V2 Paarweises Sequenzalignment

- Methoden des Sequenzalignments
- Austauschmatrizen
- Bedeutsamkeit von Alignments
- BLAST, Algorithmus – Parameter – Ausgabe http://www.ncbi.nih.gov



Diese Vorlesung lehnt sich eng an das BLAST Tutorial-Buch (links) an, Kapitel 3-9

Heute, in der 2. Vorlesung beschäftigen wir uns mit dem Alignment von 2 Proteinsequenzen.

Als ein "**Alignment**" bezeichnet man die relative Position der zweiten Sequenz bzgl. der ersten Sequenz.

Evtl. ist die erste Sequenz nach vorne etwas verlängert (z.B. um 10 Positionen) und das beste Alignment für die zweite Sequenz beginnt erst bei der Position 11.

Die beiden Sequenzen müssen ausserdem nicht über die gesamte Länge parallel zueinander laufen. Eine kann zwischendurch **Insertionen** haben, was z.B. einem verlängerten Loop auf der Proteinoberfläche entsprechen könnte.

Wir erwarten daher von einem Alignment-Algorithmus, dass wir mit damit ein optimales oder ein sehr gutes Alignment erzeugen können.

Wir stellen heute 2 verschiedene Algorithmen vor, zum einen die sogenannte dynamische Programmierung, zum anderen den BLAST-Algorithmus.

Ausserdem benötigen wir eine Kostenfunktion bzw. Bewertungsfunktion, mit der wir die Güte verschiedener Alignments bewerten können.

Sequenz-Alignment

Wenn man 2 oder mehr Sequenzen vorliegen hat, möchte man zunächst einmal

- ihre Ähnlichkeiten quantitativ erfassen
  Die ähnlichen Regionen können hierbei die ganze Sequenz, oder Teile von ihr umfassen! Lokales Alignment ↔ globales Alignment

- Entsprechungen zwischen **einzelnen Bausteinen** beider Sequenzen erfassen

- Gesetzmässigkeiten der **Konservierung** und **Variabilität** beobachten

- Rückschlüsse auf entwicklungsgeschichtliche **Verwandschaftsverhältnisse** ziehen

Wichtiges Ziel: **Annotation**, d.h. Zuordnung von strukturellen und funktionellen Merkmalen zu Gensequenzen.

Wenn man eine (neue) Proteinsequenz oder Nukleotidsequenz vorliegen hat, ist einer der ersten Schritte die Suche nach ähnlichen Sequenzen in einer der grossen Datenbank mit bekannten Sequenzen.

Es könnte sogar sein, dass man die identische Sequenz in der Datenbank findet, da schon früher einmal jemand dasselbe Protein sequenziert hat.

Andernfalls kann man ähnliche Sequenzen finden, die vermutlich auch funktionelle oder andere Annotationen besitzen. Man kann so auch die Zugehörigkeit zu Proteinfamilien ableiten.

Falls die neue Sequenz Ähnlichkeiten zu Sequenzen in anderen Spezies aufweist, können dies homologe Sequenzen zur Eingabesequenz sein, d.h. von einem gemeinsamen Vorläufer abstammen.

Wenn die Funktion der ähnlichen bzw. homologen Sequenzen bekannt ist, kann man diese vermutlich (in einem gewissen Rahmen) auf die Eingabesequenz übertragen.

Es ist übrigens stets vorteilhaft, Proteinsequenzen miteinander zu vergleichen anstelle von Nukleotidsequenzen, da das Aminosäurealphabet 20 Buchstaben enthält und der DNA-Code nur 4 Buchstaben. Deshalb sind Vergleiche von Proteinsequenzen viel sensitiver.

Ähnlichkeit von Aminosäuren



Margaret Dayhoff
http://www.nlm.nih.gov/changingthefaceofmedicine/gallery/photo_76_7.html

Margaret Dayhoff stellte die Ähnlichkeit (beobachtete Austauschhäufigkeiten zwischen verwandten Sequenzen) zwischen Aminosäuren als $\log_2$ odds Verhältnis, oder *lod score* dar.

Lod score einer Aminosäure: nehme den Logarithmus zur Basis 2 ($\log_2$) von dem Verhältnis der beobachteten Häufigkeit für ein Paar durch die zufällig für das Paar erwartete Häufigkeit.

Lod score = 0 → beobachtete und erwartete Häufigkeiten sind gleich
> 0 → ein Austauschpaar tritt häufiger auf als zufällig erwartet
< 0 → unwahrscheinlicher Austausch

Allgemeine Formel für die Bewertung s_{ij} zweier Aminosäuren *i* und *j*.

$$s_{ij} = \log \frac{q_{ij}}{p_i p_j}$$

mit den individuellen Häufigkeiten p_i und p_j, und der Paarungsfrequenz q_{ij},

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 3

Wir behandeln zuerst die **Bewertungsfunktion** für Alignments.

Dazu verwendet man sogenannte **Austauschmatrizen** für Aminosäuren. So eine Matrix enthält die „Kosten“, die für den Austausch einer bestimmten Aminosäure gegen eine andere Aminosäure anfallen. Wir nehmen zunächst einmal eine Situation an, bei der wir zwei homologe Sequenzen miteinander alignieren möchten, die von einem gemeinsamen Vorläufer abstammen. Die beiden Sequenzen unterscheiden sich mittlerweile in mehreren Positionen, an denen während der Evolution Mutationen auftraten.

Um die Kosten für einen dieser Austausche X-> Y abzuschätzen, betrachtet man die Häufigkeit, mit der solche Mutationen von Aminosäure X nach Aminosäure Y in allen Paaren von verwandten Sequenzen auftreten.

Eine der ersten, die solche Statistiken aufstellte, war Margaret Dayhoff (1925-1983). Sie erwarb einen Doktortitel in Quantenchemie und wurde später Professorin an der Georgetown University. Auf Wikipedia steht „Seit 1955 konnte sie mit einem Computersystem arbeiten und entwickelte Programme, welche die Aminosäuresequenzen homologer Proteine verschiedener Spezies verglichen und damit die Grundlage der Sequenzalignierung schufen. Seit 1965 erschien der „Atlas of Protein Sequence and Structure“, ein Sammelwerk aller bis dahin bekannten Proteinsequenzen. Die Daten wurden ab 1984 in die Protein Information Resource (PIR)-Datenbank übernommen, die 2002 in die UniProt-Datenbank mündete. Ab 1966 entwickelte Margaret Dayhoff das **PAM-Modell**, das die Wahrscheinlichkeit einer Veränderung einer Proteinsequenz zu bestimmen versucht.“

Ähnlichkeit der Aminosäuren

Beispiel: die relative Häufigkeiten von Methionin und Leucin seien 0.01 und 0.1.

Durch zufällige Paarung erwartet man 1/1000 Austauschpaare Met – Leu.

Wenn die beobachtete Paarungshäufigkeit 1/500 ist, ist das Verhältnis der Häufigkeiten 2/1.
Im Logarithmus zur Basis 2 ergibt sich ein *lod score* von +1 or 1 bit.

Wenn die Häufigkeit von Arginin 0.1 und die Paarung mit Leu die Häufigkeit 1/500 hat, dann ergibt sich ein *lod score* für ein Arg – Leu Paar von -2.322 bits.

Gewöhnlich berechnet man *nats*, multipliziert die Werte mit einem Skalierungsfaktur und rundet sie dann auf Integer Werte
→ **Austauschmatrizen** PAM und BLOSUM.

Diese ganzzahligen Werte (Integers) nennt man *raw scores*.

Hier betrachten wir zwei Beispiele, um die auf der vorigen Folie vorgestellte Formel zu verstehen. Dort setzte man die Anzahl an beobachteten Austauschen ins Verhältnis zur Häufigkeit beider Aminosäuren. Bei einer gleichmässigen Verteilung aller 20 Aminosäuren, hätte jede Aminosäure eine Häufigkeit von 1/20-tel. Wenn eine Aminosäure stattdessen eine Häufigkeit von 1/10-tel hat, also jede 10. Position diese Aminosäure enthält, dann ist sie zweifach angereichert.

Im ersten Beispiel ist dies z.B. für Leucin der Fall (dies ist nur ein Beispiel, die tatsächliche Häufigkeit von Leucin ist ein anderer Wert). Im Gegensatz dazu nehmen wir an, dass Methionin sehr selten ist und nur eine Häufigkeit von 1/100 hat. Wenn wir nun eine grosse Anzahl an zufälligen paarweisen Sequenzalignments betrachten, sollten an jeder 1000-ten Position Leucin und Methionin als Paarung auftreten (also mit einer Häufigkeit von 1/1000-tel). Dies ist die erwartete Häufigkeit. Falls man in den natürlich auftretenden Sequenzen nun an jeder 500-ten Position ein Leu/Met-Paar findet (also mit einer Häufigkeit von 1/500), dann ist die Paarung Leu/Met 2-fach angereichert. Zur Basis 2 ist das dann eine Bewertung von +1. Dies wäre vermutlich der Fall, wenn Leu und Met physikochemisch ähnlich zueinander wären, also häufig ineinander ausgetauscht würden (was nicht unbedingt der Fall ist). Wir wissen normalerweise nicht, in welche Richtung die Mutation aufgetreten ist, also von Leu nach Met, oder von Met nach Leu. Daher bewerten wir beide Austauschrichtungen mit derselben Häufigkeit.

Bewertungs- oder Austausch-Matrizen

– dienen um die Qualität eines Alignments zu bewerten

– Für Protein/Protein Vergleiche:
eine *20 x 20* Matrix für die Wahrscheinlichkeit, mit der eine bestimmte Aminosäure gegen eine andere durch zufällige Mutationen ausgetauscht werden kann.

– Matrizen werden als symmetrisch angenommen, besitzen also die Form einer Dreiecksmatrix.

Diese Statistik stellt man für alle möglichen 400 Paare zwischen den 20 natürlich auftretenden Aminosäuren auf.

Also auch für den Austausch einer Aminosäure gegen sich selbst. Dies ist die Häufigkeit, dass Aminosäuren in paarweisen Alignments konserviert sind. Diese Zahlen liegen dann auf der Diagonale der Matrix.

Da wir im Allgemeinen nicht bewerten können, in welche Richtung der Austausch erfolgte (siehe vorige Folie), werden die Matrizen als **symmetrisch** angenommen.

Substitutions-Matrizen

(1) Mutationen bevorzugen bestimmte Austausche

– Einige Aminosäuren besitzen ähnliche Codons (siehe Codon-Sonne)
– Diese werden eher durch Mutation der DNA ineinander mutiert

Beispiel

– TTT & TTC kodieren für Phe, TTA & TTG kodieren für Leu
– Durch Austausch der letzten Base können diese beiden AS gegeneinander ausgetauscht werden.

(2) Selektion bevorzugt bestimmte Austausche

– Da einige Aminosäuren ähnliche Eigenschaften und Struktur haben, werden sie leichter gegeneinander ausgetauscht als andere. Der Austausch von Aminosäuren ähnlichen Charakters (Ile, Leu) ist wahrscheinlicher (hat eine höhere Bewertung bzw. tritt häufiger in der Natur auf) als der von Aminosäuren unterschiedlichen Charakters (e.g. Ile, Asp).
– Austausche, die die Funktion des Proteins gut erhalten, bleiben vermutlich länger erhalten als andere.

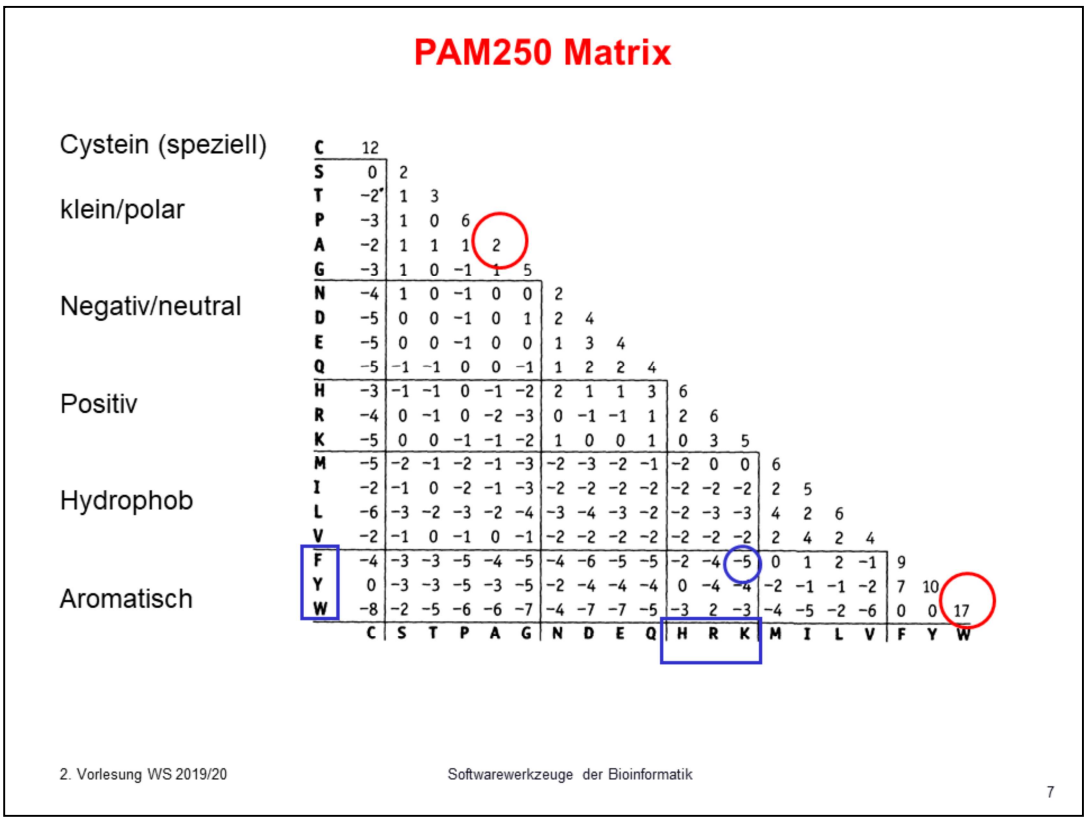
Die Häufigkeit von AS-Austauschen wird durch 2 Prinzipien bestimmt.

Zum einen ist wichtig, wie kompliziert der Austausch ist. Die Änderung findet ja tatsächlich auf Ebene der genomischen Sequenz statt (DNA).

Jede AS wird durch ein Basentriplett (Codon) kodiert. Manche AS unterscheiden sich nur durch eine Base, üblicherweise die letzte, z.B. Phe und Leu.

Andere AS unterscheiden sich in allen 3 Positionen voneinander. Natürlich kann eine Zufallsmutation den ersten Austausch leichter erzeugen als den zweiten.

Auf der anderen Seite ist wichtig, welche Austausche durch die **Selektion** in der Natur toleriert werden, d.h. ob der Austausch die Proteinfunktion beeinträchtigt (dann hätte diese Spezies mit dieser Mutante eine geringere Fitness) oder nicht.



Dies ist die sogenannte PAM250 Matrix für AS-Austausche, die auf Margaret Dayhoff zurückgeht. Mit der Bedeutung des Kürzels 250 beschäftigen wir uns später.

- Auf der x-Achse und der y-Achse sind jeweils die 20 AS im 1-Buchstaben-Code aufgelistet. Die AS sind in Blöcke mit ähnlichen Eigenschaften zusammengefasst. F, Y und W sind die 3 aromatischen AS, M, I, L und V hydrophobe AS und so weiter.

- Da die Matrix als symmetrisch angenommen wird, ist nur eine Hälfte gezeigt (Dreiecksmatrix)
- Die **Diagonale** enthält die Häufigkeiten dafür, dass eine AS in Alignments konserviert ist. Die höchsten Werte haben Trp(W) und Cys(C) mit 17 und 12. Alanin (A) hat mit 2 eine sehr niedrige Bewertung.
- Die Diagonale gehört zum Block der Austausche gegen ähnliche AS. Diese Werte sind fast alle nicht-negativ (außer für Gly -> Pro), treten also häufiger auf, als zufällig erwartet.

Werte in den anderen Blöcken sind zumeist nicht-positiv, so dass diese Austausche weniger oft auftauchen als zufällig erwartet. Zum Beispiel ist Trp(W) die größte AS. Man kann sie kaum gegen eine andere AS austauschen, ohne entweder ein “Loch” im Protein zu erzeugen (wenn Trp im Proteininneren liegt) oder die Oberflächeneigenschaften deutlich zu verändern (wenn Trp an der Oberfläche liegt). Interessanterweise sind Austausche von negativen gegen positive AS nicht streng “verboten”. Geladene AS liegen oft entweder im aktiven Zentrum von Proteinen oder an der Proteinoberfläche. Im aktiven Zentrum wäre

solch ein Austausch vermutlich fatal. Das Enzym wäre dann defect. An der Oberfläche werden solche Austausche aber oft toleriert, da die AS dort einfach die Löslichkeit des Proteins in Wasser bestimmt. Fatal wäre der Austausch wiederum, falls die Position an einer Bindungsschnittstelle für DNA oder andere Proteine liegt.

Wichtig: dies ist lediglich eine gemittelte Statistik, die die jeweilige Position einer Aminosäure im Protein nicht berücksichtigen kann.

Beispiel für eine Bewertung

Wenn sich 2 Sequenzen in 2 (oder mehreren) Positionen unterscheiden, möchte man die Wahrscheinlichkeit berechnen, daß Änderung A an Position 1 auftritt UND Änderung B an Position 2 (usw).

Man braucht also log (A × B), wobei das Malzeichen für die UND-Verknüpfung steht.

Es gilt allgemein log (A × B) = log A + log B

→ die Bewertung (Score) eines Alignments ist daher einfach die **Summe** aller Bewertungen für die Paare an Aminosäuren (Nukleinsäuren) des Alignments:

Sequenz 1: TCCPSIVARSN
Sequenz 2: SCCPSISARNT
1 12 12 6 2 5 -1 2 6 1 0 → Alignment Bewertung = 46

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 8

Proteine unterscheiden sich natürlich im Allgemeinen nicht nur an einer Position.

Wir müssen die Häufigkeit berechnen können, mit der an mehreren Positionen bestimmte Mutationen (Austausche) stattfinden.

Hierbei können wir davon profitieren, dass die Werte in der Austauschmatrix im Allgemeinen logarithmierte Häufigkeiten sind.

Die UND-Verknüpfung für das gemeinsame Auftreten mehrerer Positionen löst sich dann in eine Addition der einzelnen Bewertungen auf.

Dabei nehmen wir an, dass aufeinanderfolgende Positionen unabhängig voneinander sind (in der Praxis hängen sie natürlich etwas voneinander ab).

Dayhoff Matrix (1)

– wurde von Margaret.O. Dayhoff aufgestellt, die statistische Daten über die Austauschhäufigkeit von Aminosäuren in paarweisen Sequenzalignments sammelte

– Datensatz enthält eng verwandte Paare von Proteinsequenzen (> 85% Identität). Diese können nämlich zweifelsfrei aligniert werden.

– Aus der Frequenz, mit der Austausche auftreten, stellte sie die 20 x 20 Matrix für die Wahrscheinlichkeiten auf, mit der Mutationen eintreten.

– Diese Matrix heisst PAM 1. Ein evolutionärer Abstand von 1 PAM (point accepted mutation) bedeutet, dass es 1 Punktmutation pro 100 Residuen gibt, bzw. dass die beiden Sequenzen zu 99% identisch sind.

Die von Margaret Dayhoff entwickelten Dayhoff-Matrizen bzw. PAM-Matrizen sind vermutlich das am weitesten verbreitete Bewertungsschema.

Hier ist ein Link auf ein PDF, das die Original-Publikation von 1978 enthält:

https://chagall.med.cornell.edu/BioinfoCourse/PDFs/Lecture2/Dayhoff1978.pdf

Es gibt 400 / 2 = 200 Paare von Aminosäuren. Der Datensatz, den Dayhoff damals zur Verfügung hatte, enthielt 1572 Mutationen, also etwa 8 Datenpunkte pro Aminosäurepaar. Daraus eine vernünftige Statistik zu erhalten, ist wirklich bemerkenswert. Die PAM250-Matrix auf Folie 7 entspricht der Version in der damaligen Publikation. Neuere Versionen weichen nur geringfügig davon ab.

Dayhoff et al. untersuchten Alignments eng verwandter Sequenzen, so dass es unwahrscheinlich ist, dass sich die Häufigkeit einer bestimmten Mutation (z.B. A -> D) als Resultat einer Reihe von aufeinanderfolgenden Mutationen (z.B. A -> x -> y -> D) ergibt. Da nur wenige Proteinfamilien betrachtet wurden (72), enthält die Matrix der “accepted point mutations” viele Einträge mit 0 oder 1.

Ein evolutionärer Abstand von 1 PAM bedeutet, dass im Mittel 1 von 100 Positionen in einer Sequenz mutiert ist. Als Faustregel kann man verwenden, PAM 1 = 1 Millionen Jahre.

Allerdings gibt es keinen allgemeinen Zusammenhang zwischen PAM-Abstand und Evolutionszeit, da verschiedene Proteinfamilien mit unterschiedlichen Raten mutieren.

Dayhoff Matrix (1)

Für $i \neq j$ definieren wir $p_{ij} = c\frac{A_{ij}}{\sum_k A_{ik}}$

wobei A_{ij} die beobachtete Anzahl an Austauschen ist, und
c eine positive Konstante ist, deren Wert wir noch bestimmen müssen.

Die Diagonalelemente setzen wir auf: $p_{ii} = 1 - \sum_{j \neq i} p_{ij}$, wobei $\sum_j p_{ij} = 1$ gilt.

f_i sei die Frequenz der i-ten Aminosäure im Datensatz. Der Bruchteil an Aminosäuren, die während 1 PAM mutieren beträgt:

$$\sum_i f_i \sum_{j \neq i} p_{ij} = c \sum_i \sum_{j \neq i} f_i \left(A_{ij} / \sum_k A_{ik} \right)$$

Für PAM1 erwarten wir, dass 1% aller Aminosäuren mutieren. Damit ergibt sich:

$$c = \frac{0.01}{\sum_i \sum_{j \neq i} f_i \left(A_{ij} / \sum_k A_{ik} \right)}$$

Diese Konstante kann man oben einsetzen und erhält damit die Einträge der PAM1-Matrix.

Diese Folie erklärt, wie die Einträge in der PAM-Matrix aus der Anzahl der beobachteten Mutationsereignisse A -> B berechnet werden.

Dayhoff Matrix (2)

Aus PAM 1 kann man Matrizen für größere evolutionäre Entfernungen herstellen, indem man die Matrix **mehrfach mit sich selbst multipliziert.**

PAM250:

- 2,5 Mutationen pro Residue
- entspricht 20% Treffern zwischen zwei Sequenzen, d.h. man beobachtet Änderungen in 80% der Aminosäurepositionen.
- Dies ist die Default-Matrix in vielen Sequenzanalysepaketen.

Für den Vergleich zweier Sequenzen sollte man eine Bewertungsmatrix verwenden, deren tatsächlicher evolutionärer Abstand (Grad an Sequenzähnlichkeit) dem zwischen den beiden Sequenzen entspricht.

z.B. bedeutet PAM250 etwa 250 Mutationen pro 100 Positionen bzw. 2,5 Mutationen pro Residue. Effektiv sind jedoch noch etwa 20% der beiden Sequenzen identisch. Wie kann das sein?

PAM2 erhält man als Matrixprodukt von PAM1 mit sich selbst. In analoger Weise erhält man weiter entfernte PAM-Versionen. Der Hintergrund ist, dass man die Matrix als Übergangswahrscheinlichkeiten in einem Markov-Prozess interpretieren kann. 250 Mutationen pro 100 Positionen enthält man dann eben durch 250-fache Ausführung von einzelnen Mutationen, deren Häufigkeit durch die PAM1-Matrix beschrieben werden.

BLOSUM Matrix

Einschränkung der Dayhoff-Matrix:
Die Matrizen, die auf dem Dayhoff-Modell der evolutionären Raten basieren, sind von eingeschränktem Wert, da ihre Substitionsraten von Sequenzalignments abgeleitet wurden, die zu über 85% identisch sind.

S. Henikoff und J.G. Henikoff: verwendeten später lokale Multiple Alignments von entfernter verwandten Sequenzen ➔ **Blosum-Matrix**

Dies war möglich, da es nun bereits mehr Sequenzen sowie Algorithmen für multiple Alignments gab.



Steven Henikoff



Jorja G. Henikoff

Vorteile dieses Ansatzes:
- größere Datenmengen (es gibt mehr Sequenzen, die entfernt miteinander verwandt sind als nah verwandte)
- multiple Alignments sind robuster als paarweise Alignments

Die Dayhoffschen PAM-Matrizen stammen gewissermassen aus der Vorzeit der Sequenzanalyse.

1992, als von Jorja und Steven Henikoff die BLOSUM-Matrizen entwickelt wurden, existierten etwa 2000 Blöcke von Proteinfamilien (siehe folgende Folien).

Dies ist die Originalpublikation:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC50453/pdf/pnas01096-0363.pdf

BLOSUM Matrix (2)

Die BLOSUM Matrizen (BLOcks SUbstitution Matrix) basieren auf der BLOCKS Datenbank.

Die BLOCKS Datenbank verwendet das Konzept von Blöcken (lückenlose Aminosäure-Signaturen), die charakteristisch für eine Proteinfamilie sind.

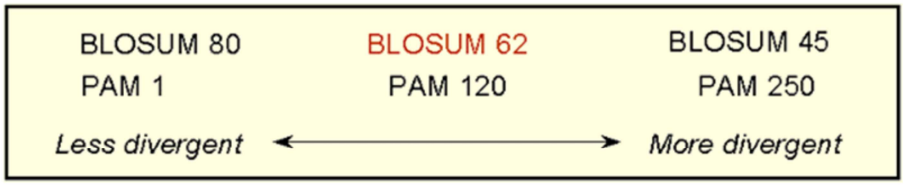
Aus den beobachteten Mutationen innerhalb dieser Blöcke wurden Austauschwahrscheinlichkeiten für alle Aminosäurepaare berechnet und als Einträge für eine *log odds* BLOSUM matrix benutzt.

Man erhält unterschiedliche Matrizen indem man die untere Schranke des verlangten Grads an Identität variiert.
z.B. wurde die BLOSUM80 Matrix aus Blöcken mit > 80% Identität abgeleitet.

Keine Kommentare.

Welche Matrix soll man benutzen?

Enge Verwandtschaft (Niedrige PAM, hohe Blosum)
Entfernte Verwandtschaft (Hohe PAM, niedrige Blosum)



Vernünftige Default-Werte: PAM250, BLOSUM62

Bei einem Level von 2000 PAM sind laut Schwartz und Dayhoff alle Unterschiede zwischen verschiedenen Aminosäurepaaren “glatt gebügelt”. Lediglich das Matrixelement für Cys/Cys ist 10% größer als zufällig erwartet.

Bei einem Abstand von 250 PAM bleiben im Mittel 20% der Aminosäuren unverändert. Allerdings ist die Mutabilität der Aminosäuren sehr unterschiedlich.

48% der Tryptophane, 41% der Cysteine und 20% der Histidin-Residuen wären noch unverändert, jedoch nur etwa 7% der Serine.

Bei den Blosum-Matrizen gelten hohe Werte für Sequenzen, die sich wenig voneinander entscheiden, bei den PAM-Matrizen gelten niedrige Werte für sehr ähnliche Sequenzen.

Gewichtung von Lücken (Gaps)

Neben der Substitutionsmatrix braucht man auch eine Methode zur Bewertung von Lücken.

Welche Bedeutung haben Insertionen und Deletionen im Verhältnis zu Substitutionen?

Lineares Modell: gleiche Kosten für das Öffnen und Verlängern von Gaps

$W_k = k\, W_1$ $\quad$ W_1 = Kosten für einen Gap

Affines Modell: unterschiedliche Kosten fürs Öffnen und Verlängern

$W_k = u\, k + v$ $\quad$ v = gap opening penalty
u = gap extension penalty

Macht das einen Unterschied?

Sequenzalignments enthalten oft auch Lücken (Gaps), die man ebenfalls bewerten muss. Generell sind Lücken ungünstig. Es ist evolutionär einfacher, eine Aminosäure gegen eine andere austauschen, als eine einzufügen bzw. zu entfernen.

Ein kurioser Fall wäre jedoch, wenn in einer Region mehrfach sehr kurze Gaps (z.B. 1-2 AS) auftreten würden, die jedoch nicht direkt hintereinander liegen.

Intuitiv würde man annehmen, dass alle kurzen Lücken in einer benachbarten Region am besten direkt hintereinander liegen sollten. Im Innern von Proteinen sind Gaps nämlich eher selten. Am ehesten treten Insertionen/Deletionen in Regionen (Loops) auf der Proteinoberfläche auf. Man bewertet daher im “affinen Modell” den Einbau von Gaps mit einem relativ hohen Kostenterm. Diesen Gap dann aber zu verlängern, “kostet” deutlich weniger als der Einbau eines Gaps (das Unterbrechen des Alignments).

Gap Penalties

Beispiel: aligniere die beiden Sequenzen TACGGGCCCGCTAC und TAGCCCTATCGGTCA.

Mit einer linearen gap penalty Funktion ist das Ergebnis (Alignment mit EMBOSS Water, Austauschmatrix DNAfull, gap opening und extension Kosten beide 1.0):

```
TACGGGCCCGCTA-C
||   | || ||| |
TA---G-CC-CTATC
```

Mit einer affinen gap penalty, lautet das Ergebnis (Gap opening 5.0, gap opening1.0):

```
TACGGGCCCGCTA
||   |||  |||
TA---GCC--CTA
```

Affine gap penalty hilft dabei, kurze verstreute Gaps zu vermeiden.

Verschiedene Programme (CLUSTAL-W, BLAST, FASTA) empfehlen unterschiedliche Default-Werte, die man wohl erst einmal verwenden sollte.

https://en.wikipedia.org/wiki/Smith–Waterman_algorithm

Hier vergleichen wir das Alignment von zwei kurzen Sequenzen mit einem linearen Gap-Bestrafungsterm (oben) und mit einem affinen Bestrafungsmodell.

Im unteren Modell erhalten wir nur 2 Gaps, im oberen 4 Gaps (drei in der unteren Sequenzen, einen in der oberen Sequenz).

Das affine Modell scheint daher sinnvoller.

Diese Folie markiert das Ende des ersten Teils von Vorlesung #2. Im zweiten Teil beschäftigen wir uns mit 2 Algorithmen, mit denen man tatsächliche Alignments konstruieren kann.

Dynamische Programmierung: Needleman-Wunsch Algorithmus

- allgemeiner Algorithmus für Sequenzvergleiche
- maximiert eine Bewertung der Ähnlichkeit
- bester Match = größte Anzahl an Residuen einer Sequenz, die zu denen einer anderen Sequenz passen, wobei Deletionen erlaubt sind.
-Der Algorithmus findet durch dynamische Programmierung das bestmögliche GLOBALE Alignment zweier beliebiger Sequenzen

- NW beinhaltet eine iterative Matrizendarstellung
  alle möglichen Residuenpaare (Basen oder Aminosäuren) – je eine von jeder Sequenz – werden in einem 2-dimensionalen Gitter dargestellt.
  alle möglichen Alignments entsprechen Pfaden in diesem Gitter.

- Der Algorithmus hat 3 Schritte: 1 Initialisierung 2 Auffüllen 3 Trace-back

Zunächst besprechen wir einen Algorithmus, der dynamische Programmierung verwendet.

Dynamische Programmierung bezeichnet eine bestimmte Herangehensweise an ein Optimierungsproblem. DP-Algorithmen bestimmen die Lösung eines Problems durch Aufteilung des Problems in Teilprobleme und systematische Speicherung von Zwischenresultaten. Dieser spezielle Algorithmus ist nach den beiden Entwicklern, Saul B. Needleman und Christian D. Wunsch, benannt und wurde 1970 publiziert.

Das Ziel ist, eine (mathematisch) optimale Abbildung zweier Sequenzen aufeinander zu finden.

Optimal bezieht sich auf eine bestimmte Bewertungsfunktion (Austauschmatrix für Aminosäuren plus Gap-Modell), die wir verwenden.

Wir tragen die beiden Sequenzen auf den Spalten und Reihen einer zweidimensionalen Matrix auf.

Links oben ist der Beginn beider Sequenzen, rechts unten deren Ende.

Eine Abbildung (alignment) beider Sequenzen aufeinander entspricht nun einem mehr oder weniger direkten Pfad von links oben nach rechts unten.

Wenn wir den Pfad entlang der Diagonalen verlängern, bilden wir die nächst folgenden Aminosäuren aufeinander ab.

Wenn der Pfad nach unten bzw. waagerecht zeigt, bauen wir in eine der beiden Sequenzen einen Gap ein.

Needleman-Wunsch Algorithm: Initialisierung

Aufgabe: aligniere die Wörter “COELACANTH” und “PELICAN” der Länge $m = 10$ und $n = 7$. Konstruiere $(m + 1) \times (n + 1)$ Matrix.

Ordne den Elementen der ersten Zeile und Reihe die Werte $-m \times gap$ und $-n \times gap$ zu.

Die Pointer dieser Felder zeigen zurück zum Ursprung.

		C	O	E	L	A	C	A	N	T	H
	0	-1 ←	-2 ←	-3 ←	-4 ←	-5 ←	-6 ←	-7 ←	-8 ←	-9 ←	-10 ←
P	↑-1										
E	↑-2										
L	↑-3										
I	↑-4										
C	↑-5										
A	↑-6										
N	↑-7										

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 18

In unserem Modellproblem möchten wir die beiden Worte COELACANTH und PELICAN miteinander alignieren.

Wir verwenden eine einfache Bewertungsfunktion, nach der eine Übereinstimmung (match) mit +1, eine Abweichung (mismatch) mit -1 und ein Gap mit -1 bewertet wird.

Im ersten Schritt des Algorithmus füllen wir die erste Zeile der Matrix und die erste Spalte mit von Null absteigenden ganzzahligen Werten auf. Diese „Randbedingungen“ erzeugen die Bewertung eventueller Gaps im Alignment.

Bei der Einfügung eines Gaps in dem Wort PELICAN nimmt das Alignment einen waagrechten Pfad. Wenn man z.B. unten rechts beginnt, würde H auf N abgebildet. Das passt nicht.

Stattdessen könnten wir in der senkrechten Sequenz am Schluss zwei Gaps einfügen, bis dann CAN auf CAN abgebildet wird.

Needleman-Wunsch Algorithm: Auffüllen

Fülle alle Matrizenfelder mit Werten und Zeigern mittels simpler Operationen, die die Werte der diagonalen, vertikalen, und horizontalen Nachbarzellen einschließen. Berechne

match score: Wert der Diagonalzelle links oben + Wert des Alignments (+1 oder -1)

horizontal gap score: Wert der linken Zelle + gap score (-1)

vertical gap score: Wert der oberen Zelle + gap score (-1).

Ordne der Zelle das Maximum dieser drei Werte zu. Der Pointer zeigt in Richtung des maximalen Werts.

		C	O	E	L	A	C	A	N	T	H
	0	-1 ←	-2 ←	-3 ←	-4 ←	-5 ←	-6 ←	-7 ←	-8 ←	-9 ←	-10 ←
P	↑ -1	↖ -1	↖ -2								

max(-1, -2, -2) = -1

max(-2, -2, -3) = -2

(Lege Konvention fest, damit Pointer bei gleichen Werten immer in eine bestimmte Richtung zeigen soll, z.B. entlang der Diagonalen.

Im zweiten Schritt des Needleman-Wunsch-Algorithmus füllen wir die restlichen Werte der Matrix auf.

Nach dem DP-Prinzip ist die Konstruktion jedes innenliegenden Matrixelements ein Teilproblem, dessen Lösung nur von seinen unmittelbaren Nachbarn, und zwar nur den 3 Nachbarzellen oben, links und diagonal links-oben abhängt.

Falls der beste Alignmentpfad nach links zeigen würde, wäre die Einfügung eines Gaps in der senkrechten Sequenz die beste Lösung.

Falls der beste Alignmentpfad nach oben zeigen würde, wäre die Einfügung eines Gaps in der waagrechten Sequenz die beste Lösung.

Ansonsten zeigt der beste Alignmentpfad nach schräg links oben – das Alignment wird also ohne Gap fortgesetzt.

Was ist nun die beste Lösung?

Dazu berechnen wir die drei rot, grün und blau gefärbten Kostenterme. Diese entsprechen den Werten der drei Nachbarzellen, wobei entweder Gapkosten (links und oben) hinzukommen, bzw. die Bewertung von match/mismatch der beiden Buchstaben, die aufeinander abgebildet werden.

In jedem Kästchen/Matrixelement wird ein **Pointer** gesetzt, der in Richtung der besten Lösung zeigt.

Alle Felder der Matrix werden nun reihenweise von links nach rechts und von oben nach unten aufgefüllt. Die Komplexität des Algorithmus ist daher n (Länge Sequenz 1) mal m (Länge Sequenz 2).

Needleman-Wunsch Algorithmus: Trace-back

Trace-back ergibt das Alignment aus der Matrix.

Starte in Ecke rechts unten und folge den Pfeilen bis in die Ecke links oben.

```
COELACANTH
-PELICAN--
```

		C	O	E	L	A	C	A	N	T	H
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
P	-1	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
E	-2	-2	-2	-1	-2	-3	-4	-5	-6	-7	-8
L	-3	-3	-3	-2	0	-1	-2	-3	-4	-5	-6
I	-4	-4	-4	-3	-1	-1	-2	-3	-4	-5	-6
C	-5	-3	-4	-4	-2	-2	0	-1	-2	-3	-4
A	-6	-4	-4	-5	-3	-1	-1	1	0	-1	-2
N	-7	-5	-5	-5	-4	-2	-2	0	2	1	0

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 20

Der dritte Schritt des Algorithmus, das **Traceback**, ist ganz einfach. Damit wird nun das beste Alignment aus den Matrixfelder konstruiert.

Man beginnt rechts unten und folgt den Pfeilen, bis der Algorithmus oben links ankommt.

Dadurch dass alle theoretisch möglichen Alignments betrachtet werden (jeder mögliche Pfad entspricht einem anderen Alignment), erhält man garantiert die optimale Lösung.

Smith-Waterman-Algorithmus

Smith-Waterman ist ein lokaler Alignment-Algorithmus. SW ist eine sehr einfache Modifikation von Needleman-Wunsch. Es gibt lediglich 3 Änderungen:

- die Matrixränder werden auf 0 statt auf ansteigende Gap-Penalties gesetzt.
- der maximale Wert sinkt nie unter 0. Pointer werden nur für Werte größer als 0 eingezeichnet.
- Trace-back beginnt am größten Wert der Matrix und endet bei dem Wert 0.

ELACAN

ELICAN

		C	O	E	L	A	C	A	N	T	H
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	1	0	0	0	0	0	0	0
L	0	0	0	0	2	1	0	0	0	0	0
I	0	0	0	0	1	1	0	0	0	0	0
C	0	1	0	0	0	0	2	0	0	0	0
A	0	0	0	0	0	1	0	3	2	1	0
N	0	0	0	0	0	0	0	1	4	3	2

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 21

Der Smith-Waterman-Algorithmus (Temple Smith und Michael S. Waterman 1981) ist eine Variante des Needleman-Wunsch-Algorithmus, die lokale Alignments erzeugt.

Um die Studierenden nicht zu verwirren, werden wir uns in dieser Vorlesung auf den Needleman-Wunsch-Algorithmus konzentrieren und die feinen Unterschiede nicht behandeln (-> Smith-Waterman ist nicht klausurrelevant).

In unserem Beispiel ist die Ausgabe dann das Alignment ELCAN/ELICAN.

BLAST – Basic Local Alignment Search Tool

- Findet das am besten bewertete lokale optimale Alignment einer Testsequenz mit allen Sequenzen einer Datenbank.
- Sehr schneller Algorithmus, 50 mal schneller als dynamische Programmierung.
- Kann verwendet werden um sehr große Datenbanken zu durchsuchen, da BLAST eine vor-indizierte Datenbank benutzt
- Ist ausreichend sensititv und selektiv für die meisten Zwecke
- Ist robust – man kann üblicherweise die Default-Parameter verwenden

Der zweite Algorithmus, den wir zur Berechnung von paarweise Alignments diskutieren, trägt den Namen BLAST. Die Originalpublikation ist:

https://pubmed.ncbi.nlm.nih.gov/2231712/

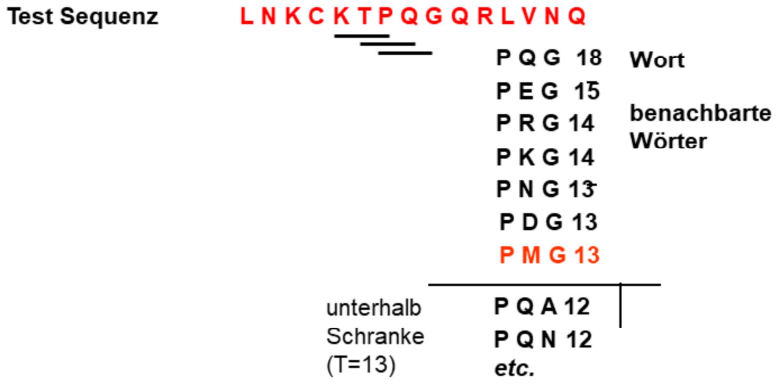
Diese Publikation von 1990 wurde in der Zwischenzeit über 86000 mal zitiert und zählt damit zu den 10-15 am häufigsten zitierten wissenschaftlichen Publikationen überhaupt.

Das Hauptmerkmal von BLAST ist seine überlegene Geschwindigkeit. Das Problem von dynamischer Programmierung ist nämlich die quadratische Komplexität m x n.

Das ist kein Problem, wenn man nur 2 Sequenzen alignieren möchte. Es wird jedoch ein sehr grosses Problem, wenn man überprüfen möchte, welche Sequenzen in einer grossen Sequenzdatenbank (am besten: alle bekannten Sequenzen) am ähnlichsten zu einer Eingabesequenz sind. Genau dies ist ja die Hauptanwendung von Sequenzalignments.

BLAST Algorithmus, Schritt 1

- Für ein gegebenes Wort der Länge *w* (gewöhnlich 3 für Proteine) und eine gegebene Bewertungs-Matrix
  erzeuge eine Liste aller Worte (*w*-mers), die eine Bewertung > *T* erhalten, wenn man sie mit dem *w*-mer der Eingabe vergleicht



Bei der DP hatte sich bewährt, das Problem in kleine Teilprobleme zu zerlegen.

BLAST verwendet zunächst dieselbe Idee. Es zerschnipselt die Eingabesequenz in kurze Worte (z.B. der Länge 3).

Der nächste Schritt unterscheidet sich von DP. Bereits jetzt berücksichtigt BLAST nämlich „ähnliche“ Worte derselben Länge.

Was ähnlich ist, wird mit einer Austauschmatrix bestimmt.

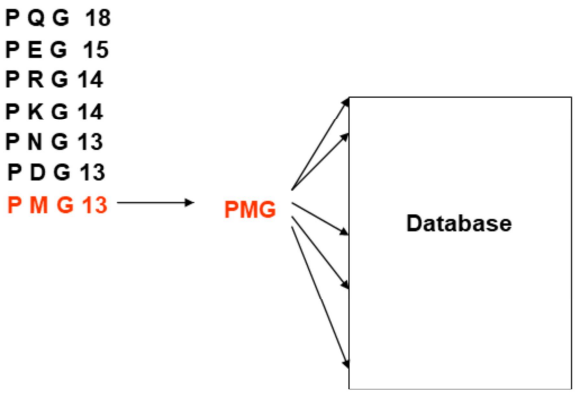
Für die Abbildung von PQG auf die identische Sequenz PQG ergibt die BLOSUM-Matrix: 7 (P/P) + 5 (Q/Q) + 6 (G/G) = 18

Das ähnlichste 3-Wort ist PEG: 7 (P/P) + 2 (E/Q) + 6 (G/G) = 15 usw.

BLAST berücksichtigt alle ähnlichen 3-Worte bis zu einer gewissen unteren Schranke.

BLAST Algorithmus, Schritt 2

jedes benachbarte Wort ergibt alle Positionen in der Datenbank, in denen es gefunden wird (hit list).

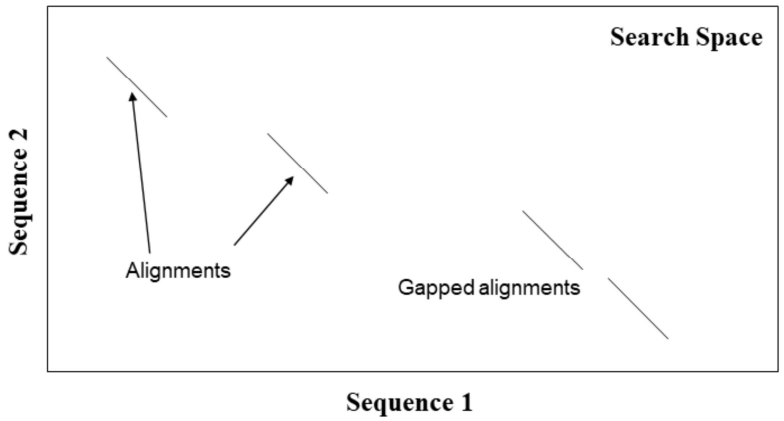


Wo findet man nun die 3-Worte in den Sequenzen der grossen Datenbank? Im Prinzip könnte man alle Sequenzen nun danach durchsuchen. Das würde jedoch lange dauern.

Der entscheidende Trick von BLAST, was ihm den Geschwindigkeitsvorteil bringt, ist eine Vor-Indizierung der Positionen aller 3-Worte in der Datenbank.

Sonst müssten diese ja bei jedem Alignment immer wieder neu berechnet werden. Stattdessen wird einmal ein grosser Gesamtindex angelegt. Danach ist das Auffinden der Treffer quasi schon erledigt.

Was ist gesucht?

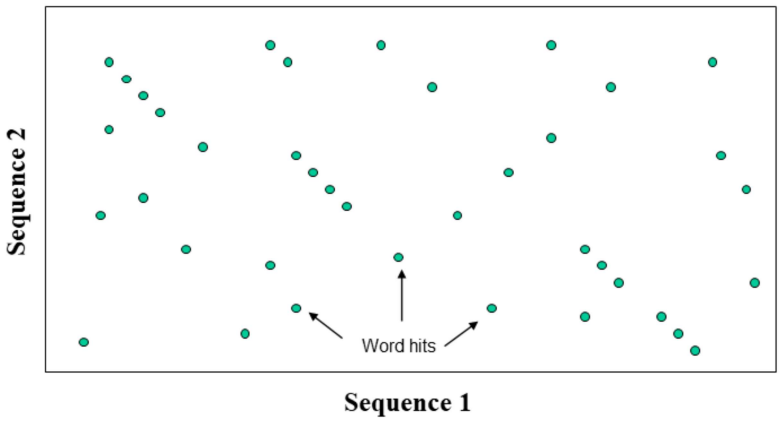


Das beste Mapping von Sequenz 1 auf Sequenz 2 entspricht einem unterbrochenen Pfad in dieser Diagonalmatrix.

Man kann sich das Alignment nun wieder genauso wie beim Needleman-Wunsch-Algorithmus als einen Pfad in der Matrix vorstellen, die von beiden Sequenzen aufgespannt wird.

Im Endeffekt erwartet man für das Alignment einen ununterbrochenen Pfad von rechts unten nach links oben.

Seeding

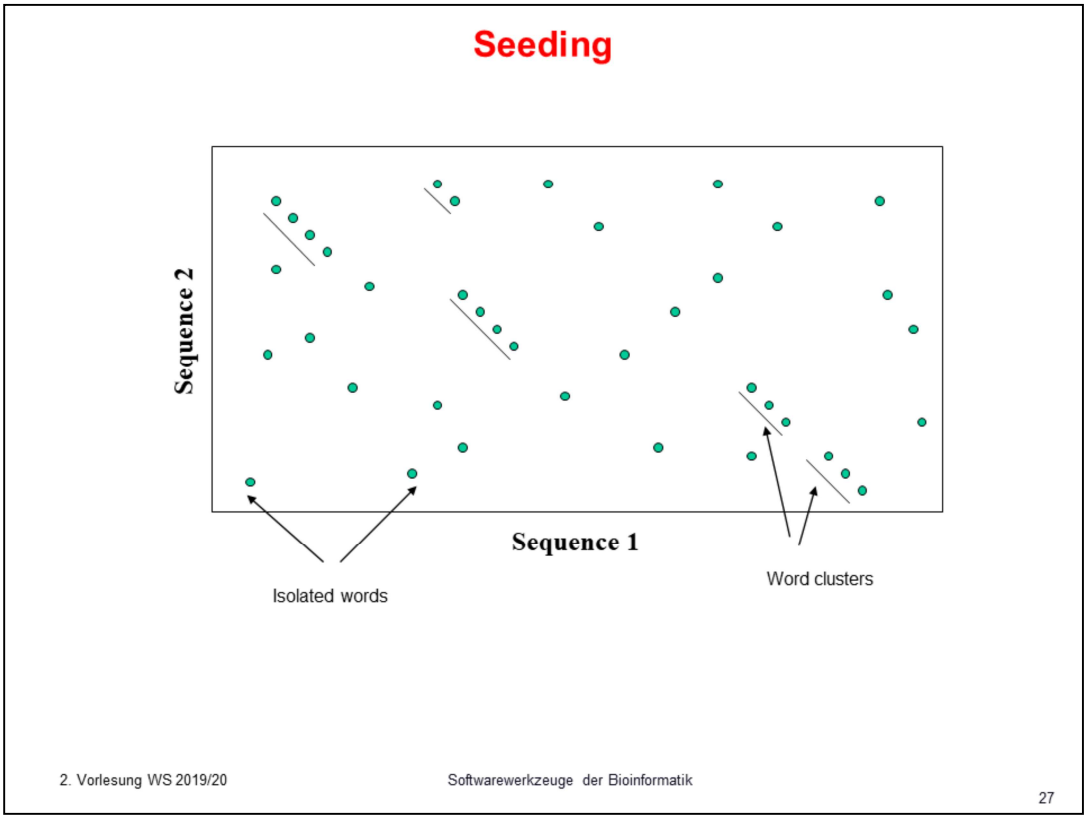


Können wir aus diesen „Word hits“ ein gutes Alignment konstruieren?

Als Ergebnis der 3-Wort-Suche von BLAST erhalten wir alle gezeigten “Treffer”. Jeder Punkt steht für ähnliche 3-Worte in den beiden Sequenzen.

Wenn wir dieses Bild mit dem auf der vorigen Folie vergleichen, können wir bereits erahnen, wo der optimale Alignment-Pfad liegen mag.

Er wird mehrere aufeinanderfolgende Treffer miteinander verbinden. Etliche Treffer rechts oben oder links unten werden nicht auf dem Pfad liegen. Treffer zwischen 3-Worten sind nämlich recht häufig und können auch zufällig auftreten.



Im nächsten Schritt verbindet BLAST benachbarte Treffer zu Wort-Clustern. Isolierte Wort-Treffer werden später nicht mehr betrachtet.

BLAST Algorithmus: Erweiterungsschritt

- das Programm versucht, den Seed in beide Richtungen auszudehnen indem solange Residuenpaare hinzugefügt werden bis die zusätzliche Bewertung kleiner als ein Schrankenwert ist.

- Nachdem die Ausdehnung beendet wurde, wird das Alignment so "zurückbeschnitten" dass es die maximale Bewertung erhält.



High-scoring Segment Pair (HSP)

Im letzten Schritt versucht der BLAST-Algorithmus nun, diese Wort-Cluster maximal in beide Richtungen zu verlängern. Dabei können auch Gaps eingefügt werden.

Man erhält als Ergebnis entweder ein Alignment über die gesamte Länge der beiden Sequenzen oder ein lokales Alignment der Abschnitte, die am besten passen.

Nachbarschaft für 3-Buchstaben-Worte

BLOSUM62		PAM200	
Wort	Bewertung	Wort	Bewertung
RGD	17	RGD	18
KGD	14	RGE	17
QGD	13	RGN	16
RGE	13	KGD	15
EGD	12	RGQ	15
HGD	12	KGE	14
NGD	12	HGD	13
RGN	12	KGN	13
AGD	11	RAD	13
MGD	11	RGA	13
RAD	11	RGG	13
RGQ	11	RGH	13
RGS	11	RGK	13
RND	11	RGS	13
RSD	11	RGT	13
SGD	11	RSD	13
TGD	11	WGD	13

Kommentar:
Sowohl die Auswahl der Austauschmatrix wie die Wahl des Cut-offs T wird den Seeding-Schritt beeinflussen.

2. Vorlesung WS 2019/20 Softwarewerkzeuge der Bioinformatik 29

Hier ist gezeigt, dass durch Auswahl einer unterschiedlichen Bewertungsmatrix leicht unterschiedliche 3-Worte berücksichtigt werden.

BLAST Eingabe

Notwendige Schritte um BLAST einzusetzen (im Zeitalter des Internets!):

Wähle einen **Webserver** (EBI = European Bioinformatics Institute, NCBI = National Center for Biotechnology Information …)

- gib Testsequenz ein (cut-and-paste)
- wähle die Nukleotid bzw. Aminosäure-Sequenzdatenbank, die durchsucht werden soll
- wähle Parameter um Output zu steuern (Zahl der Sequenzen …)
- wähle Parameter für das Alignment (z.B. Austauschmatrix, Filter,….)

Testsequenz =
MAFIWLLSCYALLGTTFGCGVNAIHPVLTGLSKIVNGEEAVPGTWPWQVTLQDRSGFHF
CGGSLISEDWVVTAAHCGVRTSEILIAGEFDQGSDEDNIQVLRIAKVFKQPKYSILTVNND
ITLLKLASPARYSQTISAVCLPSVDDDAGSLCATTGWGRTKYNANKSPDKLERAALPLLT
NAECKRSWGRRLTDVMICGAASGVSSCMGDSGGPLVCQKDGAYTLVAIVSWASDTCS
ASS GGVYAKVTKIIPWVQKILSSN

BLAST kann nun bequem in einen Webbrowser verwendet werden, z.B. über

https://blast.ncbi.nlm.nih.gov/Blast.cgi?ALIGNMENTS=50&ALIGNMENT_VIEW=Pairwise&AUTO_FORMAT=Semiauto&CLIENT=web&DATABASE=nr&DESCRIPTIONS=100&ENTREZ_QUERY=(none)&EXPECT=20000&FORMAT_BLOCK_ON_RESPAGE=None&FORMAT_ENTREZ_QUERY=(none)&FORMAT_OBJECT=Alignment&FORMAT_TYPE=HTML&GAPCOSTS=9+1&I_THRESH=0.005&LAYOUT=TwoWindows&MATRIX_NAME=PAM30&NCBI_GI=on&PAGE=Proteins&PROGRAM=blastp&QUERY=LSMDNNRNLDLDSII&SERVICE=plain&SET_DEFAULTS.x=14&SET_DEFAULTS.y=5&SHOW_LINKOUT=on&SHOW_OVERVIEW=on&WORD_SIZE=2&END_OF_HTTPGET=Yes

Sie werden BLAST in der Übung selbst ausprobieren. Die Frage, die wir beantworten möchten, lautet: welche Sequenzen in der Datenbank (oder Fragmente davon) sind am ähnlichsten zu meiner Eingabesequenz? Die Suche sollte schnell beendet sein und möglichst empfindlich sein. Leider sind dies jedoch gegensätzliche Anforderungen.

Etwa 25-30% aller Proteine enthalten Regionen “geringer Komplexität”, in denen entweder eine einzige oder ein paar Aminosäuren häufig wiederholt werden. Die Funktion dieser Regionen ist meist nicht verstanden. Manche dieser Sequenzmotive, z.B. GGGGG, PPPPP, TTTPTT, GGGGSGG und KKKKK werden mittlerweile mit der Tendenz zur Entfaltung bzw. Unordnung in Verbindung gebracht. Solche Regionen aufeinander abzubilden wäre für uns wenig hilfreich. Man verwendet daher Filter um solche Regionen im Alignment-

Schritt auszublenden. BLAST ersetzt solche Buchstaben durch X und ignoriert sie im Anschluss.

BLAST Ausgabe (1)

Please wait ...

```
BLASTP 2.2.2 [Dec-14-2001]


Reference:
Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schäffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= blast.seq  [Unknown form], 261 bases, 32E76D0B checksum.
         (261 letters)

Database: swissprot
           101,602 sequences; 37,315,215 total letters

Searching.................................................done
```


Dies ist die erste Seite der üblichen Ausgabe. Heutzutage ist die nr-Datenbank natürlich viel größer als hier gezeigt.

BLAST Ausgabe (2)

Kleine Wahrscheinlichkeit deutet an, dass der Treffer wohl nicht zufällig zustande kam.

Sequences producing significant alignments:	Score (bits)	E Value
swissprot:CTRB_HUMAN Chymotrypsinogen B precursor (EC 3.4.21.1).	433	e-121
swissprot:CTR2_CANFA Chymotrypsinogen 2 precursor (EC 3.4.21.1).	386	e-107
swissprot:CTRB_RAT Chymotrypsinogen B precursor (EC 3.4.21.1).	383	e-106
swissprot:CTRB_BOVIN Chymotrypsinogen B (EC 3.4.21.1).	348	4e-96
swissprot:CTRA_BOVIN Chymotrypsinogen A (EC 3.4.21.1).	330	1e-90
swissprot:CTRA_GADMO Chymotrypsin A precursor (EC 3.4.21.1).	286	2e-77

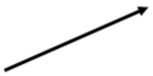
Dies sind die besten Treffer der Ausgabe. Meist ist die erste Sequenz sogar identisch mit der Eingabesequenz, sofern diese (als bekannte Sequenz) bereits in der Datenbank enthalten war.

Der letzte Wert gibt den E-value für die Wahrscheinlichkeit, dass ein Treffer dieser Bewertung (score in der vorletzten Spalte) oder mit einer besseren Bewertung in einer Datenbank dieser Größe zufällig zustande kommen kann.

Man sieht, dass alle dieser Treffer sehr signifikant sind und ähnlich gute Bewertungen haben. Die Proteinnamen aller Treffer sind eng verwandt. EC steht für die **Enzyme Classification Number**.

BLAST Ausgabe (3)

swissprot:CO2_HUMAN Complement C2 precursor (EC 3.4.21.43) (C3/C...	55	1e-07
swissprot:CO2_MOUSE Complement C2 precursor (EC 3.4.21.43) (C3/C...	53	3e-07
swissprot:ACH2_LONAC Achelase II protease (EC 3.4.21.-).	52	1e-06
swissprot:GD_DROME Serine protease gd precursor (EC 3.4.21.-) (G...	46	4e-05
swissprot:ACRO_CAPHI Acrosin (EC 3.4.21.10) (Fragment).	39	0.009
swissprot:CTRP_PENMO Chymotrypsin (EC 3.4.21.1) (Fragment).	36	0.047
swissprot:VSPA_CERCE Cerastotin (EC 3.4.21.-) (Fragments).	35	0.098
swissprot:EL2B_HORSE Neutrophil elastase 2B (EC 3.4.21.-) (Prote...	35	0.13
swissprot:CERC_SCHMA Cercarial protease precursor (EC 3.4.21.-) ...	34	0.26
swissprot:EL2A_HORSE Neutrophil elastase 2A (EC 3.4.21.-) (Prote...	33	0.42
swissprot:HPT_RABIT Haptoglobin beta chain (Fragment).	31	1.4
swissprot:NMT1_ASPPA NMT1 protein homolog.	30	4.8



Niedrige Bewertungen mit hohen Wahrscheinlickeiten deuten an, dass dies wohl keine guten Treffer sind.

Das ist das Ende der Ergebnisliste. Die Namen der Treffer sind (zumeist) sehr verschieden von „Chymotrypsinogen Precursor“.

Es gibt ein „Chymotrypsin“, das ist aber als Fragment gekennzeichnet, also vermutlich sehr kurz.

Die Bewertungen sind nun viel niedriger als vorher und der E-Value in der letzten Spalte nähert sich Eins.

Wir schauen uns auf der nächsten Folie an, wie der E-Value definiert ist.

Karlin-Altschul Statistik: E-value

Karlin und Altschul leiteten die Bewertung der Signifikanz eines Alignments ab (hier ohne Herleitung):

$$E = kmne^{-\lambda S}$$

Die Anzahl an Alignments (*E*), die man während einer Suche in einer Sequenzdatenbank mit *n* Sequenzen mit einer *m* Buchstaben langen Suchsequenz zufällig erhält, ist eine Funktion der Größe des Suchraums *(m × n)*, der normalisierten Austauschbewertungen *(λS)*, und einer Konstanten *(k)*.

Der E-Value spielt bei BLAST-Alignments eine ähnliche Rolle wie der bekannte statistische p-Wert. In der Tat ergibt sich der E-Value aus dem p-Wert multipliziert mit der Anzahl der Sequenzen in der Datenbank.

Der p-Wert ist die Wahrscheinlichkeit, dass man zwischen dem HSP in der Eingabesequenz und **einer** zufällig ausgewählten Sequenz ein Alignment mit mindestens genauso guter Bewertung erhält, wie den mit BLAST gefundenen Treffer. Wenn man aber eine Datenbank mit einer hohen Anzahl an Sequenzen hat, steigt die W'keit für einen Zufallstreffer proportional (linear) mit der Größe der Datenbank an. Ebenso hängt der E-Value linear von der Länge der Eingabesequenz ab. Wenn diese doppelt so lange ist, ist die Chance auf einen Zufallstreffer doppelt so hoch wie zuvor.

Der Term exp(- lambda x Bewertung durch Austauschmatrix) bildet den Kern der Karlin-Altschul-Statistik. Dahinter steckt die Annahme einer **Poisson-Verteilung**. Mehr Details dazu bei:

https://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html

und in

https://www.pnas.org/content/pnas/87/6/2264.full.pdf

Bedeutung des Alignments in BLAST

E-Wert (Erwartungswert)

- E = P × Anzahl der Sequenzen in Datenbank
- E entspricht der Anzahl an Alignments einer bestimmten Bewertung, die man zufällig in einer Sequenz-Datenbank dieser Grösse erwartet (wird z.B. für ein Sequenzalignment E=10 angegeben, erwartet man 10 zufällige Treffer mit der gleichen Bewertung).
  Dieses Alignment ist also nicht signifikant.
- Treffer werden in BLAST nur ausgegeben, wenn der E-Wert kleiner als eine vorgewählte Schranke ist.

Der E-Wert (E) gibt – wie erwähnt – die Anzahl an zufälligen Treffern an, die in einer Datenbank dieser Größe mit mindestens genauso guter Bewertung "erwartet" werden. Der E-Wert sinkt exponentiell mit der Bewertung (S) für das Alignment der beiden Sequenzen ab. Gewissermaßen bezeichnet der E-Wert das zufällige Hintergrundrauschen für Treffer zwischen Sequenzen.

Ein E-Wert von 1 bedeutet, dass man in einer Datenbank dieser Größe einen genauso guten Zufallstreffer erwarten kann. Solch ein Ergebnis ist dann natürlich nicht bedeutsam. Das ist, als ob man ein Experiment durchführt, und hinterher die Korrektheit der Ausgangshypothese nur mit 50:50 bewerten kann.

BLAST verwendet für den E-Wert 10 als Default! **Diesen Wert sollte man in den meisten Anwendungen deutlich tiefer setzen**.

Grobe Anhaltspunkte für E-Wert

- Orthologe Sequenzen haben üblicherweise extrem signifikante Bewertungen, z.B. DNA 10^{-100}, Protein 10^{-30}
- Eng verwandte paraloge Sequenzen haben sehr signifikante Bewertungen Protein 10^{-15}
- Entfernt verwandte Homologe können schwierig zu identifizieren sein Protein 10^{-4}

- **Orthologe**:
  verwandte Sequenzen in verschiedenen Spezien, die von einem gemeinsamen Vorläufer abstammen
  - z.B. α Hämoglobin von Mensch, Maus und Huhn

- **Paraloge**:
  verwandte Sequenzen in derselben Spezies, entstanden durch Genduplikation
  -z.B. die Gene α und β Hämoglobin

Als oberste Schranke für den E-Wert verwenden wir normalerweise 10^{-4}. Treffer oberhalb dieser Grenze schauen wir uns nicht an. In diesem Bereich gibt es andere, besser geeignete Tools wie z.B. PSI-BLAST, das wir auf Folie 44 kurz behandeln werden.

Traditionelle BLAST Programme

Program	Database	Query	Typical uses
BLASTN	Nucleotide	Nucleotide	Mapping oligonucleotides, cDNAs and PCR products to a genome, screening repetitive elements; cross-species sequence exploration; annotating genomic DNA sequencing reads
BLASTP	Protein	Protein	Identifying common regions between proteins; collecting related proteins for phylogenetic analyses
BLASTX	Protein	Nucleotide translated into protein	Finding protein-coding genes in genomic DNA; determining if a cDNA corresponds to a known protein
TBLASTN	Nucleotide translated	Protein	Identifying transcripts, potentially from multiple organisms, similar to a given protein; mapping a protein to genomic DNA into protein
TBLAST	Nucleotide translated into protein	Nucleotide translated into protein	Cross-species gene prediction at the genome or transcript level; searching for genes missed by traditional methods or not yet in protein databases

Es gibt verschiedene Varianten des BLAST-Programms für unterschiedliche Zwecke. Es ist immer vorteilhaft, Proteinsequenzen miteinander zu vergleichen.

BLAST Ausgabe (4)

```
>swissprot:CTRB_HUMAN Chymotrypsinogen B precursor (EC 3.4.21.1).
          Length = 263

 Score =  433 bits (1222), Expect = e-121
 Identities = 220/263 (83%), Positives = 252/263 (95%), Gaps = 2/263 (0%)

Query: 1   MAFIWLLSCYALLGTTFGCGVNAIHPVLTGLSKIVNGEEAVPGTWPWQVTLQDRSGFHFC 60
           MAF+WLLSC+ALLGTTFGCGV AIHPVL+GLS+IVNGE+AVPG+WPWQV+LQD++GFHFC
Sbjct: 1   MAFLWLLSCWALLGTTFGCGVPAIHPVLSGLSRIVNGEDAVPGSWPWQVSLQDKTGFHFC 60

Query: 61  GGSLISEDWVVTAAHCGVRTSEILIAGEFDQGSDEDNIQVLRIAKVFKQPKYSILTVNND 120
           GGSLISEDWVVTAAHCGVRTS++++AGEFDQGSDE+NIQVL+IAKVFK PK+SILTVNND
Sbjct: 61  GGSLISEDWVVTAAHCGVRTSDVVVAGEFDQGSDEENIQVLKIAKVFKNPKFSILTVNND 120

Query: 121 ITLLKLASPARYSQTISAVCLPSVDDD--AGSLCATTGWGRTKYNANKSPDKLERAALPL 178
           ITLLKLA+PAR+SQT+SAVCLPS DDD  AG+LCATTGWG+TKYNANK+PDKL++AALPL
Sbjct: 121 ITLLKLATPARFSQTVSAVCLPSADDDFPAGTLCATTGWGKTKYNANKTPDKLQQAALPL 180

Query: 179 LTNAECKRSWGRRLTDVMICGAASGVSSCMGDSGGPLVCQKDGAYTLVAIVSWASDTCSA 238
           L+NAECK+SWGRR+TDVMIC  ASGVSSCMGDSGGPLVCQKDGA+TLV IVSW SDTCS
Sbjct: 181 LSNAECKKSWGRRITDVMICAGASGVSSCMGDSGGPLVCQKDGAWTLVGIVSWGSDTCST 240

Query: 239 SSGGVYAKVTKIIPWVQKILSSN 261
           SS GVYA+VTK+IPWVQKIL++N
Sbjct: 241 SSPGVYARVTKLIPWVQKILAAN 263
```


Dieser beste Treffer von Folie 32 ist ein Beispiel für einen sehr guten Treffer.

„Query“ ist die Eingabesequenz. „Subject“ ist der Treffer. Zwischen beiden Sequenzen sind die Buchstaben aufgelistet, wenn beide Sequenzen an dieser Stelle denselben Buchstaben enthalten. Ein „+“ bedeutet, dass die beiden Aminosäuren ähnlichen Charakter haben, also z.B. die Paarung von Isoleucin und Leucin an Position 4.

Translatierte Proteinsequenzen starten normalerweise mit „M“ = Methionin.

In diesem Fall stimmen die beiden Sequenzen in 83% aller Positionen überein. Inklusive der ähnlichen Positionen sind es sogar 95%. In 263 Positionen gibt es nur 2 Lücken (gaps). Die beiden Sequenzen sind vermutlich ortholog.

BLAST Ausgabe (5)

```
>swissprot:VSP5_TRIMU Mucrofibrase 5 precursor (EC 3.4.21.-).
          Length = 257

 Score =  103 bits (280), Expect = 3e-22
 Identities = 74/232 (31%), Positives = 110/232 (46%), Gaps = 10/232 (4%)

Query: 34  IVNGEEAVPGTWPWQVTLQDRSGFHFCGGSLISEDWVVTAAHCGVRTSEILIAGEFDQGS 93
           I+ G+E      P+ V +      + CGG+LI+E+WV+TAAHC     EI +     +
Sbjct: 25  IIGGDECNINEHPFLVLVYYDD--YQCGGTLINEEWVLTAAHCNGENMEIYLGMHSKKVP 82

Query: 94  DEDNIQVLRIAKVFKQPKYSILTVNNDITLLKLASPARYSQTISAVCLPSVDDDAGSLCA 153
           ++D  + +   K F     +    N DI L++L   P R S  I+ + LPS     GS+C
Sbjct: 83  NKDRRRRVPKEKFFCDSSKNYTKWNKDIMLIRLNRPVRKSAHIAPLSLPSSPPSVGSVCR 142

Query: 154 TTGWGRTKYNANKSPDKLERAALPLLTNAECKRSW-GRRLTDVMICGA--ASGVSSCMGD 210
             GWG          PD     A + LL     C+ ++ G    T    +C      G  SC GD
Sbjct: 143 IMGWGTISPTKVTLPDVPRCANINLLDYEVCRAAYAGLPATSRTLCAGILEGGKDSCGGD 202

Query: 211 SGGPLVCQKDGAYTLVAIVSWASDTCS-ASSGGVYAKVTKIIPWVQKILSSN 261
           SGGPL+C  +G +    IVSW  D C+     G+Y  V   + W++ I++ N
Sbjct: 203 SGGPLIC--NGQFQ--GIVSWGGDPCAQPHEPGLYTNVFDHLDWIKGIIAGN 250
```


Dieser Treffer hat einen E-Wert von 10^{-22}, ist also immer noch ein sehr signifikanter Treffer.

Nur 31% der Positionen sind identisch, 46% positiv und es gibt 10 Gaps.

BLAST Ausgabe (6)

```
>swissprot:HPT_RABIT Haptoglobin beta chain (Fragment).
          Length = 40

 Score = 31.3 bits (74), Expect = 1.4
 Identities = 15/41 (36%), Positives = 22/41 (53%), Gaps = 1/41 (2%)

Query: 34 IVNGEEAVPGTWPWQVTLQDRSGFHFCGGSLISEDWVVTAA 74
          I+ G     G++PWQ  +  R      G +LISE W++T A
Sbjct: 1  IIGGSLDAKGSFPWQAKMVSRHNL-VTGATLISEQWLLTTA 40


>swissprot:NMT1_ASPPA NMT1 protein homolog.
          Length = 342

 Score = 29.6 bits (69), Expect = 4.8
 Identities = 11/34 (32%), Positives = 22/34 (64%)

Query: 72  TAAHCGVRTSEILIAGEFDQGSDEDNIQVLRIAK 105
           TA  CG+  ++ +I G+ D G   +N+Q++ +A+
Sbjct: 137 TAVRCGMNVTKAIIRGDIDAGIGLENVQMVELAE 170
```

Obwohl ein hoher Anteil an identischen und positiven Positionen vorliegt, haben beide Treffer aufgrund ihrer kurzen Länge sehr hohe E-Werte.

Solche „Treffer“ für kurze Sequenzabschnitte können oft zufällig sein.

Dies sind zwei Beispiele vom Ende der Treffer-Liste. Da die Sequenzen sehr kurz sind, ist die Chance klein, eine sehr hohe Bewertung zu erzielen. Beachten Sie, dass der Anteil an identischen und positiven Positionen höher ist als bei dem signifikanten Treffer auf der vorherigen Folie! Allerdings sind diese Sequenzen einfach zu kurz. Diese Treffer haben keine biologische Bedeutung.

Tips für den Einsatz von BLAST

Verwende nicht stur die Standardparameter “You get what you look for”.

Führe gegebenenfalls Kontrollen durch, besonders in der twilight zone.
z.B. Schüttle die Sequenz durcheinander und wiederhole die Suche.
Falls die variierte Sequenz ähnliche Ergebnisse liefert, beruht das Alignment auf einer systematischen Verfälschung, oder die Parameter sind nicht empfindlich genug gewählt

Setze **Komplexitätsfilter** ein, wenn erforderlich.

Maskiere Repeats in genomischer DNA.

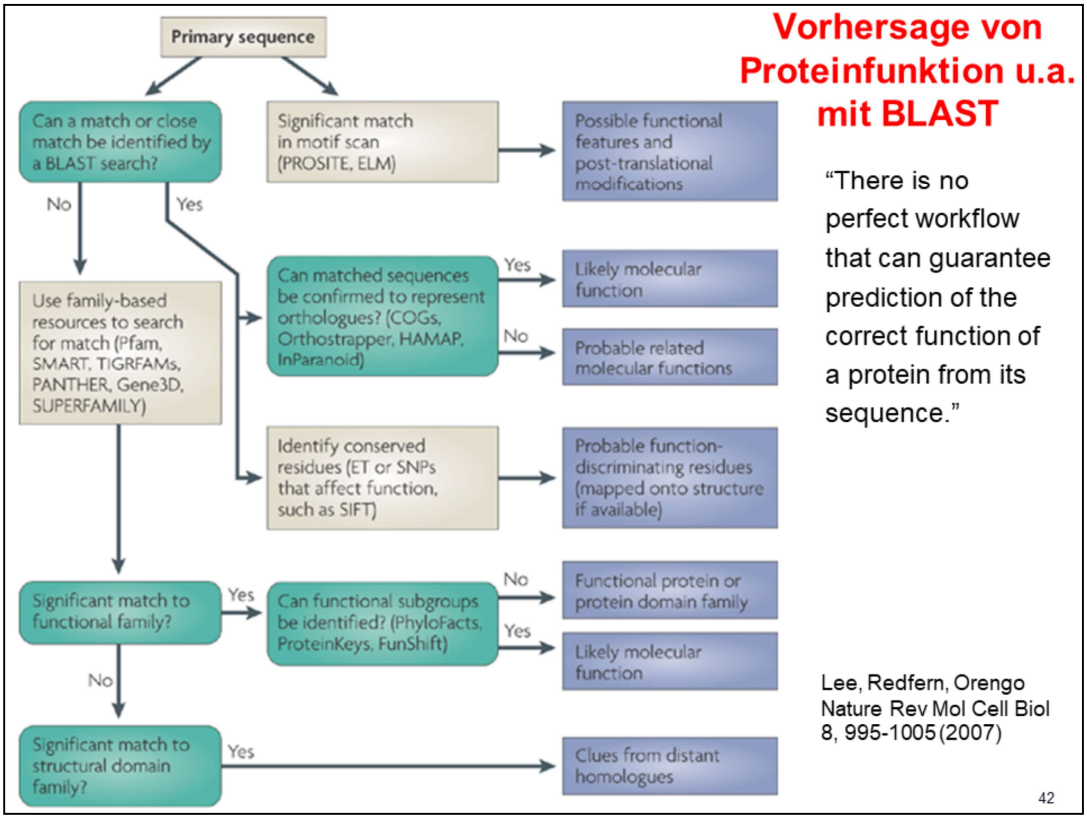
Teile lange Genomsequenzen in Stücke auf um die Suche zu beschleunigen.

Dies sind ein paar praktische Tips für den Einsatz von BLAST. Der Begriff „twilight zone“ wird in der nächsten Vorlesung V3 erklärt.

Komplexitätsfilter unterdrücken Regionen geringer Komplexität, siehe Kommentarfeld auf Folie 30.

Repeatregionen sind häufig in genomischer DNA und kommen zum Teil Millionen-fach im Genom vor. Solche „Treffer“ sind nicht hilfreich. Wir behandeln das Thema Repeats kurz in Vorlesung V4.

Bei langen Sequenzen kann es vorteilhaft sein, die Sequenzen zu unterteilen um die Laufzeit einzugrenzen.



Dies ist ein Überblick, wie man die Funktion einer unbekannten Eingabesequenz durch Bioinformatik-Tools entschlüsseln kann.

Bemerkenswerterweise steht gleich am Anfang eine BLAST-Suche. Dies ist also der wichtigste Schritt. Deshalb ist BLAST auch so erfolgreich.

Wieviel Sequenzidentität ist erforderlich?

Beispiel: EC-Klassifizierung

- *EC 3* Enzyme sind **Hydrolasen** (Enzyme, die Wasser nutzen um ein anderes Moleküle zu spalten)
- *EC 3.4* sind Hydrolasen, die auf **Peptidbindungen** einwirken
- *EC 3.4.11* sind Hydrolasen, die die N-terminale Aminosäure von einem **Polypeptide** abspalten
- *EC 3.4.11.4* sind diejenigen, die die N-terminale Aminosäure von einem **Tripeptide** abspalten

40% paarweise Sequenzidentität zwischen 2 Proteinsequenzen ist eine zuverlässige untere Schranke um die ersten 3 Ziffern der EC-Nummer des einen Proteins dem anderen zuzuordnen.

Um alle 4 EC-Ziffern mit mindestens 90% Trefferquote zuzuordnen, benötigt man >60% Sequenzidentität.

Lee, Redfern, Orengo
Nature Rev Mol Cell Biol
8, 995-1005 (2007)

In diesem Beispiel untersuchten dieselben Autoren wie auf der vorherigen Folie, welchen Grad an Sequenzidentität Enzyme aufweisen, die zu derselben Hierarchieebene der *Enzyme Classification* gehörten.

EC steht für *Enzyme Classification*, siehe https://www.qmul.ac.uk/sbcs/iubmb/

2 Enzyme mit denselben 4 EC-Ziffern haben im Allgemeinen mehr als 60% Sequenzidentität.

Mit diesem Beispiel bekommt man etwas ein Gefühl dafür, wieviel Maß man an Übereinstimmung man gewöhnlich erwarten kann.

PSI-BLAST: Position-Specific Iterated BLAST

- Entfernte Verwandtschaften lassen sich besser durch Motiv- oder Profil-Suchen entdecken als durch paarweise Vergleiche
- PSI-BLAST führt zunächst eine BLAST-Suche mit Gaps durch und identifiziert signifikante Treffer (z.B. 500 beste Treffer mit E-value < 0.001)
- Berechne aus den beobachteten Häufigkeiten der 20 Aminosäuren in den einzelnen Positionen des Alignments die Wahrscheinlichkeit, mit der die Aminosäuren an den Positionen auftauchen (können). Dies nennt man ein **Sequenzprofil** (siehe Vorlesung 6).
- Das PSI-BLAST Programm benutzt das erstellte Sequenzprofil mit der Dimension $L \times 20$ anstelle der normalen 20×20 **Austauschmatrizen** für die nächsten Runden der Datenbank-Suche.
- PSI-BLAST kann iterativ verwendet werden bis keine neuen signifikanten Alignments mehr gefunden werden.
- Fazit: benutze PSI-BLAST um entfernt verwandte Sequenzen zu finden.

Entfernt verwandte Sequenzen entdeckt man am besten durch Motifsuchen oder Profilsuchen.

Auf Folie 7 (PAM250-Matrix) haben wir das Problem erwähnt, dass die Statistik über die Austausche zwischen Aminosäuren üblicherweise nicht berücksichtigt, wo im Protein die entsprechenden Aminosäuren liegen. Natürlich würde es aber eine große Rolle spielen, ob die Positionen auf der Oberfläche, im Inneren des Proteins, oder in seinem aktiven Zentrum liegen. Dieses Problem führte zur Entwicklung von **PSI-BLAST**. Die Idee ist hier, dass für jede Position der Eingabesequenz eine eigene Austauschmatrix konstruiert wird. Man benötigt dazu keine Kenntnisse über die Proteinstruktur, sondern erzeugt die Matrix „aus den Daten“, d.h. aus der Statistik nah verwandter Sequenzen. Die in der Natur auftretenden Mutationen berücksichtigen natürlich die Eigenheiten der Positionen. Mutationen, die eine kritische Funktion des Proteins beschädigen, führen meist zur Aussterben des mutierten Organismus, und werden daher nicht beobachtet.

Aus den mit BLAST identifizierten Treffern wird ein **Sequenzprofil** erstellt (eine Matrix mit der Dimension Länge der Sequenz x 20 Aminosäuren + Gaps). Dieses Sequenzprofil enthält für jede Position die beobachtete Häufigkeit aller 20 Aminosäuren an dieser Position. In weiteren Iterationen wird nun jeweils das Sequenzprofil aus der früheren Iteration verwendet um weitere Treffer zu finden.

PSI-BLAST ist insbesondere geeignet um entfernt verwandte Sequenzen zu finden. Der Grund dafür ist, dass die positionsspezifische Bewertung sensitiver ist als eine über alle Positionen gemittelte Bewertung.

Zusammenfassung

Paarweises Sequenzalignment ist heute Routine, aber nicht trivial.

Mit **dynamischer Programmierung** (z.B. Smith-Waterman) findet man garantiert das Alignment mit optimaler Bewertung.
Vorsicht: die Bewertungsfunktion ist nur ein Modell der biologischen Evolution.

Die schnellste Alignmentmethode ist BLAST und seine Derivate wie BLAT.
Es ergibt sehr robuste und brauchbare Ergebnisse für Proteinsequenzen.

Multiple Sequenzalignments sind in der Lage, entferntere Ähnlichkeiten aufzuspüren und bieten ein besseres funktionelles Verständnis von Sequenzen und ihren Beziehungen

Kommt nächste Woche dran.

In dieser Vorlesung haben wir zunächst das Bewertungsschema (PAM / Blosum-Matrizen) für Sequenzvergleiche kennengelernt.

Im Anschluss wurden 2 Sorten von Alignment-Algorithmen vorgestellt. Das Verständnis dieser Methoden ist eine wichtige Voraussetzungen dafür, dass man die Qualität der mit diesen Tools erzielten Ergebnisse zuverlässig einschätzen kann.