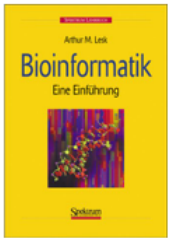
V3 - Multiples Sequenz Alignment und Phylogenie

Literatur: Kapitel 4 in Buch von David Mount

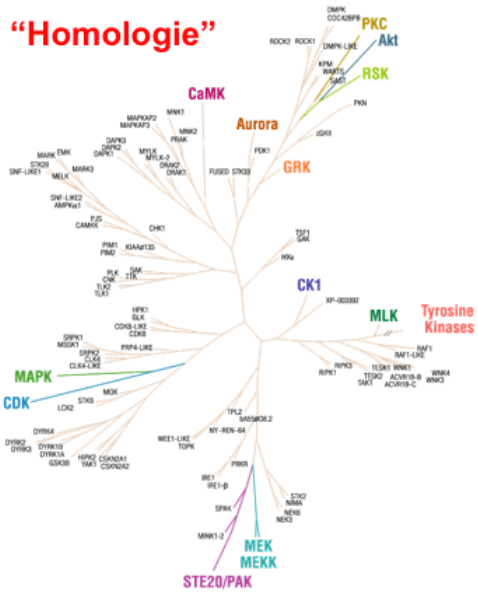
Thioredoxin-Beispiel heute aus Buch von Arthur Lesk



In Vorlesung #3 werden wir das Thema Sequenzalignment von Vorlesung #2 weiter vertiefen und uns heute mit dem Alignment von mehreren bzw. vielen Sequenzen beschäftigen.

Definition von "Homologie"

- **Homologie**: **Sequenzähnlichkeit**, die durch Abstammung von einem gemeinsamen Ursprungsgen herrührt – die Identifizierung und Analyse von Homologien ist eine zentrale Aufgabe der **Phylogenie**.
- Ein **Alignment** ist eine Hypothese für die positionelle Homologie zwischen Basenpaaren bzw. Aminosäuren.
- Homologie ist eine gute Basis, um auf eine ähnliche **Proteinstruktur** zu schließen. Aussagen über ähnliche **Funktion** sind schwieriger.



Phylogenie aller menschlichen Proteinkinasen

http://www.cellsignal.com

Den Begriff „Homologie" hatten wir bereits in V2 (Folie 36) erwähnt.

Homologie bedeutet die evolutionäre Abstammung mehrerer Proteinsequenzen von einer gemeinsamen Vorläufersequenz, siehe

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3820096/

Man unterscheidet zudem „**orthologe**" Sequenzen – das sind homologe Sequenzen, die bei der Artenbildung entstanden sind - und „**paraloge**" Sequenzen – das sind homologe Sequenzen, die durch Genduplikation entstanden sind. Dies sind die gebräuchlichen Definitionen, die wir in dieser Vorlesung verwenden werden. William Pearson erwähnt in seinem Artikel allerdings, dass diese Begriffe manchmal auch in leicht unterschiedlichen Bedeutungen verwendet werden.

Leitfragen für V3

Frage 1: Können wir aus dem Vergleich von Protein- (bzw. DNA-) Sequenzen etwas über evolutionäre Prozesse lernen?

Ansatz 1: vergleiche die Aminosäuresequenzen von homologen Proteinen aus verschiedenen Organismen (erster Teil von V3) und leite daraus phylogenetische Stammbäume ab (zweiter Teil von V3).

Methode: (1) suche homologe Proteine in verschiedenen Organismen (BLAST bzw. PSIBlast), (2) führe multiples Sequenzalignment durch (Clustal, MAFFT)

Ansatz 2: vergleiche die kompletten Genomsequenzen verschiedener Organismen (Breakpoint-Analyse) und leite daraus phylogenetische Stammbäume ab (wird in dieser Vorlesung nicht behandelt).

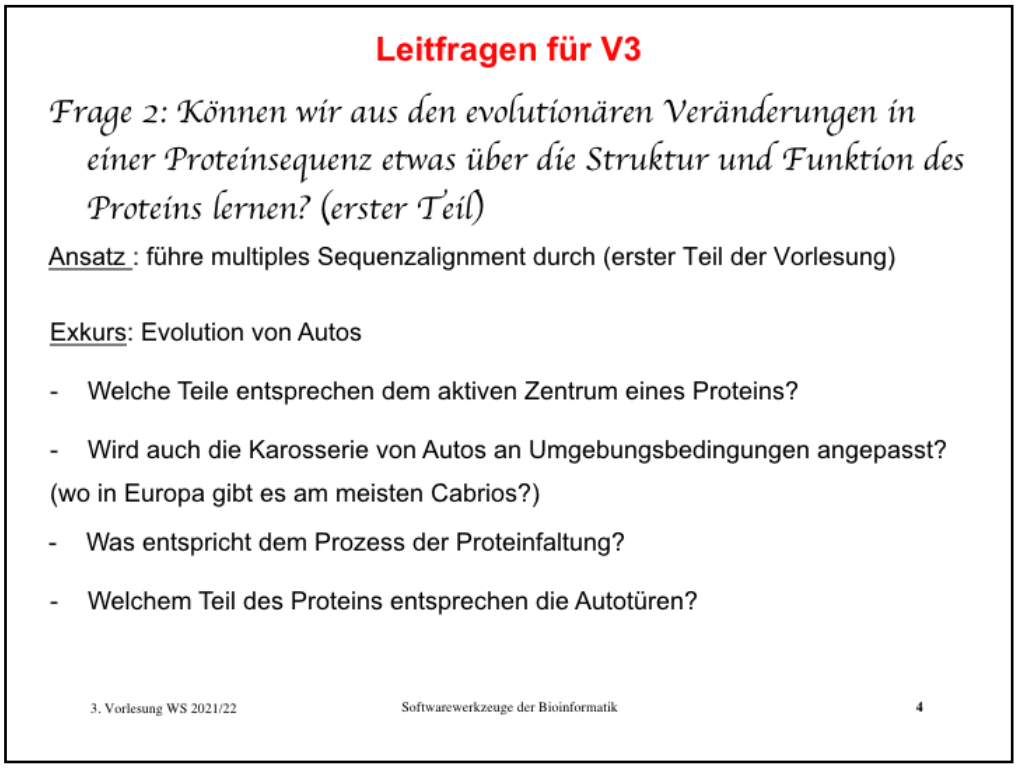
Eine wichtige Anwendung von multiplen Sequenzalignments ist die Aufklärung der evolutionären Stammbäume aller Spezies. Dies klingt vermutlich zunächst einmal nicht sehr aufregend. Man sollte annehmen, dass Stammbäume zwischen allen Spezies eigentlich wohlbekannt sein müssten. Allerdings ergaben sich durch die massiven Genomsequenzierungsprojekte der letzten Jahre auch auf dem Gebiet der Stammbäume einige Überraschungen, siehe z.B.

https://www.nature.com/articles/nmicrobiol201648

, das auf Alignments von sechzehn ribosomalen Proteinen beruht, oder

https://www.nature.com/articles/s41467-019-13443-4

Wichtig sind solche Stammbäume natürlich auch bei der Nachverfolgung von Epidemien, wie bei der andauernden Covid-19-Epidemie.



Eine weitere wichtige Frage betrifft den Zusammenhang zwischen **Proteinstruktur** und **Proteinfunktion**. Wir werden dies später anhand eines Beispiels (Thioredoxin) aus dem Buch von Arthur Lesk diskutieren. Zur Vorsicht füge ich gleich jetzt diesen Hinweis an: Aus einer ähnlichen Proteinstruktur kann man nicht immer darauf schließen, dass zwei Proteine haargenau in ihrer Funktion übereinstimmen. Allerdings kann man aus multiplen Sequenzalignments viel mehr über Proteinfunktion lernen als aus paarweisen Alignments. Deshalb kann man die auf dieser Folie gestellte Frage getrost mit „ja“ beantworten – ja, aus den evolutionären Veränderungen kann man etwas über die Struktur und Funktion des Proteins lernen.

Als (lustigen) Exkurs diskutieren wir nun die Evolution von Autos. Angenommen, wir wären Marsmenschen und wüssten nichts über das Leben auf der Erde. Wir hätten jedoch ein superstarkes Teleskop und könnten beobachten, was auf der Erde so vor sich geht. Was könnten wir aus unseren Beobachtungen über die Rolle und die Funktion von Autos lernen? Sicherlich würden wir zunächst nach gemeinsamen Merkmalen suchen und daraus dann unsere Schlüsse ziehen. Genauso machen wir es dann mit Proteinsequenzen.

Alignments können einfach oder schwer sein

```
GCGGCCCA TCAGGTACTT GGTGG
GCGGCCCA TCAGGTAGTT GGTGG
GCGTTCCA TCAGCTGGTT GGTGG
GCGTCCCA TCAGCTAGTT GGTGG
GCGGCGCA TTAGCTAGTT GGTGA
******** ********** *****
```

Einfach

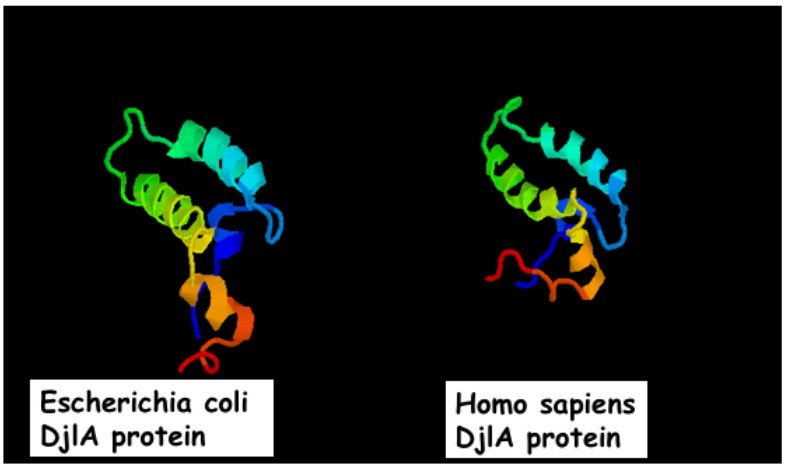
```
TTGACATG CCGGGG---A AACCG
TTGACATG CCGGTG--GT AAGCC
TTGACATG -CTAGG---A ACGCG
TTGACATG -CTAGGGAAC ACGCG
TTGACATC -CTCTG---A ACGCG
******** ?????????? *****
```

Schwierig wegen Insertionen und Deletionen (indels)

Kann man beweisen, dass ein Alignment korrekt ist?

Das obere Alignment sieht sehr einfach aus und könnte vermutlich von einem kleinen Kind korrekt gelöst werden. Das untere Beispiel ist da schon deutlich schwieriger. Zunächst stellen wir uns die Frage, ob man beweisen kann, daß ein multiples Sequenzalignment korrekt ist. Um eine gute Methode entwickeln zu können, müssen wir nämlich die korrekte Antwort kennen.

Protein-Alignment kann durch tertiäre Strukturinformationen geführt werden



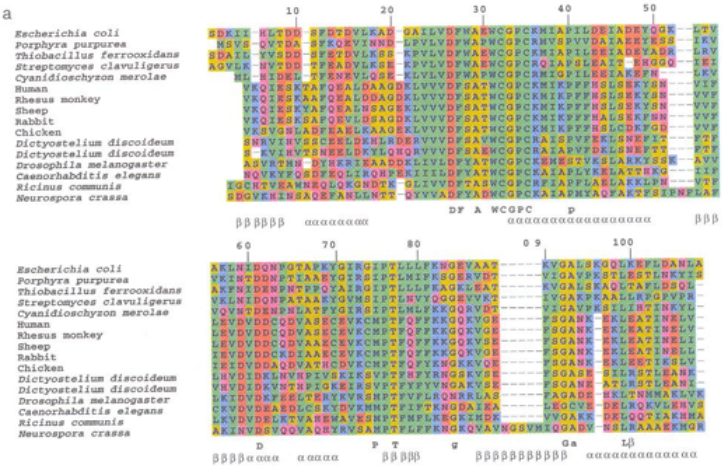
Gaps eines Alignments sollten vorwiegend in Loops liegen, nicht in Sekundär-struktur-elementen.

Anhand von 3D-Strukturen kann man bewerten, ob ein Sequenzalignment sinnvoll ist.
Mathematisch beweisen im strikten Sinne kann man dies jedoch nicht.

Balibase (https://pubmed.ncbi.nlm.nih.gov/16044462/) und HOMSTRAD (https://pubmed.ncbi.nlm.nih.gov/14681395/) sind Zusammenstellungen von strukturbasierten Referenzalignments von Proteinfamilien. Diese beiden Datensätze werden oft für Benchmarks von multiplen Sequenzalignment-Programmen benutzt.

Farbe	Aminosäuretyp	Aminosäuren
gelb	klein, wenig polar	Gly, Ala, Ser, Thr
grün	hydrophob	Cys, Val, Ile, Leu Pro, Phe, Tyr, Met, Trp
violett	polar	Asn, Gln, His
rot	negativ geladen	Asp, Glu
blau	positiv geladen	Lys, Arg

MSA für Thioredoxin-Familie



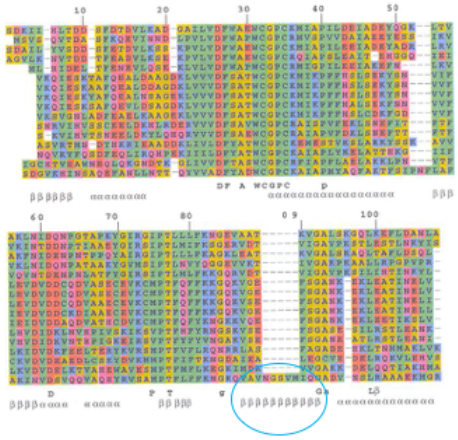
Gezeigt ist hier ein multiples Sequenzalignment für Sequenzen des Proteins Thioredoxin aus 16 verschiedenen Organismen. Wie der Name sagt, reduziert dieses Enzym Schwefelatome von Liganden (meist Cysteinresiduen in anderen Proteinen). Um die Übersichtlichkeit des Alignments zu verbessern, sind die Aminosäuren entsprechend ihrem physikochemischen Typ **eingefärbt**. Rot und blau sind die geladenen Aminosäuren, grün die hydrophoben Aminosäuren.

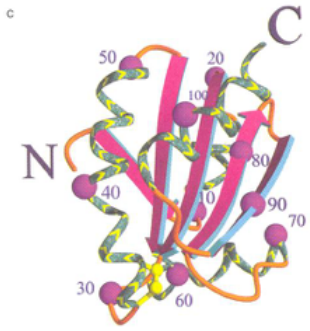
In 2 Regionen hat das Thioredoxin aus *Neurospora crassa* in der untersten Reihe **Insertionen** (eine bei Position 51, eine bei Position 89).

Infos aus MSA von Thioredoxin-Familie

Thioredoxin: aus 5 beta-Strängen bestehendes beta-Faltblatt,
das auf beiden Seiten von alpha-Helices flankiert ist.

gemeinsamer Mechanismus: Reduktion von Disulfidbrücken in Proteinen





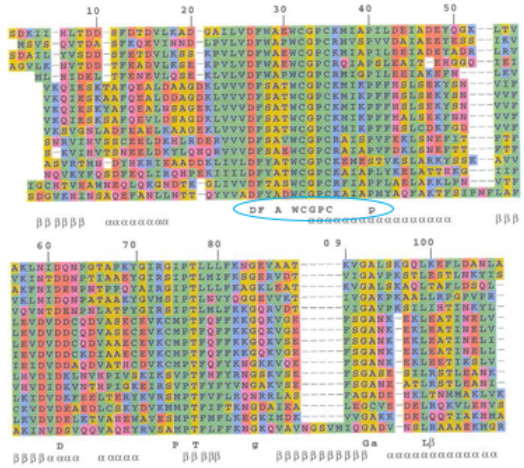
3. Vorlesung WS 2021/22 Softwarewerkzeuge der Bioinformatik 8

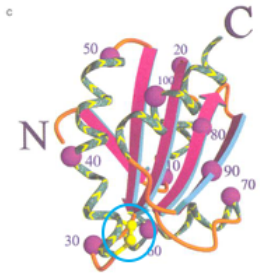
Hier ist rechts zum Vergleich die mit Röntgenkristallographie bestimmte Proteinstruktur von Thioredoxin gezeigt. Das Protein enthält 4 schraubenförmige Alpha-Helices und 4 als Pfeile dargestellte Beta-Stränge. Die Beta-Stränge lagern sich alle zu einem **Beta-Faltblatt** zusammen. Die Alpha-Helices liegen davor, daneben und dahinter. N- und C-Terminus sind ebenfalls markiert. Die Sequenzpositionen sind in 10er-Abständen beschriftet.

Im Sequenzalignment sind die **Sekundärstrukturelemente** ebenfalls markiert. In diesen Regionen erwartet man eine recht gute Konservierung. Dies ist jedoch bei der zweiten Insertion in *Neurospora crassa* (blau umkreist) nicht der Fall. Diese Insertion liegt mitten in einem beta-Strang und wird daher eine strukturelle Änderung des Proteins in diesem Organismus bewirken. Dies ist eine der Ausnahmen von der Regel, die es in der Biologie immer wieder gibt.

Infos aus MSA von Thioredoxin-Familie

1) Die am stärksten konservierten Abschnitte entsprechen wahrscheinlich dem **aktiven Zentrum**. Die **Disulfidbrücke** zwischen Cys32 und Cys35 gehört zu dem konservierten WCGPC[K oder R] Motiv. Andere konservierte Sequenzabschnitte, z.B. Pro76Thr77 und Gly92Ala93 sind an der Substratbindung beteiligt.





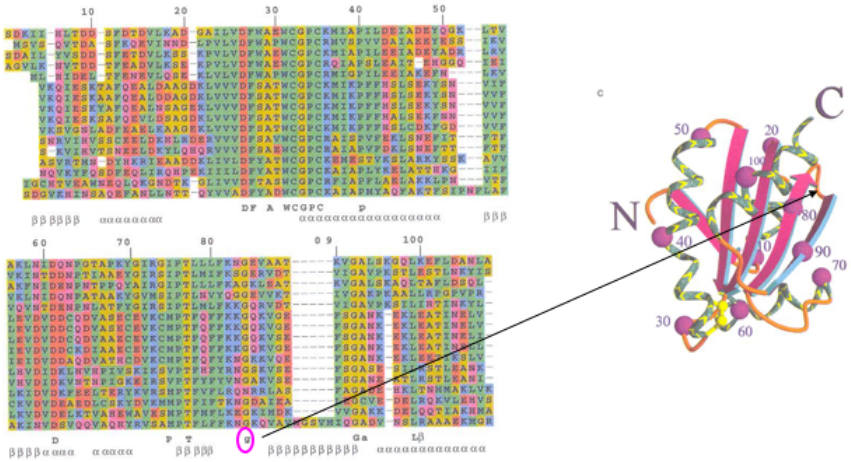
3. Vorlesung WS 2021/22 Softwarewerkzeuge der Bioinformatik 9

Nun beschäftigen wir uns mit denjenigen Aminosäurepositionen, die entweder perfekt oder fast perfekt **konserviert** sind. Diese beiden Fälle werden durch Groß- bzw. Kleinbuchstaben gekennzeichnet.

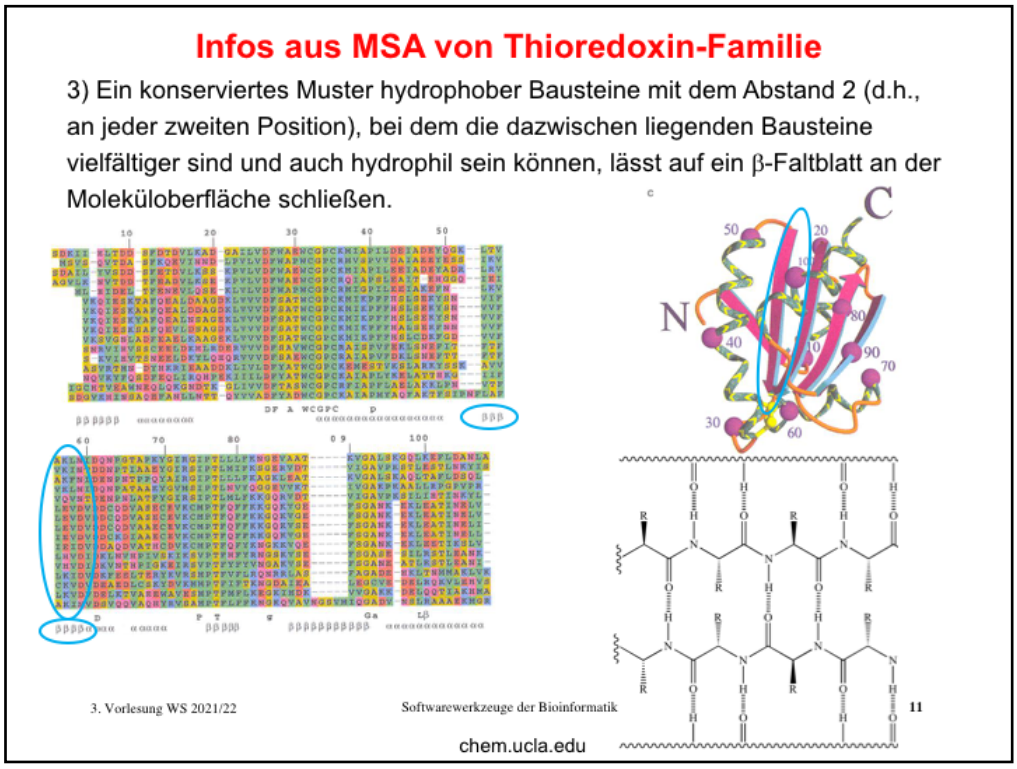
Die beiden Positionen 32 und 35 (blau umkreist) enthalten perfekt konservierte **Cysteine**. Sie liegen in der Struktur eng beieinander und können eine **Disulfidbrücke** ausbilden. Wenn sich ein Substrat in geeigneter Orientierung anlagert, das ebenfalls eine Disulfidbrücke besitzt und diese dann gerade neben Cys32-Cys35 liegt, kann sich eine dieser beiden Disulfidbrücken schließen, dabei Protonen abgeben und sich die andere unter Aufnahme der Protonen öffnen. Dies ist die reduzierende Funktion des Enzyms. Bei Aufnahme von Protonen sinkt die Oxidationszahl der Schwefelatome von 0 auf -1, Schwefel wird also reduziert. Bei der Reaktion ist daher zunächst die Disulfidbrücke auf dem Thioredoxin geöffnet und später die des Substrats. Die beiden Cysteine des Thioredoxins sind in der Natur perfekt konserviert, da jede Variante davon „defekt“ wäre. Das Glycin33 und Prolin34 dazwischen sind ebenfalls konserviert, da diese beiden Residuen vermutlich die spezielle relative räumliche Anordnung von C32 und C35 ermöglichen.

Infos aus MSA von Thioredoxin-Familie

2) Abschnitte mit vielen Insertionen und Deletionen entsprechen vermutlich **Schleifen** (loops) an der Oberfläche. Eine Position mit einem konservierten Gly oder Pro lässt auf eine Wendung der Kette (‚turn') schließen.



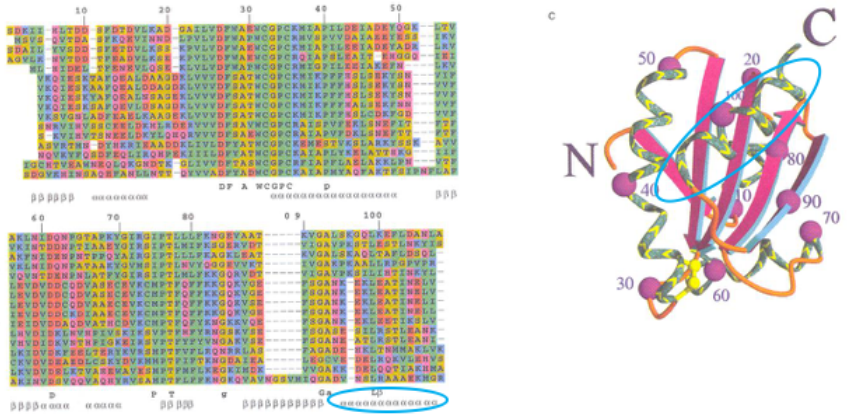
Eine weitere (fast) perfekt konservierte Residue ist G84 (pink umkreist). Es liegt am C-Terminus des beta-Strangs, der bei Position 80 endet. Dort muss die Aminosäurekette eine enge Windung machen um sich gleich wieder in antiparalleler beta-Anordnung an den beta-Strang anzulagern. Das Rückgrat von Glycin ist als einzige Aminosäure quasi frei drehbar (siehe Vorlesung V5). Vermutlich ist diese Eigenschaft für die Ausbildung der engen Windung erforderlich.



Der beta-Strang von Position 52 bis 60 liegt außen an dem beta-Faltblatt (in der Abb. oben rechts blau markiert). In einem beta-Faltblatt (siehe Abb. unten rechts) zeigen die Seitenketten (R) abwechselnd nach „oben“ und nach „unten“. Wenn das **beta-Faltblatt** nun auf der Proteinoberfläche liegt, zeigt die Seitenkette an jeder zweiten Position „ins Protein“ hinein und an den Positionen dazwischen ins Lösungsmittel (hier Wasser). Daher ist es energetisch günstig, wenn die ins Protein zeigenden Seitenketten hydrophob sind und die dazwischen polar. Genau dieses grün /polare **Pattern** sieht man für diesen beta-Strang in dem multiplen Sequenzalignment.

Infos aus MSA von Thioredoxin-Familie

4) Ein konserviertes Muster hydrophober Aminosäurereste mit dem Abstand von ungefähr 4 lässt auf eine α-Helix schließen.



Eine alpha-Helix hat die Periodizität 3,6. D.h. nach jeweils 3,6 Positionen ist eine Umdrehung beendet. Die letzte Helix der Sequenz (blau markiert) liegt ebenfalls auf der Proteinoberfläche. Im multiplen Sequenzalignment sieht man daher (etwas idealisiert) eine Periodizität grün – polar – polar – grün – grün – polar – polar – grün. Dies ist hier nahezu der Fall. Man kann daher anhand des MSAs Ideen über die räumliche Anordnung der Sekundärstrukturelemente entwickeln.

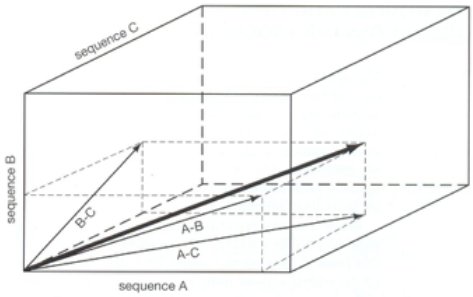
Multiples Sequenzalignment mit dynamischer Programmierung

Liefert wieder optimales Ergebnis.

Für 2 Sequenzen betrachten wir alle möglichen Pfade (Alignments) in einer Matrix n1 * n2.

Für 3 Sequenzen wird Würfel aufgespannt

D.h. dynamische Programmierung von 3 Sequenzen hat Komplexität n1 * n2 * n3 mit den Sequenzlängen n1, n2, n3.



Dies ist extrem aufwändig und nur für sehr kleine *n* möglich!

Nun beschäftigen wir uns mit verschiedenen Algorithmen, mit denen man multiple Sequenzalignments berechnen kann. Wie bei der paarweisen Alignierung gibt es (zumindest theoretisch) die Möglichkeit, **dynamische Programmierung** einzusetzen. Wir hatten in V2 allerdings schon gesehen, dass die algorithmische Komplexität *n1* x *n2* beträgt.

Für 3 Sequenzen müssten wir entsprechend einen Würfel konstruieren und alle Elemente des Würfels berechnen, mit der Komplexität *n1* x *n2* x *n3*, wobei *n1* bis *n3* jeweils die Länge der Sequenzen 1 bis 3 ist. Für mehr als 3 Sequenzen müsste man einen höher dimensionalen Würfel konstruieren … dies ist zu aufwändig und wird in der Praxis nicht eingesetzt

Progressives multiples Sequenzalignment

- wurde von Feng & Doolittle 1987 vorgestellt
- ist ein **heuristisches** Verfahren.
  Daher ist nicht garantiert, das "optimale" Alignment zu finden.
- Das MSA-Alignment wird anhand eines Führungsbaums (guide tree) erstellt. Diesen kann man z.B. aus paarweisen Sequenzalignments (alle gegen alle) erstellen, oder mit effizienteren Methoden.
- weitverbreitete Implementationen in **Clustal** (Des Higgins) bzw. **ClustalW** ("W" steht für Gewichte (weights)) und **ClustalOmega** (Des Higgins)

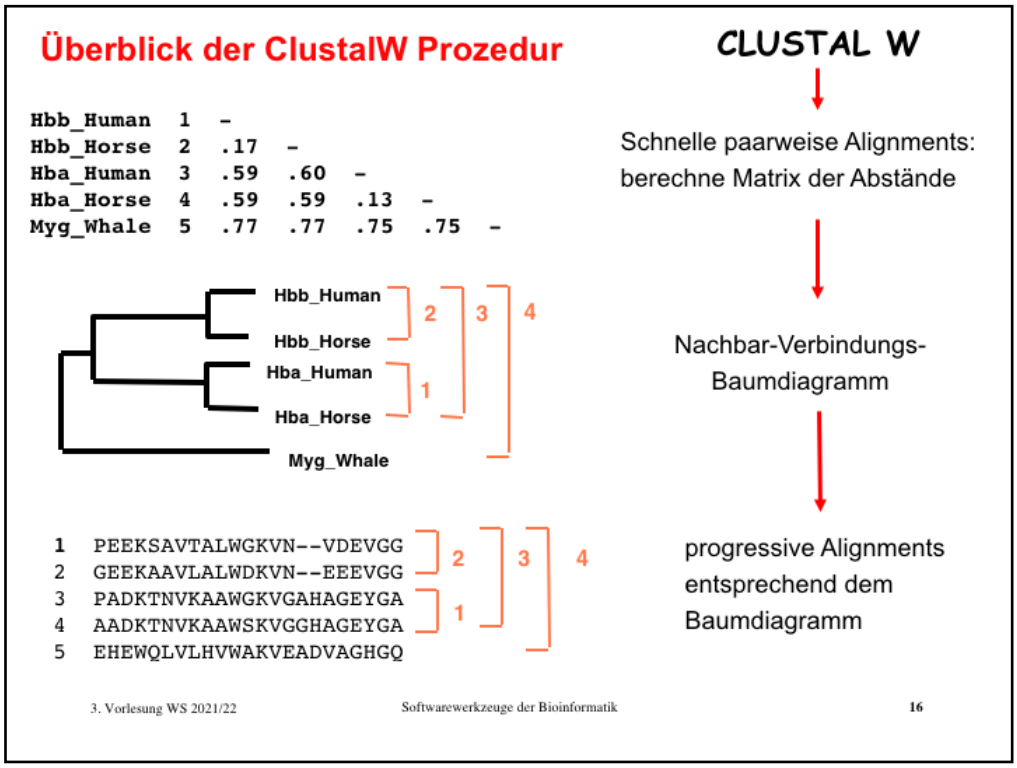
Stattdessen verwenden fast alle MSA-Tools **heuristische Verfahren** (wie BLAST).
Dies ist die Publikation von Feng & Doolittle
https://link.springer.com/article/10.1007/BF02603120

Link für ClustalOmega: http://www.clustal.org/omega/

ClustalW- Paarweise Alignments

- Berechne alle möglichen paarweisen Alignments zwischen *n* Sequenzen. Es gibt *(n-1)+(n-2)...(n-n+1)* solche Alignments.
- Berechne aus diesen isolierten paarweisen Alignments den "Abstand" zwischen jedem Sequenzpaar. Den Abstand könnte man als 1/Ähnlichkeit der Sequenzen definieren.
- Erstelle eine **Abstandsmatrix**.
- Aus den paarweisen Distanzen wird ein **Nachbarschafts-Baum** erstellt
- Dieser Baum gibt die **Reihenfolge** an, in der das progressive Alignment ausgeführt werden wird.

Ohne Kommentare.



Im oberen Beispiel sind die Sequenzabstände für Hämoglobin a und b aus Mensch und Pferd sowie für Myoglobin aus Pottwal aufgetragen. Myoglobin ist am weitesten von den anderen Sequenzen entfernt. Die beiden Hämoglobine a (0.13 Abstand) und b (0.17 Abstand) sind jeweils am ähnlichsten zueinander. Daraus ergibt sich der in der Mitte gezeigt **Abstandsbaum**. Anhand diesem wird das unten gezeigte Alignment erstellt. Erst werden die beiden Paare 1 und 2 miteinander aligniert. Dann wird das Paar der Hbb‘s gegen das Paar der Hba‘s aligniert. Dabei werden die Hba‘s und Hbb‘s als feste Blöcke behandelt. Der Gap mit 2 Positionen wird daher in beide Sequenzen in Block 2 auf dieselbe Weise eingefügt.

ClustalW- Vor- und Nachteile

Vorteil:

- Geschwindigkeit.

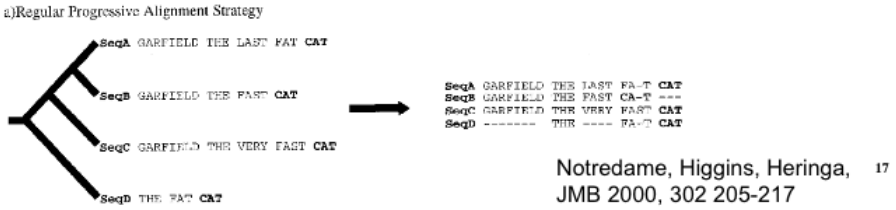
Nachteile:

- keine objektive Funktion.
- Keine Möglichkeit zu quantifizieren, ob Alignment gut oder schlecht ist (vgl. E-value für BLAST)
- Keine Möglichkeit festzustellen, ob das Alignment "korrekt" ist

Mögliche Probleme:

- Prozedur kann in ein lokales Minimum geraten.

  D.h. falls zu einem frühen Zeitpunkt ein Fehler im Alignment eingebaut wird, kann dieser später nicht mehr korrigiert werden, da die bereits alignierten Sequenzen fest bleiben.



Notredame, Higgins, Heringa, JMB 2000, 302 205-217 17

Der Hauptvorteil von progressivem Alignment ist seine höhere Geschwindigkeit. Allerdings gibt es keine statistische Größe wie den E-Wert zur Bewertung der Qualität des MSA. Für das MSA-Tool T-Coffee gibt es ein Bewertungsschema namens TCS, siehe https://academic.oup.com/mbe/article/31/6/1625/2925802

Dies vergibt zwar Bewertungen, aber nicht aufgrund einer statistischen Verteilung.

Bei progressivem Alignment befolgt man das Prinzip "**once a gap, always a gap**." (Feng & Doolittle 1987). Man schaut also nicht zurück.

MSA mit MAFFT-Programm

Ziel: entdecke **lokale Verwandtschaft** zwischen zwei Sequenzen (homologe Segmente) durch Analyse der **Korrelation**.

Dies geht mit der **Fast Fourier Transformation** sehr schnell.

Allerdings braucht man dazu eine **numerische Darstellung** der beiden Sequenzen.

Annahme: evolutionär besonders wichtig sind das **Volumen** und die **Polarität** jeder Aminosäure.

Bilde daher zwei Vektoren der Länge *n*, die die Volumina und Polaritäten aller *n* Aminosäuren enthalten.

Eine große Herausforderung bei der Berechnung von MSAs ist die Geschwindigkeit.

Ein sehr erfolgreiches Tool ist MAFFT, das die Fast Fourier Transformation (FFT) verwendet. FFT ist eine Technik, um sehr schnell Fourier-Transformationen von numerischen Daten zu berechnen. Fourier-Transformationen sind wiederum vorteilhaft um Faltungsintegrale zweier Funktionen zu berechnen. Anstelle aufwändig ein Integral über das Produkt beider Funktion zu berechnen, kann man die beiden fouriertransformierten Funktionen einfach miteinander multiplizieren. Genaueres hierzu behandeln wir z.B. in der Vorlesung Bioinformatik III.

Von welchen numerischen Daten sprechen wir überhaupt? Bisher drehte sich ja alles um Sequenzen, die aus Buchstaben bestehen. Aus den Sequenzen erzeugen wir daher zwei numerische Vektoren mit den Volumina und Polaritäten aller Aminosäuren.

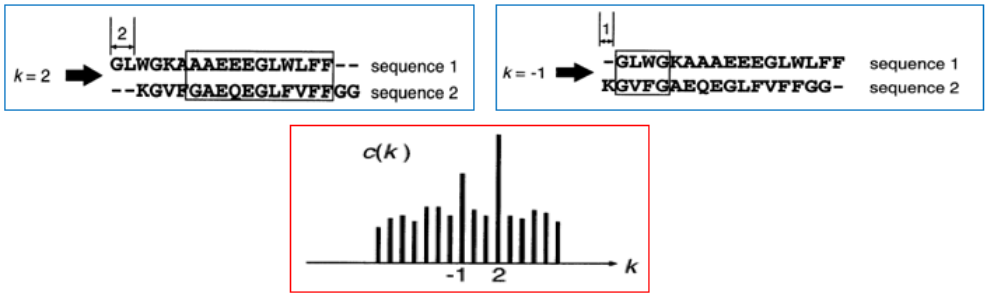
MSA mit MAFFT-Programm

Berechne die Korrelation der beiden Vektoren v_1 , v_2 mit den Aminosäure-Volumina für jede mögliche Verschiebung k: $$c_v(k) = \sum_{1 \le n \le N, 1 \le n+k \le M} \hat{v}_1(n)\hat{v}_2(n+k).$$

und analog die Korrelation der Vektoren mit den Aminosäure-Polaritäten.

Bilde dann die Summe der beiden Korrelationen: $$c(k) = c_v(k) + c_p(k),$$

Schritt 1: Finde passende (d.h. möglicherweise homologe) Segmente mit maximaler Korrelation

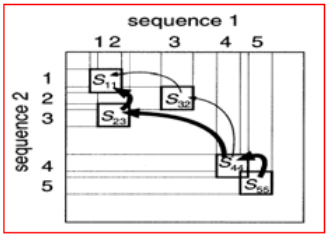


Ein Alignment (ohne gaps) entspricht einer bestimmten relativen Verschiebung der beiden Sequenzen zueinander. In den blau umrandeten Kästchen sehen wir links, dass die untere Sequenz 2 um zwei Positionen relativ zu Sequenz 1 verschoben ist. Im rechten Beispiel ist die obere Sequenz um eine Position verschoben. Wir möchten nun für jede mögliche Verschiebung der beiden Sequenzen die Korrelation (Produkt) der beiden Vektoren bilden. Genau dies ist in der oberen Formel dargestellt. Die Variable *k* bezeichnet hier die Verschiebung. Anstelle der Aminosäure-Buchstaben verwenden wir nun entweder die Volumina *v* oder die Polaritäten. MAFFT addiert die beiden Korrelationen dann einfach zu dem Gesamt-Score.

In der unteren Abbildung (orange umrandet) ist gezeigt, dass die Bewertung für die Verschiebung 2 die beste Bewertung erhält. Im linken Bsp. passt tatsächlich ein längerer Bereich aufeinander (dünner, schwarzer Kasten) als im rechten Beispiel.

MSA mit MAFFT-Programm

Schritt 2: Bilde paarweise Alignments mit eingeschränkter lokaler dynamischer Programmierung:



Schritt 3: erstelle progressiv multiples Alignment:

- Schnelle Berechnung einer Distanzmatrix:
  - → gruppiere 20 Aminosäuren in 6 physikochemische Gruppen
  - → zähle 6-Tuples, die beide Sequenzen gemeinsam haben (vgl. Blast)
- konstruiere Baum mit UPGMA-Methode
- Baue multiples Alignment analog auf

Schritt 4: verfeinere MSA interativ durch Aufteilen des MSAs in 2 Bereiche und Re-Alignierung

Wie gesehen, erzeugt MAFFT zuerst optimale Alignments für kurze, lückenlose Abschnitte. Diese müssen dann aneinandergefügt werden. Hierzu wird eine progressive Alignment-Strategie verwendet. Die Details davon ignorieren wir.

Zusammenfassung

Progressive Alignments sind die am weitesten verbreitete Methode für multiple Sequenzalignments.

Sehr sensitive Methode ebenfalls: Hidden Markov Modelle (z.B. in HMMer).

Multiples Sequenzalignment ist **nicht trivial**.

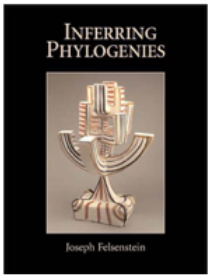
Manuelle Nacharbeit kann in Einzelfällen das Alignment verbessern.

Multiples Sequenzalignment erlaubt **Denken** in **Proteinfamilien** und **Proteinfunktionen**.

Als Zusammenfassung des ersten Teils gilt: progressive multiple Alignments sind heute ein Standardverfahren. Die einschlägigen Tools wie ClustalW (62000 mal zitiert), MAFFT (8600 mal zitiert) oder ClustalOmega (8900 mal zitiert) sind extrem weitverbreitete Tools.

Rekonstruiere Phylogenien aus einzelnen Gensequenzen

Material dieser Vorlesung aus
- Kapitel 6, DW Mount „Bioinformatics“
und aus Buch von Julian Felsenstein.



Eine **phylogenetische Analyse** einer Familie verwandter Nukleinsäure- oder Proteinsequenzen bestimmt, wie sich diese Familie durch Evolution entwickelt haben könnte.
Die evolutionären Beziehungen der Sequenzen können durch Darstellung als Blätter auf einem Baum veranschaulicht werden.

Phylogenien, oder evolutionäre Bäume, sind die Grundlage um Unterschiede zwischen Arten zu beschreiben und statistisch zu analysieren.
Es gibt sie seit über 140 Jahren und seit etwa 40 Jahren mit Hilfe von statistischen, algorithmischen und numerischen Verfahren.

Im zweiten Teil der heutigen Vorlesung beschäftigen wir uns mit der Konstruktion von Phylogenien (Stammbäume mehrerer Organismen) aus multiplen Sequenzalignments. Die Konstruktion von Phylogenien war eine der ersten Forschungsrichtungen, die sich auf biologische Sequenzdaten stützten.

Vor der Zeit der modernen Genetik wurden Phylogenien mit traditionellen Werkzeugen wie der vergleichenden Anatomie und der Embryologie erstellt. Das hier erwähnte Buch von Joseph Felsenstein ist etwas für Spezialisten. Es geht sehr detailliert auf alle Aspekte der Berechnung von Phylogenien ein.

3 Hauptansätze für Phylogenien einzelner Gene

- **maximale Parsimonie** (z.B. Sankoff-Algorithmus, wird heute behandelt)

- **Distanzmatrix** (z.B. Neighbor-Joining Algorithmus, wird heute behandelt)

- **maximum likelihood** (wird hier nicht behandelt)

Häufig verwendete **Programme** für die Berechnung von Phylogenien:
Traditionell:
PHYLIP (phylogenetic inference package – J Felsenstein)
PAUP (phylogenetic analysis using parsimony – Sinauer Assoc)

Modern:

BEAST	RAxML
MEGA	MrBayes
SplitsTree	PhyML

Bei der **maximalen „Parsimony“** (dt. in etwa „maximale Sparsamkeit“) werden diejenigen Bäume bevorzugt, die am wenigsten evolutionären Wandel (Mutationsereignisse) benötigen, um die beobachteten Daten (Sequenzen) zu erklären.

Neighbor-joining basiert auf dem „Minimum Evolution“-Kriterium für phylogenetische Bäume: Ausgehend von einem zunächst sternförmigen „Baum“, in dem alle Taxa mit einem „Zentrum“ verbunden sind, werden paarweise die DNA- bzw. Proteinsequenzen mit der geringsten genetischen Distanz ausgewählt und zu einem Ast des Baumes vereinigt.

Parsimonie Methoden

Luca Cavalli-Sforza

Edwards & Cavalli-Sforza (1964):
derjenige evolutionäre Baum ist zu bevorzugen,
der „den **minimalen Anteil an Evolution**“ enthält.

→ suche Phylogenien, die gerade so viele Zustandsänderungen beinhalten, dass wir mit ihnen die evolutionären Vorgänge rekonstruieren können, die uns zu den vorhandenen Daten (Sequenzen) führen.

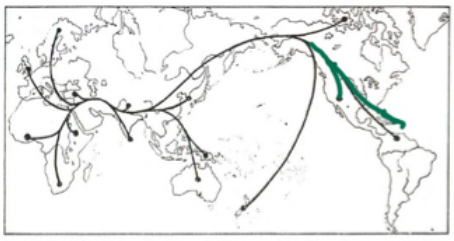


FIG. 1. Topology of the minimum-evolution tree uniting fifteen human populations; constructed on the basis of the frequency of blood-group alleles.

(1) Für jede vorgeschlagene Phylogenie müssen wir in der Lage sein, die Vorgänge zu rekonstruieren, die am wenigsten Zustandsänderungen benötigen.

(2) Wir müssen unter allen möglichen Phylogenien nach denen suchen können, die eine minimale Anzahl an Zustandsänderungen beinhalten.

Luca Cavalli-Sforza war ein italienischer Populationsgenetiker, https://de.wikipedia.org/wiki/Luigi_Luca_Cavalli-Sforza . Ursprünglich hatte er Medizin studiert und auch als Arzt gearbeitet.

1990 initiierte Cavalli-Sforza das Human Genome Projekt, das als Ziel die Entschlüsselung des menschlichen Genoms hatte und dieses Ziel 2001 erfolgreich erreichte.

Laut https://hydrodictyon.eeb.uconn.edu/people/plewis/courses/phylogenetics/lectures/2020/Parsimony.pdf veröffentlichten Michener, C. D., and R. R. Sokal. 1957. A Quantitative Approach to a Problem in Classification. Evolution 11:130-162 bereits im Jahr 1957 die erste mit einem Computer berechnete Phylogenie. Allerdings benutzten sie noch nicht das Prinzip der Parsimonie.

Edwards, A. W. F., and L. L. Cavalli-Sforza. 1964. Reconstruction of evolutionary trees. pp. 67-76 in Phenetic and phylogenetic classification, ed. V. H. Heywood and J. McNeill. Systematics Association Publ. No. 6, London. waren die ersten, die das **Parsimonie-Prinzip** auf die Analyse der Verteilung der menschlichen Blutgruppen anwendeten (siehe Abbildung oben).

Ein einfaches Beispiel

Gegeben seien 6 Buchstaben lange Sequenzen aus 5 Spezies,
die die Werte 0 oder 1 annehmen können

	Characters					
Species	1	2	3	4	5	6
Alpha	1	0	0	1	1	0
Beta	0	0	1	0	0	0
Gamma	1	1	0	0	0	0
Delta	1	1	0	1	1	1
Epsilon	0	0	1	1	1	0

Erlaubt seien Austausche $0 \rightarrow 1$ und $1 \rightarrow 0$.

Der anfängliche Zustand an der Wurzel des Baums kann 0 oder 1 sein.

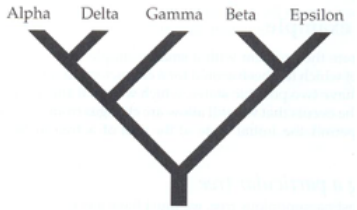
Dieses Beispiel stammt aus dem Buch von Felsenstein.

Wir nehmen in diesem einfachen Beispiel an, dass wir Sequenzen von 5 Spezies vorliegen haben, die jeweils 6 Buchstaben lang sind. Zur Vereinfachung können die Buchstaben in diesem Beispiel nur die Werte 0 und 1 haben (also nicht ACGT).

Ziel: wir möchten den evolutionären Verlauf (Phylogenie) konstruieren, wie diese 5 Spezies aus einer Vorläuferspezies hervorgegangen sein könnten.

Bewerte einen bestimmten Baum

Um den Baum höchster Parsimonität zu finden, müssen wir berechnen können, wie viele Zustandsänderungen für einen gegebenen Baum nötig sind.



	Characters					
Species	1	2	3	4	5	6
Alpha	1	0	0	1	1	0
Beta	0	0	1	0	0	0
Gamma	1	1	0	0	0	0
Delta	1	1	0	1	1	1
Epsilon	0	0	1	1	1	0

Dieser Baum stelle die Phylogenie des ersten Buchstabens dar.

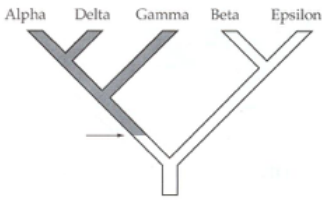
Diese Aufgabe besteht aus zwei Teilen: (1) welches ist die beste Baum-Topologie und (2) wann haben die Mutationen auf dem Baum stattgefunden?

Wir beschäftigen uns erst einmal mit dem zweiten Problem und nehmen an, dass wir bereits eine gute Baum-Topologie gefunden haben. Zunächst einmal betrachten wir die Werte des **ersten Buchstabens** im roten Kasten.

Bewerte einen bestimmten Baum

Es gibt zwei gleich gute Rekonstruktionen,
die jede nur eine Buchstabenänderung benötigen.

Sie nehmen unterschiedliche Zustände an der Wurzel des Baums an
und unterschiedliche Positionen für die eine Änderung.



or

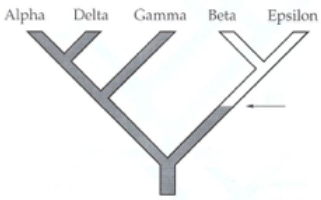


Figure 1.2: Alternative reconstructions of character 1 on the phylogeny of Figure 1.1. The white region of the tree is reconstructed as having state 0, the shaded region as having state 1. The two reconstructions each have one change of state. The changes of state are indicated by arrows.

Da der erste Buchstaben in den 5 vorliegenden Spezies Werte von 0 und 1 hat, der gemeinsame Vorläufer aber nur entweder 0 oder 1 haben kann, benötigen wir mindestens 1 Mutationsereignis. In unserem Beispiel haben die Spezies Beta und Epsilon den Wert 0 und die anderen Spezies den Wert 1. Der vorhandene Baum passt gut zu diesen Werten, da Beta und Epsilon nahe beieinander liegen. Wir können alle beobachteten Werte durch 1 Mutationsereignis darstellen, das entweder im linken oder rechten Ast liegen könnte, je nachdem welchen Wert der Vorläufer hatte. Aufgrund des ersten Buchstabens können wir den Wert des Vorläufers daher für diese Baumtopologie nicht eindeutig bestimmen.

Bewerte einen bestimmten Baum

Hier sind drei gleich gute Rekonstruktionen für den zweiten Buchstaben gezeigt, die jeweils zwei Zustandsänderungen benötigen.

	Characters					
Species	1	2	3	4	5	6
Alpha	1	0	0	1	1	0
Beta	0	0	1	0	0	0
Gamma	1	1	0	0	0	0
Delta	1	1	0	1	1	1
Epsilon	0	0	1	1	1	0

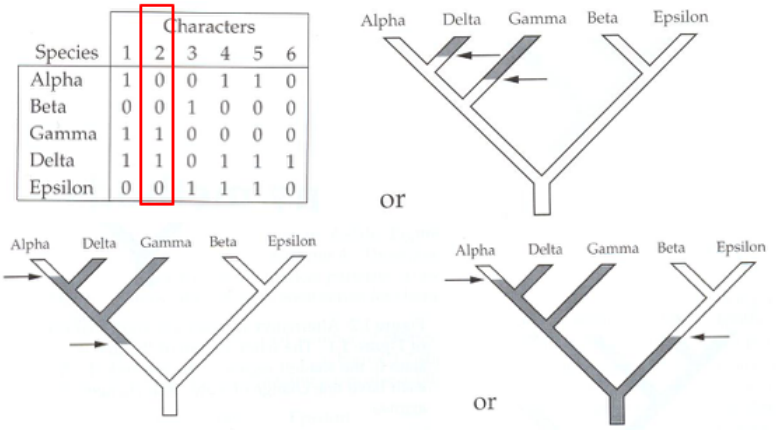


Figure 1.3: Reconstructions of character 2 on the phylogeny of Figure 1.1. The white regions have state 0, the shaded region state 1. The changes of state are indicated by arrows.

Nun schauen wir uns den zweiten Buchstaben an. Beta und Epsilon haben wieder beide den Wert 0. Das passt. Allerdings hat Alpha ebenfalls den Wert 0. Für die gegebene Baumtopologie benötigen wir daher eine zweite Mutation, die an drei verschiedenen Stellen liegen kann, je nachdem welchen Wert der Vorgänger hatte.

Bewerte einen bestimmten Baum

Die gesamte Anzahl an Zustandsänderungen auf diesem Baum ist
1 + 2 + 1 + 2 + 2 + 1 = 9

Rekonstruktion der Zustandsänderungen auf diesem Baum

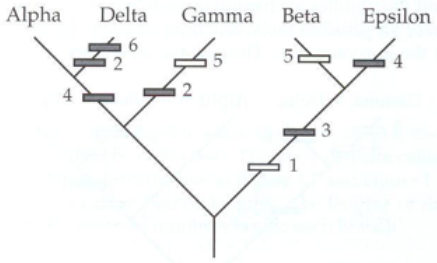


Figure 1.7: Reconstruction of all character changes on the phylogeny of Figure 1.1. The changes are shown as bars across the branches, with a number next to each indicating which character is changing. The shading of each box indicates which state is derived from that change.

Wenn man alle 6 Buchstaben betrachtet, benötigen wir für diese Baum-Topologie mindestens 9 Mutationsereignisse. In der Abbildung ist deren Position gezeigt und der Buchstaben, in dem die jeweilige Mutation auftritt, beschriftet.

Bewerte einen bestimmten Baum

Ein anderer Baum, der nur 8 Zustandsänderungen benötigt.

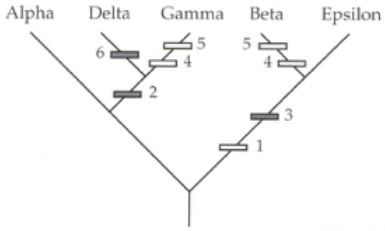
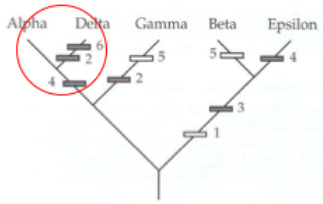


Figure 1.8: Reconstruction of all changes on the most parsimonious phylogeny for the data of Table 1.1. It requires only 8 changes of state. The changes are shown as bars across the branches, with a number next to each indicating which character is changing. The shading of each box indicates which state is derived from that change.

erster Baum (zum Vergleich)



Die prinzipiell minimal erforderliche Anzahl an Zustandsänderungen ist 6, da es 6 Buchstaben gibt, die jeweils 2 Zustände annehmen können.
Für das gezeigte Beispiel sind jedoch 8 Zustandsänderungen erforderlich.

Wenn wir den Pfad, der zu Delta führt, leicht abwandeln, erhalten wir einen modifizierten Baum, der nur 8 Mutationen benötigt. Gemäß dem Prinzip der maximalen Parsimonie ist dieser zu bevorzugen.

Finde den besten Baum durch heuristische Suche

Die naheliegende Methode, den Baum höchster Parsimonie zu finden ist,
ALLE möglichen Bäume zu betrachten und einzeln zu bewerten.

Leider ist die Anzahl an möglichen Bäumen üblicherweise zu groß (n^{n-2} nach Cayley).

→ verwende heuristische Suchmethoden, die versuchen,
die besten Bäume zu finden ohne alle möglichen Bäume zu betrachten.

(1) Konstruiere eine erste Abschätzung des Baums
und verfeinere diesen durch kleine Änderungen
= finde „benachbarte" Bäume.

(2) Wenn irgendwelche dieser Nachbarn besser sind,
verwende diese und setze die Suche fort.

Im Prinzip müssten wir also alle möglichen Baum-Topologien durchsuchen. Allerdings zeigte **Arthur Cayley** (https://en.wikipedia.org/wiki/Cayley%27s_formula) dass die Anzahl der möglichen Bäume für *n* Knoten rasant ansteigt. Man verwendet daher iterative, heuristische Verfahren um (fast) optimale Baum-Topologien zu konstruieren. Zum Schluss der heutigen Vorlesung werden wir die Neighbor-Joining-Methode im Detail besprechen, mit der man gute Baum-Topologien konstruieren kann.

Zähle evolutionäre Zustandsänderungen als Modell für evolutionäre Kosten eines gegebenen Evolutionsbaums

Hierfür existieren zwei verwandte Algorithmen, die beide die *dynamische Programmierung verwenden*: Fitch (1971) und Sankoff (1975)

- bewerte eine Phylogenie Buchstabe für Buchstabe

- betrachte jeden Buchstaben als Baum mit Wurzel an einem geeigneten Platz.

- propagiere eine Information von oben nach unten durch den Baum;
beim Erreichen der Blätter ist die Anzahl der Zustandsänderungen bekannt.

Dabei werden die Zustandsänderungen oder internen Zustände an den Knoten des Baums nicht konstruiert.

Bevor wir also später zur Berechnung von Baum-Topologien kommen, möchten wir erst einmal für eine gegebene Topologie den optimalen Wert des Knotens an der Wurzel berechnen. Dafür verwenden wir den **Sankoff-Algorithmus**. Dieser verwendet wiederum das Prinzip der dynamischen Programmierung sowie das Prinzip der maximalen Parsimonie.

Sankoff Algorithmus

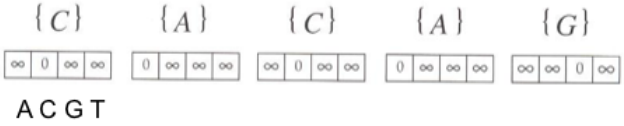
David Sankoff

Gesucht: Modell für Evolution einer Nukleotid-Position.

Zu einem gegebenem Baum, wähle dazu im unteren Endknoten (der zum Ur-Vorläufer gehört) den minimalen Wert, der die minimalen „evolutionären Kosten“ für diesen Buchstaben ausdrückt.

$$S = \min_i S_0(i)$$

Bekannt ist, welche Nukleotidbasen in den heutigen Sequenzen an dieser Position gefunden wird.
Daher ordnen wir an der Spitze des Baums jeder Sequenz die Kosten „0“ für die heute beobachtete Base zu und setzen die Kosten für die anderen 3 Basen auf Unendlich.



Nun brauchen wir einen Algorithmus, der die evolutionären Kosten *S(i)* für den jeweiligen Vorläufer zweier Knoten berechnet.

Wir betrachten in diesem Beispiel wiederum die Buchstaben mehrerer Sequenzen an einer einzelnen Position. Das Ziel ist die Berechnung des Wertes des Vorläuferknotens. Gemäß dem Prinzip der maximalen Parsimonie ordnen wir ihm den Wert geringster evolutionärer Kosten zu, wobei Mutationsereignisse als Kosten betrachtet werden. Dies wird durch die Formel ausgedrückt. Die Variable *i* läuft in diesem Fall über die 4 Nukleotide A, C, G und T.

Es ist bekannt, dass Pyrimidine leichter in Pyrimidine mutieren können und Purine leichter in Purine, als über Kreuz. Die Pyr -> Pyr und Pur -> Pur-Austausche werden wir deshalb mit geringeren Kosten belegen.

Bekannt ist lediglich, welche Buchstaben in den heutigen Sequenzen enthalten sind. Wir setzen die evolutionären Kosten für diese Buchstaben auf 0 und für die anderen Möglichkeiten auf unendlich, da diese Zustände nicht angenommen werden können. Wir rechnen nun in der Evolution rückwärts und werden alle Mutationsereignisse mit Kosten belegen.

Sankoff-Algorithmus

Nenne die beiden Kind-Knoten *l* und *r* (für „links“ und „rechts“).

Die evolutionären Kosten für den direkten Vorgänger *a* (für „ancestor“) seien

$$S_a(i) = \min_j \left[c_{ij} + S_l(j)\right] + \min_k \left[c_{ik} + S_r(k)\right]$$

D.h. die geringst mögliche Kosten dafür, dass Knoten *a* den Zustand *i* hat, sind die Kosten c_{ij} um in der linken Vorgängerlinie vom Zustand *i* zum Zustand *j* zu gelangen plus die bis dahin bereits angefallenen Kosten $S_l(j)$.

- Wähle den Wert *j*, der diese Summe minimiert.

- Entsprechende Berechnung für die rechte Vorgängerlinie, bilde Summe.

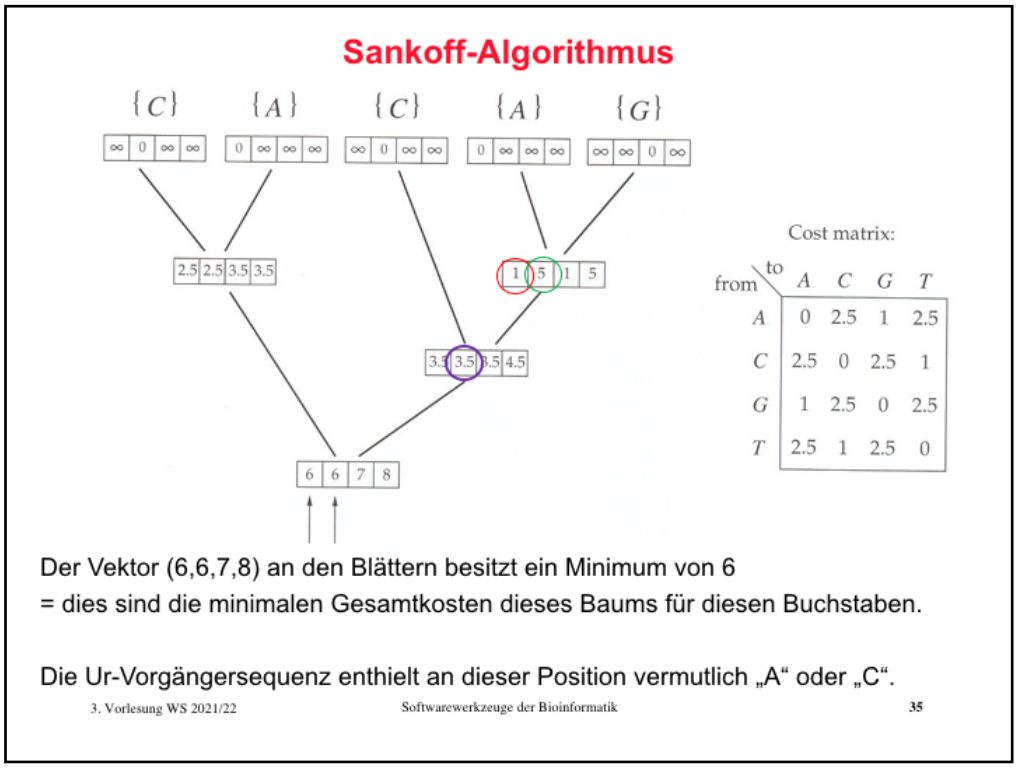
- Wende diese Gleichung sukzessiv auf den ganzen Baum von oben nach unten an.

- Berechne $S_0(i)$ und die minimalen Kosten für den Baum: $S = \min_i S_0(i)$

Jeder Vorläufer-Knoten im Baum hat zwei „Kinder“-Knoten, die links bzw. rechts oberhalb von der Senkrechten liegen. Die Kosten des Vorläufers beinhalten zum einen die Werte seiner beiden Kinder S_l und S_r. Diese sind zu diesem Zeitpunkt bekannt, da wir den Baum von oben nach unten auffüllen werden. Die Werte der heutigen Sequenzen haben wir ja bereits eingesetzt.

Außerdem müssen wir berücksichtigen, ob bei der Vererbung auf die Kinder ein **Mutationsereignis** stattfand, das wir mit den Kosten c_{ij} bewerten.

Wir betrachten alle möglichen Zustände *j* = A,C,G,T des Vorläufer-Knotens und ordnen jedem Zustand dessen minimale Kosten zu. Wir brauchen uns also nicht für eine der Möglichkeiten zu entscheiden. Das ganze wird auf der nächsten Folie anhand eines **Beispiels** im Detail erklärt.



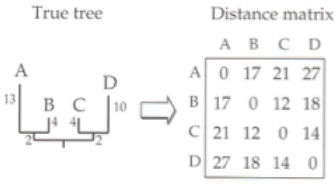
Die heute in den 5 Spezies beobachteten Sequenzen enthalten an einer bestimmten Position die Buchstaben C, A, C, A und G (oben von links nach rechts). Die Baum-Topologie betrachten wir als vorgegeben. Als evolutionäre Kosten für Mutationsereignisse nehmen wir die Werte in der Tabelle an. Wie bereits erläutert, werden die „einfachen“ Austausche von A nach G (bzw. umgekehrt) und von T nach C mit geringen Kosten (1) bewertet. Der Austausch zwischen Pyrimidinen und Purinen wird dagegen mit höheren Kosten (2,5) bewertet. Wenn keine Mutation auftritt, entstehen natürlich keine Kosten.

Wir betrachten zunächst einmal den Vorläufer der beiden rechten Spezies. Für den Zustand A des Vorläufers (roter Kreis), erhalten wir folgende Kosten: das linke Kind hat ebenfalls den Zustand A mit **Kosten 0**, es ist keine Mutation erforderlich (**Kosten = 0**). Das rechte Kind hat den Zustand G mit **Kosten 0.** Der Vorläufer müsste jedoch von A nach G mutieren, was **Kosten von 1** verursacht. Die Summe dieser 4 Terme beträgt 1. Deshalb tragen wir in den roten Kreis den Wert 1 ein. Falls der Vorläufer den Zustand C hätte (grüner Kreis), brauchen wir zum linken Kind eine Mutation von C nach A (Kosten 2,5). Außerdem brauchen wir zum rechten Kind eine Mutation von C nach G (Kosten 2,5). Die Summe ergibt 5. usw.

Manchmal sind die Dinge aber etwas komplizierter als in diesem Fall. Dazu betrachten wir den Vorläufer des Knotens, den wir gerade diskutiert haben und dessen Zustand C (lila Kreis). Dessen linkes Kind hat den Zustand C mit Kosten 0. Es wäre keine Mutation erforderlich (Kosten 0). Unter den möglichen Zuständen des rechten Kindes ist es jedoch vorteilhaft, wenn diese entweder A oder G sind (jeweils Kosten 1) und durch Mutation aus G hervorgehen (jeweils Kosten 2,5). Deshalb

erhält der Zustand G des Vorgängers die Summe 3,5 dieser 4 Terme zugeordnet. Der **Urvorläufer** (Wurzel) hat die Kosten 6, 6, 7 und 8 für die 4 Zustände. Unter diesen sind A und C am günstigsten.

Konstruiere einen guten Baum: neighbor-joining Methode



Unsere Ausgangsbasis ist die Distanzmatrix (üblicherweise aus paarweisen Alignments bestimmt).
Daraus möchten wir den evolutionären Baum konstruieren, der möglichst gut den wahren Verlauf der Evolution angibt.

Neighbor-joining Algorithmus wurde durch Saitou und Nei (1987) eingeführt.
Der Algorithmus verwendet Clustering und das Modell minimaler Evolution.

„Modell minimaler Evolution“
wähle unter den möglichen Baumtopologien
diejenige mit minimaler Gesamtlänge der Äste.

Wenn die Distanzmatrix den wahren Baum exakt abbildet, garantiert die Neighbor-joining Methode, als Methode der geringsten Quadrate, den optimalen Baum zu finden.

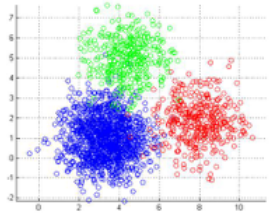
Zum Abschluss von V3 diskutieren wir die **Neighbor Joining-Methode**, mit der man durch „Verbinden der (jeweils) nächsten Nachbarn“ Baumtopologien konstruieren kann.

Die Ausgangsbasis ist eine **Distanzmatrix** zwischen allen Spezies. Wir nehmen an, dass sie die Abstände zwischen den Spezies perfekt darstellt. Im Beispiel links oben ist solch ein idealer Fall gezeigt, wo die Abstände im (unbekannten) Baum und in der Distanzmatrix perfekt übereinstimmen. Z.B. ist der Abstand der Knoten A und B 13+4=17 und der Abstand von A und D 13+2+2+10=27. In solch einem Fall liefert die Neighbor joining-Methode den optimalen Baum.

Wenn man aber z.B. in der Distanzmatrix nur den Abstand von A und B von 17 auf 18 erhöhen würde und alle anderen Abstände gleich lassen würde, dann ließe sich kein perfekt passender Baum mehr finden.

Clustern

- Clustern gruppiert eine Menge von Objekten in Teilmengen oder “Cluster”
- Objekte in einem Cluster sind ähnlicher zueinander als zu Objekten in anderen Clustern



http://www.mathworks.com/matlabcentral/fileexchange/screenshots/6432/original.jpg

Hierarchisches Clustern:

•Hierarchische Darstellung (als Baum dargestellt)

•Es gibt zwei Varianten:

- *Agglomerativ (bottom-up):* man beginnt unten und verbindet jeweils die ähnlichsten Objekte.
- *Divisiv (top-down):* man beginnt oben und teilt jeweils die unterschiedlichsten Objekte in einzelne Cluster auf.

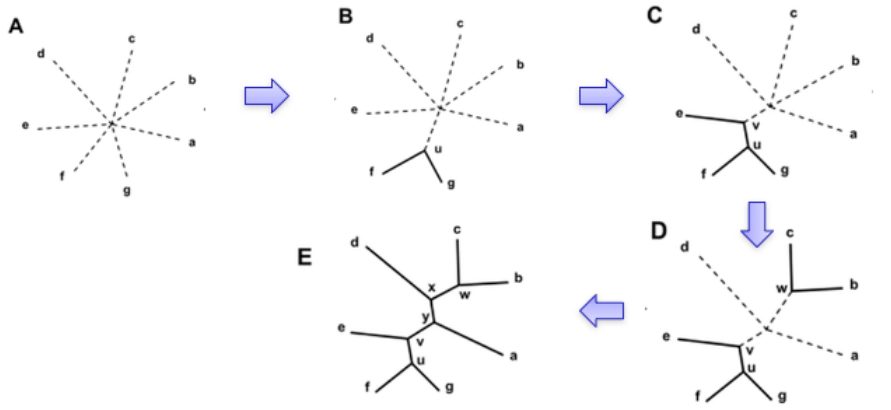
Jain et al., 1999

Hastie (The Elements of Statistical Learning), Hein ML Skript

Der Neighbor-joining Algorithmus gehört zu den **agglomerativen Clustering-Methoden**.

Neighbor-joining Algorithmus - Überblick

- Anfängliche Baumkonfiguration: Stern (A)
- Wähle bei jedem Schritt die zwei Taxa mit minimaler genetischer Distanz und füge eine Verzweigung zwischen ihnen ein -> neue Knoten *u, v, w*
- Berechne genetische Distanzen bezüglich der neuen Knoten
- Fahre fort bis alle Taxa eingefügt sind und die Stern-Struktur verschwunden ist



wikipedia.org

Diese Folie zeigt einen Überblick über die verschiedenen Schritte des Algorithmus. Bei jedem Schritt werden die zwei nächsten Nachbarn miteinander verbunden und ein Vorläuferknoten konstruiert. In B ist das der neue Knoten *u* für die Kind-Knoten *f* und *g*. Im Anschluss betrachtet man dann die Abstände zwischen *a, b, c, d, e* und dem neuen Knoten *u*. *f* und *g* werden nicht mehr betrachtet. In C werden *e* und *v* durch den neuen Knoten v miteinander verbunden.

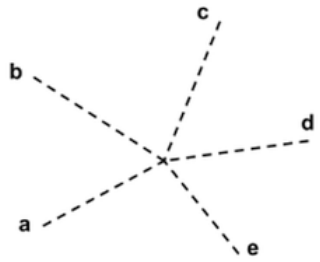
Wir müssen also in jedem Schritt berechnen, welche beiden Knoten die geringste Distanz zueinander haben. Dann müssen wir die Position des neuen Knoten berechnen.

Neighbor-joining: Beispiel

n Taxa: *a* bis *e*

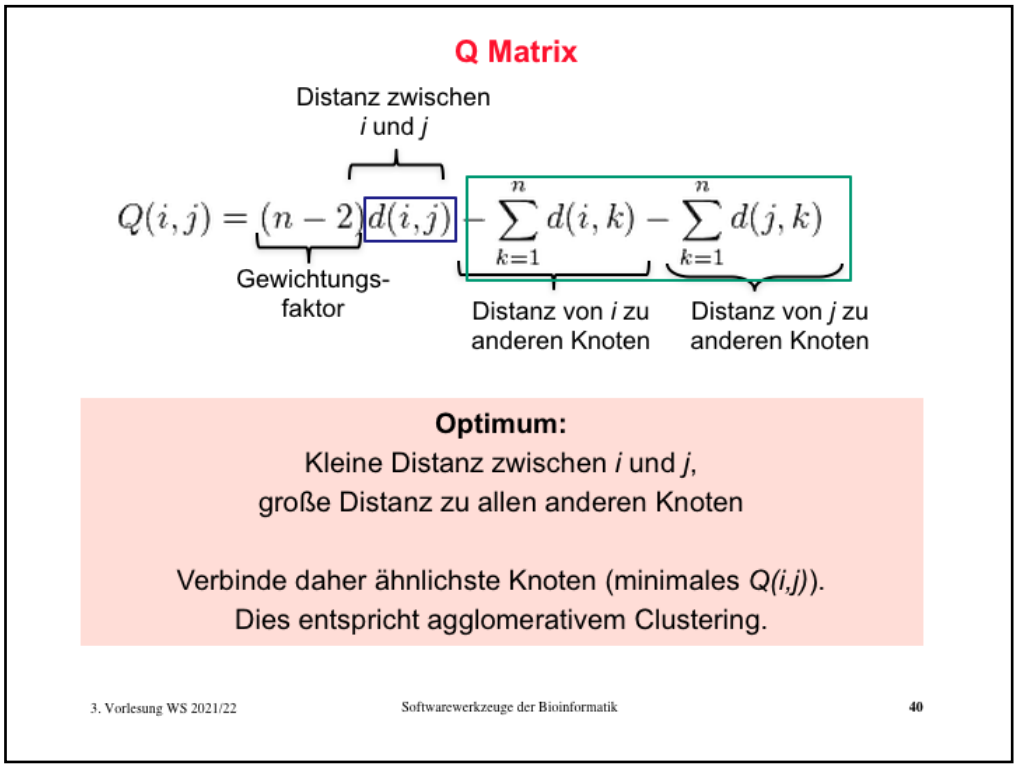
Distanzmatrix (symmetrisch):

	a	*b*	*c*	*d*	*e*
a	0	5	9	9	8
b		0	10	10	9
c			0	8	7
d				0	3
e					0

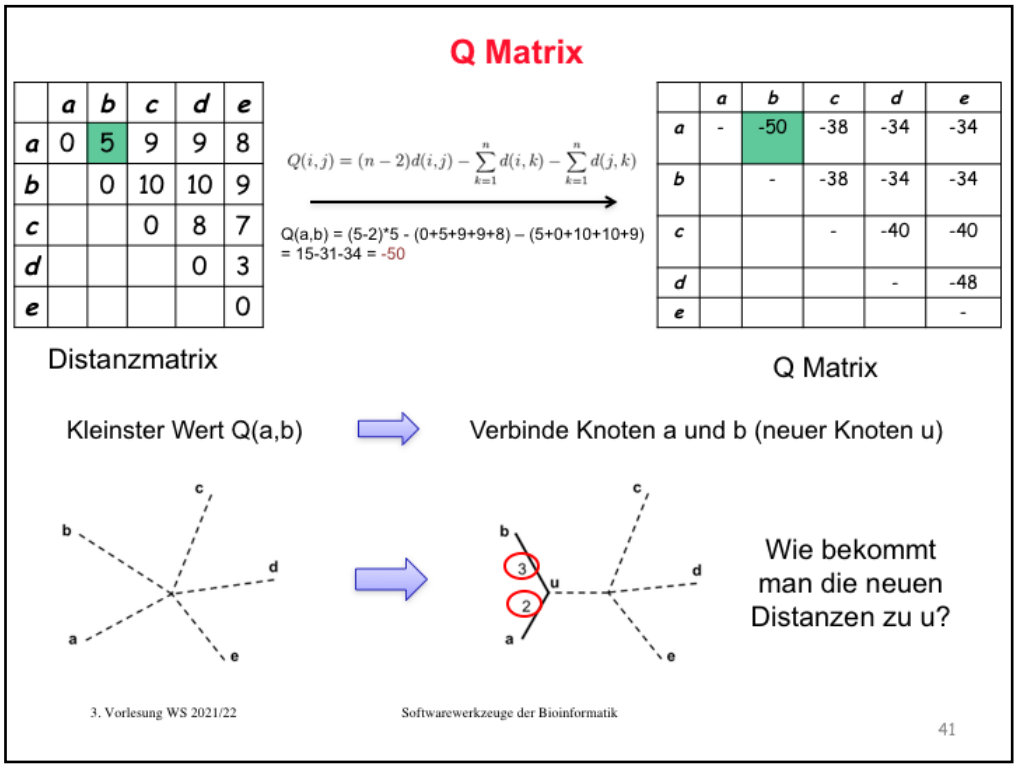


Anfängliche Stern-Struktur

Dies ist die Ausgangssituation: in diesem vereinfachten Beispiel ist eine Distanzmatrix zwischen 5 Knoten vorgegeben. Im Folgendem werden wir zwischen den Beispielen mit 5 und 7 Knoten hin- und herspringen. Es ergibt sich jeweils im Kontext, welcher Fall gemeint ist.



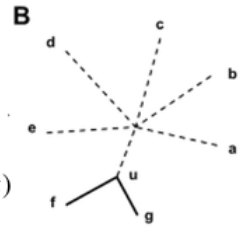
Aus der Distanzmatrix berechnen wir, welche Knoten am nächsten zueinander sind. Dabei betrachten wir ihre Distanz relativ zu ihren mittleren Distanzen zu den anderen Knoten (hintere zwei Summenterme). Wir könnten nun entweder die Summe der Distanzen durch *n* dividieren. Alternativ multiplizieren wir einfach die Distanz der beiden Knoten mit (*n*-2). Das Knotenpaar *i, j* mit minimalem *Q(i,j)* liegt am nächsten beieinander.



Hier ist die Q-Matrix gezeigt, die sich für die gegebene Distanzmatrix ergibt. Für das Knotenpaar *(a,b)* wird die Berechnung explizit nachvollzogen. Es gibt zu Beginn $n = 5$ Knoten. In den späteren Schritten gibt es entsprechend weniger Knoten. Das Paar *(a,b)* hat den minimalen Q-Wert. Deshalb werden *a* und *b* miteinander verbunden.

Berechne Distanz zu u

Falls *f* und *g* dieselbe Distanz zu allen anderen Knoten hätten, würde man *u* in die Mitte von *f* und *g* setzen: $\frac{d(f,g)}{2}$



Dies ist meist nicht der Fall.
Die mittleren Distanzen sind: $r(i) = \frac{1}{(n-2)} \sum_{k=1}^{n} d(i,k)$

Berechne damit die Position von *u*:

$$\delta(f,u) = \frac{d(f,g) + r(f) - r(g)}{2}$$

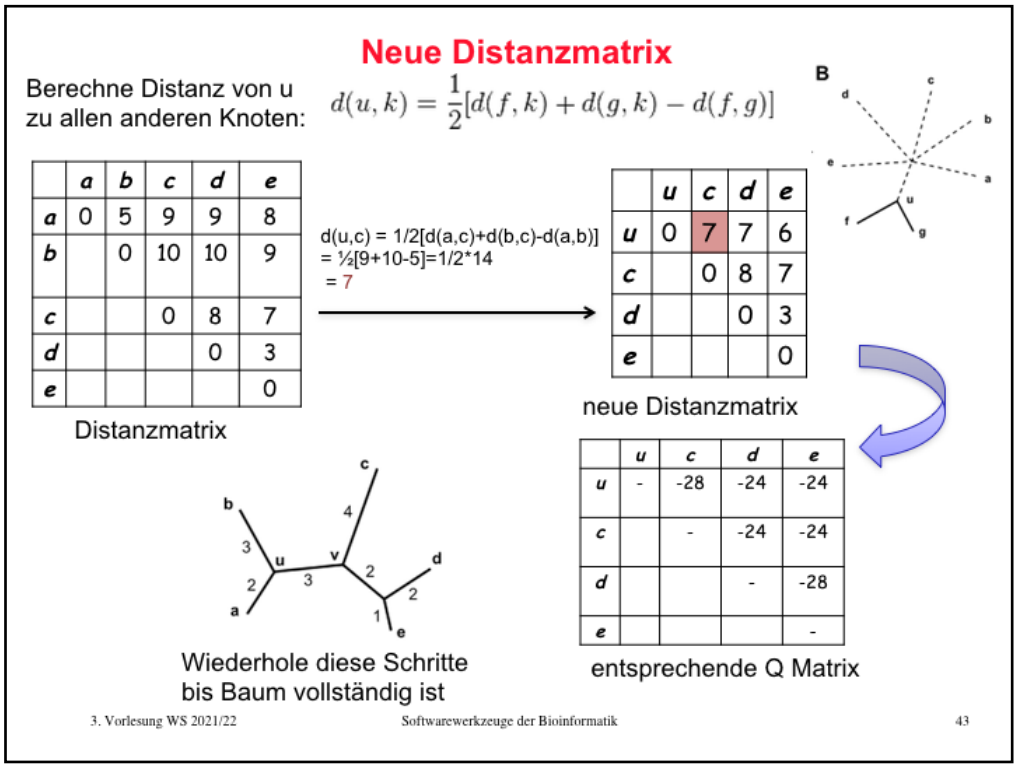
$$\delta(f,u) = \frac{1}{2} d(f,g) + \underbrace{\frac{1}{2(n-2)} \left[\sum_{k=1}^{n} d(f,k) - \sum_{k=1}^{n} d(g,k) \right]}$$

Mittlere Distanz zu allen anderen taxa (*r(f)-r(g)*)

Astlängen von a und b:
- δ(a,u) = 2
- δ(b,u) = 3

Der Abstand von *(f,g)* ist aus der Distanzmatrix bekannt. Im einfachsten Fall könnte man *u* in die Mitte von *(f,g)* setzen. Dann wären aber vermutlich die Abstände von den anderen Knoten nicht mehr korrekt.

Deshalb betrachten wir den mittleren Abstand von *f* zu allen Knoten und subtrahieren den mittleren Abstand von *g* zu allen Knoten. Wenn die Differenz positiv ist, liegt *g* näher an den anderen Knoten. Die Differenz ergibt somit, um welchen Betrag der neue Knoten *u* aus der Mitte heraus verschoben wird.



	a	b	c	d	e
a	0	5	9	9	8
b		0	10	10	9
c			0	8	7
d				0	3
e					0

	u	c	d	e
u	0	7	7	6
c		0	8	7
d			0	3
e				0

	u	c	d	e
u	-	-28	-24	-24
c		-	-24	-24
d			-	-28
e				-

Nachdem die neue Position von *u* gefunden wurde, muss dessen Abstand zu den anderen Knoten neu berechnet werden. Dies ergibt sich aus den Abständen seiner Kindknoten zu den anderen Knoten verringert um die Hälfte ihrer Distanz. Die alten Knoten werden nun aus der Distanzmatrix gelöscht und durch *u* ersetzt. Die Abstände zwischen den übrigen Knoten *c*, *d* und *e* bleiben erhalten. Ihre Abstände zu dem neuen Knoten *u* werden ergänzt. Daraus kann man wieder die *Q*-Matrix berechnen usw.

neighbor-joining Methode zusammengefasst

(1) Berechne für jedes Blatt $u_i = \sum_{j \neq i}^{n} \frac{D_{ij}}{n-2}$

(2) Wähle *i* und *j* sodass $D_{ij} - u_i - u_j$ minimal ist.

(3) Verbinde *i* und *j*. Berechne die Astlängen von *i* zum neuen Knoten (v_i) und vom *j* zum neuen Knoten (v_j) als

$$v_i = \frac{1}{2} D_{ij} + \frac{1}{2}\left(u_i - u_j\right)$$

$$v_j = \frac{1}{2} D_{ij} + \frac{1}{2}\left(u_j - u_i\right)$$

(4) Berechne den Abstand zwischen dem neuen Knoten (*ij*) und den übrigen Blättern als

$$D_{(ij)k} = \frac{D_{ik} + D_{jk} - D_{ij}}{2}$$

(5) Lösche die Blätter *i* und *j* aus den Listen und ersetze sie durch den neuen Knoten, (*ij*), der nun als neues Blatt behandelt wird.

(6) Falls mehr als 2 Knoten übrig bleiben, gehe nach Schritt (1) zurück. Andernfalls verbinde die zwei verbleibenden Knoten (z.B. *l* und *m*) durch einen Ast der Länge D_{lm}.

Dies ist eine Zusammenfassung aller Schritte des neighbor-joining-Algorithmus. Sie entspricht dem, was wir gerade besprochen haben.

Zusammenfassung

Multiple Sequenzalignments geben sehr wertvolle Einblicke in **Struktur und Funktion** von **Proteinfamilien**.

Globale dynamische Programmierung ist viel zu aufwändig.

Man benötigt heuristische Verfahren.

ClustalW: progressives Alignment, geleitet durch biologische Intuition; aber langsame Laufzeit.

Es gibt nun viel schnelle Verfahren z.B. MAFFT bzw. ClustalOmega.

Die Rekonstruktion von phylogenetischen Bäumen
beruht auf multiplen Sequenzalignments.

Die abgeleitete Phylogenie beruht stets auf Annahmen darüber,
wie Evolution abläuft (z.B. minimale Parsimonie).

In V3 haben wir uns zunächst mit der biologischen Relevanz von multiplen Sequenzalignments beschäftigt. Diese geben deutlich mehr Einblick in die Proteinstruktur und –funktion. Im Anschluss haben wir 2 Software-Tools besprochen, Clustal und MAFFT, mit denen man MSAs berechnen kann. In der zweiten Vorlesungshälfte haben wir uns mit Phylogenien beschäftigt. Der Umgang mit Sankoff-Algorithmus und Neighbor-Joining-Algorithmus ist **klausurrelevant**.