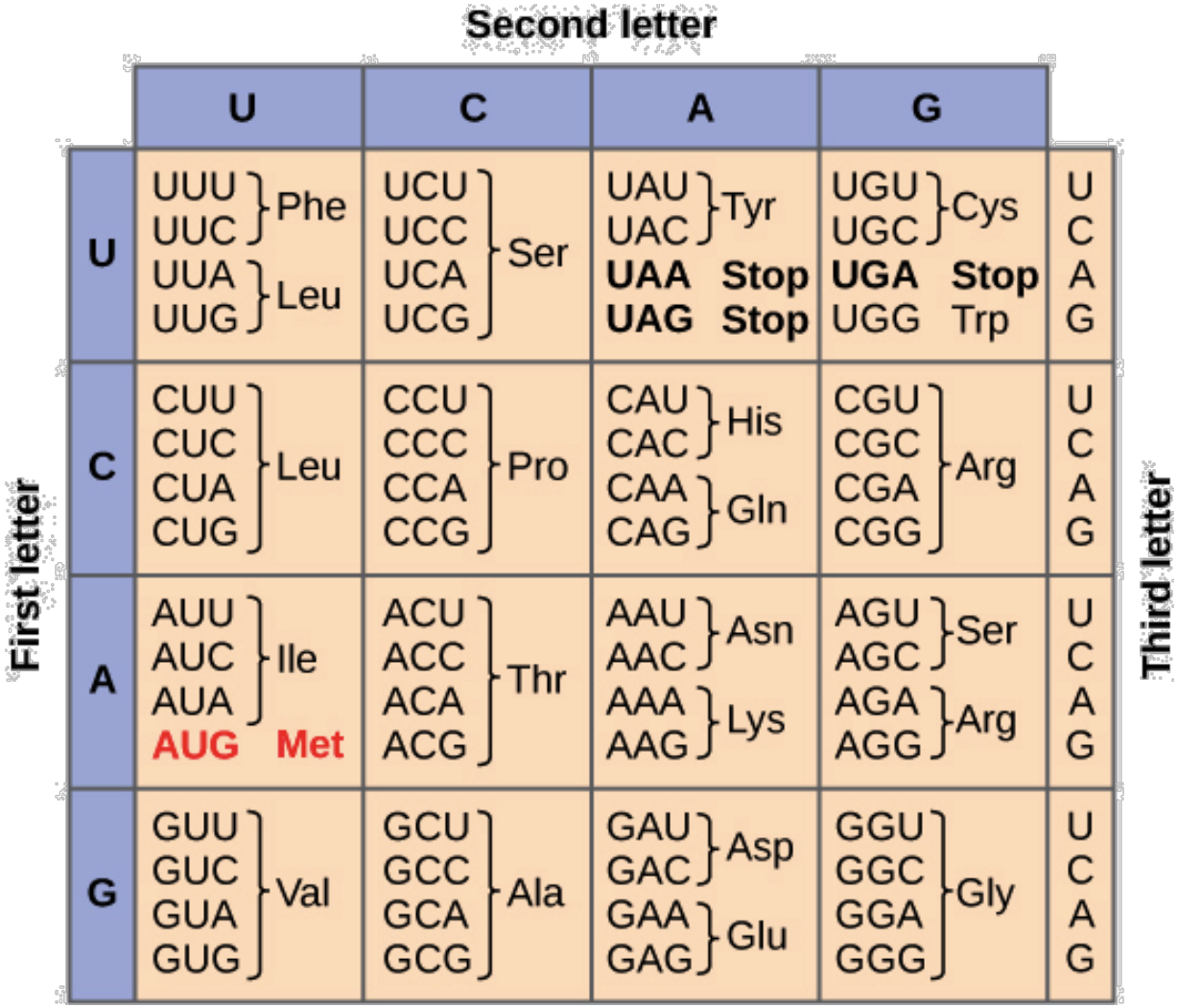
Reference: `https://en.wikipedia.org/wiki/Genetic_code`

Genetics: protein coding

- A DNA/RNA string that encodes a protein is can be seen as a sequence of codons, for instance `ATG-ACC-AGG-ATC-TTT-AAG-TGA`
- `ATG` is start codon; `TGA` is stop codon
- translated to amino acids: methionine-threonine-arginine-...
- encoded to string representing amino acids: MTR...

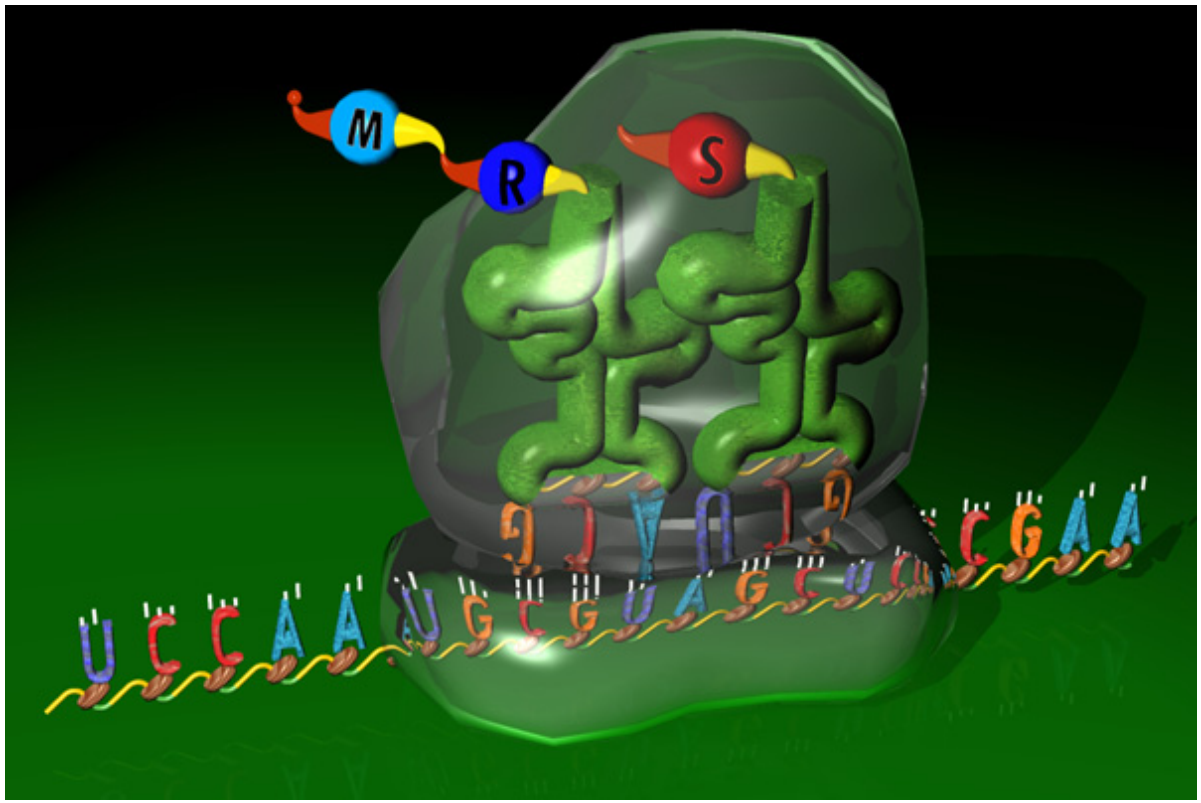
Reference: `https://en.wikipedia.org/wiki/DNA_codon_table`

Genetics: the codon table



Some genetics

- The ribosome translates the mRNA triplet code into a protein



Biological sequence databases

- SWISSPROT, GENBANK
- Contain both protein sequences and base sequences
- Generic query:
  Find protein sequences similar to
  MKYMTVTDLNNAGATV...

Biological sequence databases

SWISSPROT example entries (FASTA format):

```
>gi|1171675|sp|P42268|NDD_BPR70 NUCLEAR DISRUPTION PROTEIN
MKYMTVTDLNNAGATVIGTIKGGEWFLGTPHKDILSKPGFYFLVSEFDGSCV
SARFYVGNQRSKQGFSAVLSHIRQRRSQLARTIANNNMAYTVFYLPASKMKP
LTTGFGKGQLALAFTRNHHSEYQTLEEMNRMLADNFKFVLQAY
>gi|123527|sp|P05228|HRP2_PLAFA HISTIDINE-RICH PROTEIN PRECURSOR
(CLONE PFHRP-III)
MVSFSKNKVLSAAVFASVLLLDNNNSEFNNNLFSKNAKGLNSNKRLLHESQA
HAGDAHHAHHVADAHHAHHVADAHHAHHAANAHHAANAHHAANAHHAANAHH
AANAHHAANAHHAANAHHAANAHHAANAHHAANAHHAANAHHAANAHHAANA
HHAADANHGFHFNLHDNNSHTLHHAKANACFDDSHHDDAHHDGAHHDDAHHD
GAHHDDAHHDGAHHDGAHHDGAHHNATTHHLHMKYMTVTDLNNAGATV
```

Protein based sequence similarity

- Two DNA fragments are *homologous* if they show similarities based on common descent
- Below left, we see two homologous fragments. Not only do we have eight matching letters, also the S-N, Q-K and G-A pairings are likely (+) due to electrochemical properties of the amino acids
- We are looking for a formal notion of *sequence similarity* that comprises *letter distance* and *gapping*

```
seq1:  GSAQVKGHGKKVA
mtch:  G+ +VK+HGKKV
seq2:  GNPKVKAHGKKVL
```

```
seq3:  HV---D--DMPNAL
mtch:  ++           +L
seq4:  QLQVTGVVVTDATL
```

Scoring

- There exists a classic solution for approximate string matching based on dynamic programming and edit distance
- We will refine this approach according to the specific properties of this domain, especially the letter distance
- Analysis of collections of protein strings provides us with probabilities of letters, when picking them randomly
- Analysis of collections of protein strings that represent known homologies provides us with probabilities of letter pairings

Reference: `http://en.wikipedia.org/wiki/Edit_distance`

Scoring model

- Each string x consists of a list of symbols x_i
- Symbol a has probability q_a, based on relative frequency
- A pair of symbols a, b has combined probability p_{ab} under the Match assumption, expressing the probability to be seen together in case of homology

Scoring model

- According to the *Random Model R*, the probability to observe x and y is

$$P(x, y|R) = \prod_i q_{x_i} \prod_i q_{y_i}$$

- According to the *Match Model M*, the probability to observe x and y is related to the probability of the pairings

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

Scoring model

- The *odds-ratio*

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

  is an indicator for homology
- For mathematical reasons, we prefer to do our calculations in log-space

Scoring model: log space

- The *odds-ratio*

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

- The *log-odds ratio* for character pairs

$$s(a,b) = log(\frac{p_{ab}}{q_a q_b})$$

- The *log-odds ratio* for strings x and y

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

Scoring model: Blocks Substitution Matrix (BLOSUM)

- BLOSUM matrices represent log-odds ratios
- Several variants, for instance:
  - BLOSUM80: used for strongly related proteins
  - BLOSUM62: midrange
  - BLOSUM45: distantly related proteins
- Below you see a part of the BLOSUM50 matrix
- D, E and K charged; V, I and L hydrophobe

	D	E	K	V	I	L
D	8	2	-1	-4	-4	-4
E	2	6	1	-3	-4	-3
K	-1	1	6	-3	-3	-3
V	-4	-3	-3	5	4	1
I	-4	-4	-3	4	5	2
L	-4	-3	-3	1	2	5

Reference: `https://en.wikipedia.org/wiki/BLOSUM`

Scoring model: gaps

```
seq1:  GSAQVKGHGKKVA | seq3:  HV---D--DMPNAL
mtch:  G+ +VK+HGKKV  | mtch:  ++           +L
seq2:  GNPKVKAHGKKVL | seq4:  QLQVTGVVVTDATL
```

- We give a penalty for gaps with length g

$$\gamma(d) = -gd$$

- Based on empirical tuning, $d = 8$ is often suggested

Intermezzo: example alignments

Using BLOSUM50 and $\gamma(d) = -gd, d = 8$, calculate the scores for the following alignments:

```
seq1:  GSAQVKGHGKKVA | seq3:  HV---D--DMPNAL
mtch:  G+ +VK+HGKKV  | mtch:  ++           +L
seq2:  GNPKVKAHGKKVL | seq4:  QLQVTGVVVTDATL
```

Intermezzo: example alignments

Using BLOSUM50 and $\gamma(d) = -gd, d = 8$, calculate the scores for the following alignments:

```
seq1:  GSAQVKGHGKKVA
mtch:  G+ +VK+HGKKV
seq2:  GNPKVKAHGKKVL
```

$8+1-1+2+5+6+0+10+8+6+6+5-2$
$= 54$

```
seq3:  HV---D--DMPNAL
mtch:  ++           +L
seq4:  QLQVTGVVVTDATL
```

$1+1-24-1-16-4-1-1-1+0+5$
$=$ -41

Alignment algorithms

- Gapping enlarges the search space dramatically
- But, we can apply *dynamic programming*
- The optimal alignment between strings $x = x_1 \ldots x_m$ and $y = y_1 \ldots y_n$ can be expressed in the optimal alignments of subsequences of x and y

Alignment algorithms: Needleman-Wunsch

- Suppose we know optimal alignments for
  - $x_1 \ldots x_{m-1}$ and $y_1 \ldots y_{n-1}$
  - $x_1 \ldots x_m$ and $y_1 \ldots y_{n-1}$
  - $x_1 \ldots x_{m-1}$ and $y_1 \ldots y_n$
- The optimal alignment for $x_1 \ldots x_m$ and $y_1 \ldots y_n$ can be determined by choosing the best option from:
  - solution for $x_1 \ldots x_{m-1}$ and $y_1 \ldots y_{n-1}$; pair x_m with y_n
  - solution for $x_1 \ldots x_{m-1}$ and $y_1 \ldots y_n$; pair x_m with gap
  - solution for $x_1 \ldots x_m$ and $y_1 \ldots y_{n-1}$; pair y_n with gap

Needleman-Wunsch

- Now we can fill the dynamic programming matrix from upper left to bottom right
- The score F for entry i, j can be calculated as follows:

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

- Arrows indicate which of the three options was chosen for the calculation of $F(i,j)$

Needleman-Wunsch: example

Align HEAGAWGHEE with PAWHEAE $\gamma(d) = -gd, d = 8$

Initialized matrix:

	-	H	E	A	...
-	0	← -8	← -16	← -24	
P	↑ -8				
A	↑ -16				
W	↑ -24				
...					

Needleman-Wunsch: example

Align HEAGAWGHEE with PAWHEAE $\quad \gamma(d) = -gd, d = 8$

One step:

	-	H	E	A	...
-	0	← -8	← -16	← -24	
P	↑ -8	↖ -2			
A	↑ -16				
W	↑ -24				
...					

Needleman-Wunsch: example

Align HEAGAWGHEE with PAWHEAE $\quad \gamma(d) = -gd, d = 8$

Two more steps:

	-	H	E	A	...
-	0	← -8	← -16	← -24	
P	↑ -8	↖ -2	↖ -9		
A	↑ -16	↑ -10			
W	↑ -24				
...					

Needleman-Wunsch: example

Align HEAGAWGHEE with PAWHEAE $\gamma(d) = -gd, d = 8$

Six more steps:

	-	H	E	A	...
-	0	← -8	← -16	← -24	
P	↑ -8	↖ -2	↖ -9	↖ -17	
A	↑ -16	↑ -10	↖ -3	↖ -4	
W	↑ -24	↑ -18	↑ -11	↖ -6	
...					

Needleman-Wunsch: example

Align HEAGAWGHEE with PAWHEAE $\qquad \gamma(d) = -gd, d = 8$

default is ↖

	-	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**

Needleman-Wunsch: example

- Alignment is finished when lower right field is reached
- This field contains the alignment score: $+1$
- Time complexity is $O(mn)$
- Backward arrows indicate the alignment path
- Corresponding alignment ?

Needleman-Wunsch: example

- Alignment is finished when lower right field is reached
- This field contains the alignment score: $+1$
- Time complexity is $O(mn)$
- Backward arrows indicate the alignment path
- Corresponding alignment:

```
seq1:  HEAGAWGHE-E
seq2:  --P-AW-HEAE
```

Global versus local alignment

- Global alignment: score $= +1$:

```
seq1:   HEAGAWGHE-E
seq2:   --P-AW-HEAE
```

- Local alignment: score $= +21$:

```
seq1:   HEA
seq2:   HEA
```

- Local matches are much more interesting, especially when comparing a relatively short query string to a long database string

Local alignment: Smith-Waterman

- Question: How do you adapt Needleman-Wunsch to find local matches?
- Remember: the score F according to N-W for entry i, j can be calculated as follows:

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

Local alignment: Smith-Waterman

- Question: How do you adapt Needleman-Wunsch to find local matches?
- Make it possible to start anywhere in the matrix from scratch, i.e. with score $= 0$
- Make it possible to stop anywhere in the matrix
- The score F for entry i, j can be calculated as follows:

$$F(i,j) = max \begin{cases} 0, & (start\ new\ alignment) \\ F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases}$$

Smith-Waterman: example

Align HEAGAWGHEE with PAWHEAE $\gamma(d) = -gd, d = 8$

A less exciting start:

	-	H	E	A	...
-	0	0	0	0	
P	0	0	0	0	
A	0	0	0		
W	0	0			

Smith-Waterman: example

Align HEAGAWGHEE with PAWHEAE $\gamma(d) = -gd, d = 8$

	-	H	E	A	...
-	0	0	0	0	
P	0	0	0	0	
A	0	0	0	5	
W	0	0	0	0	

Smith-Waterman: example

Align HEAGAWGHEE with PAWHEAE $\gamma(d) = -gd, d = 8$

default is $\nwarrow$; backpointer irrelevant for zero fields

	-	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	$\uparrow 12$	18	22	<14	<6
E	0	$\uparrow 2$	16	<8	0	0	$\uparrow 4$	$\uparrow 10$	18	28	20
A	0	0	$\uparrow 8$	21	<13	5	0	4	$\uparrow 10$	$\uparrow 20$	27
E	0	0	6	$\uparrow 13$	18	12	<4	0	4	16	26

Alignment = ?

Smith-Waterman: example

Align HEAGAWGHEE with PAWHEAE

	-	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

Alignment:

```
seq1:  AWGHE
seq2:  AW-HE
```

Heuristics

- Complexity of dynamic programming algorithms is $O(mn)$, where m = length of query string and n = length of database
- Unsatisfying for large databases, heuristic required
- Two step approach
  - step 1 (filtering): select a number of promising candidate sections in the database
  - step 2 (expansion): apply further analysis to select best matches to query
- *Blast*-approach: heuristic based on k-gram filtering
- k = 3 for protein string matching (20 char alphabet)
- k = 11 for base string matching (ACGT alphabet)

Heuristics: BLAST

- Example: 3-gram match (a *hit*) between HEAGAWGHEE and PAWHEAE
- A hit points to positions in x and y that are candidates for further processing by an expansion algorithm
- First observation: if size of k
  - increases, then precision increases, recall decreases
  - decreases, then precision decreases, recall increases

BLAST Protein search

- Blast uses 3-grams for protein matching and 11-gram for base string matching
- Having just one 3-gram match (hit) between two strings gives a lot of false positives
- Blast applies other techniques to influence precision and recall

BLAST increasing precision

- *Two hit diagonal* principle
- A database string is a candidate when it shares two hits with the query string *on the same diagonal*, i.e. with the same distance between the hits
- OK:

```
qu:   CWYWRWYYC
db:   RRWYWAWYYRR
```

- Wrong:

```
qu:   CWYWRWYYC
db:   RRWYWABCWYYRR
```

BLAST increasing recall

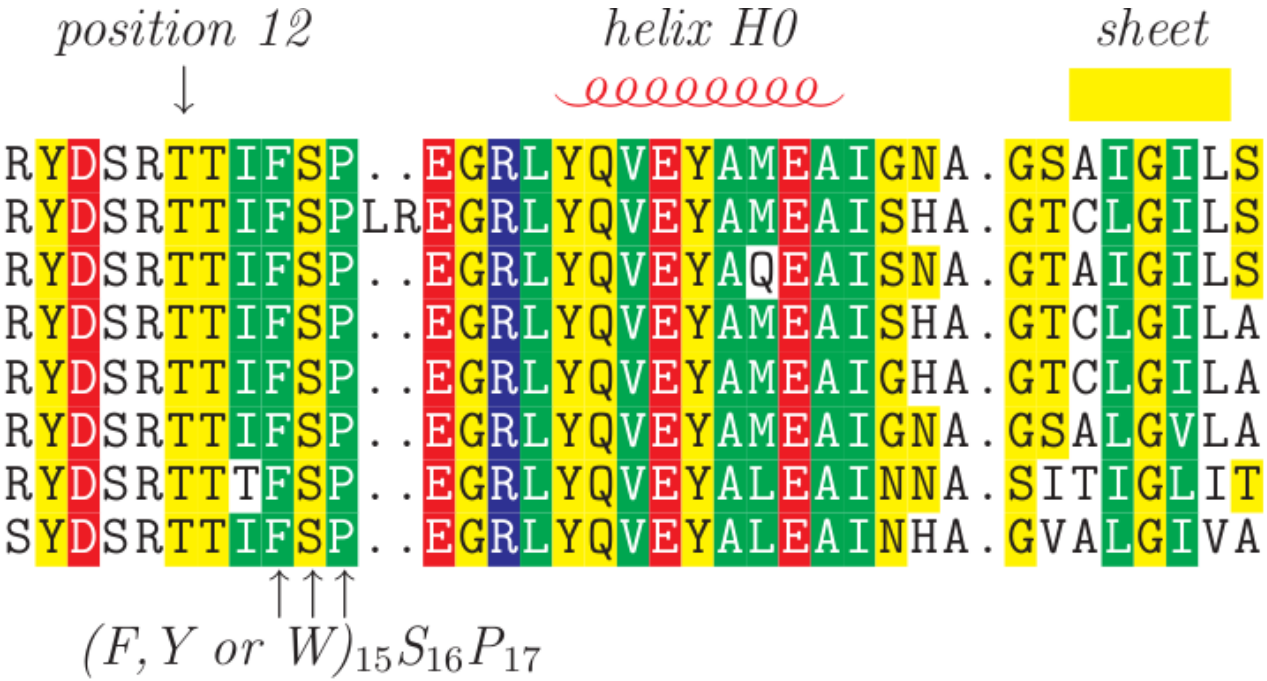
- Extended version of hit notion
- Two 3-grams match if their score exceeds a threshold (default $= 11$)
- Example: `HEAGAWGHEE` and `PAWHEAE`
- Score for `GAW` - `PAW` is $-2 + 5 + 15 = +18$

BLAST scores & probabilities

- Intuitively, it is clear that a high score ($\gg 0$) indicates a homology, ...
- ... whereas a negative score makes it unlikely.
- Altschul (see references) gives a way to calculate probabilities from raw scores, ...
- ... but we will not go into further detail here.

Looking further...

- Multiple sequence alignment



- Individual variations within a gene

References

- Durban, Eddy e.a., *Biological Sequence Analysis*
- Altschul e.a., *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*