# Alignment Scores für Proteine

……a……  
……b……

Wahrscheinlichkeit, dass aus Aminosäure a Aminosäure b wird:

P(a⬄b)=fa\*fb

fa=Anzahl der AS a/Anzahl aller Aminosäuren in der Sequenz (Bei zufälliger Entwicklung!)

qab=Aminosäurenab/Summe aller Aminosäuren

odds ratio r=qab/fa\*fb (Beobachtete Veränderung vs erwartete zufällige Veränderung)

odds ratio >1 -> positiv selektiert, häufiger als Zufalls

=1 -> zufällig

<1 -> negativ selektiert, seltener als Zufall

Bei mehreren Aminosäuren in einer Sequenz kann alles aufmultipliziert werden

Sab = log(qab/(fa\*fb)) S=Score Substitution Matrix Element

SAB=Summe aller Sab in der Sequenz -> Summer aller Substitution Scores der Aminosäuren darin

SAB… Alignment Score der beiden Sequenzen

Odds Ratio aus Score: o=2^(S/2) (halfbit/logit)

Aa,b=Anzahl der Substitutionen a->b

mb…relative Mutierbarkeit einer Aminosäure b

mb=Summe über alle anderen AS a (Aa,b /fb) -> Normiert auf mAlanin

Ma,b=Λ\*mb\*Aa,b/Summe über alle anderen AS a (Aa,b)

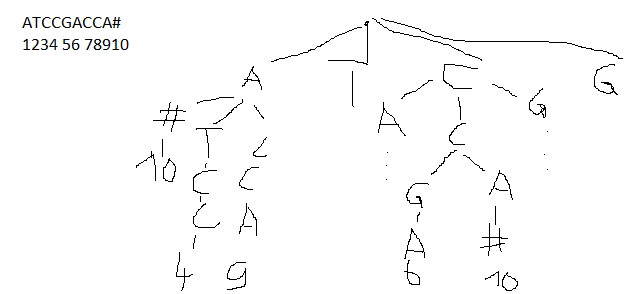
->Ma,b Eintrag in die PAM 1 an der Stelle a,b

PAM2ab = (PAM1ab)2

PAM1-Wert: Wahrscheinlichkeit, dass diese Substitution in einer 100 BP-Sequenz einmal auftritt.

## FASTA, BLAST, PSI-BLAST

Indexbäume: Verfahren, mit dem effektiv nach Positionen von Substrings gesucht werden kann.



Hashing: Mathematische Funktion, die Zeichenketten Zahlenwerten zuweist.

Für 4 Zeichen (ATCG): H(a0,a1,…,an)=a0\*4^0+a1\*4^1+….+an\*4^5

### FASTA

k-Tupel-Suche von identischen Teilsequenzen.

K=2 Proteine -> 20^2=400 Mögliche Tupel

Übergabe der Tupel an die Datenbank

Vergleich mit jeder Sequenz in der Datenbank:

1. Vergleich der Query-Sequenz mit Candidate-Sequenz bezüglich der Tupel (bsp. Als Dotplot). Sequenzen mit weniger als … Hits fliegen raus.
2. Bewertung der Hits (perfekte Matches) mit Austauschmatrix, Versuch die Lücken zu überbrücken (entlang von Diagonalen, keine Gaps, nur Missmatches)
3. Zusammenfassung der Hits mit den Missmatches dazwischen, Bewertung mit Gesamtscore -> Auswahl von 10 besten Diagonal Runs
4. Verbindung der Diagonalen mit Gaps, Lückenkosten abhängig von der Lückenlänge. Muss nicht immer perfekt Anfang und Ende der Runs verbinden.
5. Suche nach dem Weg mit dem höchsten Score, anhand der Run-Scores (oder dem Score der verbundenen Teilstücke) und den gap costs dazwischen
6. Entlang des ermittelten Pfades noch mal den „schmalen Bereich“ eingrenzen und den Pfad mit „dynamischer Programmierung XD“ nach dem optimalen Alignment durchsuchen

### BLAST

3-Tupel-Suche

Proteine: 20^3 = 8000 Möglichkeiten

Query: LNKCKTPQGQ

1. Aufteilung in 3er-Paare und Aufstellen von Ähnlichen Wörtern, Bewertung nach Substitutionsmatrix

|  |  |
| --- | --- |
| Score | Wort (Original: PQG) |
| 18 | PQG |
| 15 | PEG |
| 14 | PRG |
| 14 | PKG |
| 13 | PDG |
| 13 | PMG |
| 12 | PQA |
| 12 | PQG |

1. Cutoff der Liste nach heuristischem Grenzwert
2. Übergabe des Tupels und der ähnlichsten Wörter in die Datenbank
3. Weiteres Vorgehen nach FASTA

* Beeinflussbar mit der Wahl der Substitutionsmatrix und dem Cutoff-Grenzwert
* Zusätzlich Rückgabe eines E-Werts, der die Signifikanz-Grenze darstellt, der zufällige von systematischen Treffern trennt. Je kleiner, desto besser. E>1 -> Übereinstimmung wahrscheinlich nur Zufall

### PSI-BLAST

Position-specific Iterated BLAST

Problem: BLOSSUM oder PAM sind u.U. zu allgemein für die untersuchte Peptidsequenz

1. Suche nach verwandten Sequenzen mit allgemeiner Matrix (BLOSSUM)
2. Aufbau einer spezifischen Austauschmatrix und Nutzung in 1)
3. Wiederholen, bis sich die erhaltenen Sequenzen nicht mehr ändern

Sequenz 1: AAKLQN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | A | K | L | Q | N |
| A |  |  |  |  |  |  |
| C |  |  |  |  |  |  |
| D |  |  |  |  |  |  |
| E |  |  |  |  |  |  |
| F |  |  |  |  |  |  |
| G |  |  |  |  |  |  |
| H |  |  |  |  |  |  |
| I |  |  |  |  |  |  |
| K |  |  |  |  |  |  |
| L |  |  |  |  |  |  |
| . |  |  |  |  |  |  |
| . |  |  |  |  |  |  |
| . |  |  |  |  |  |  |

# MSA – Multiples Sequenz Alignment

Anzahl Rechenschritte für Erstellung der Matrix analog zu Needleman-Wunsch:

lm\*(2m-1)

l… Länge der Sequenzen

m… Anzahl der Sequenzen

# UPGMA

Algorithmus:

* Finde die beiden (oder mehr, wenn gleicher Abstand) nächsten Knoten i und j -> di,j ist geringster Abstand
* Fasse i und j zu neuem Knoten x zusammen
* dix=djx=dij/2
* Neue Abstände berechnen zu anderen Knoten (k) dxk=(dik+djk)/2

# Weblogos (aus MSA)

Informationsgehalt Sobs=-(pA\*log2pA+ pA\*log2pA+…)

Auftragen der Informationsgehalte der einzelnen Stellen, aufgeteilt nach den Inf.-Gehältern der einzelnen Zustände. Aufgetragen: R=Smax-Sobs, wobei Smax der Fall mit idealer Gleichverteilung der Wahrscheinlichkeiten ist. -> Höhe der Balken bedeutet, wie sicher man sich sein kann, dass der Buchstabe erscheint!