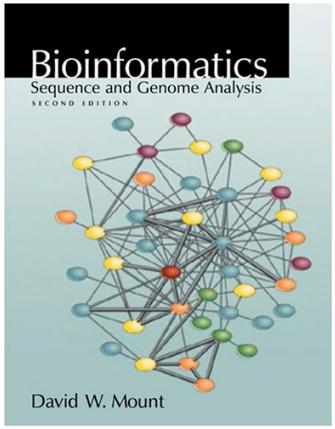
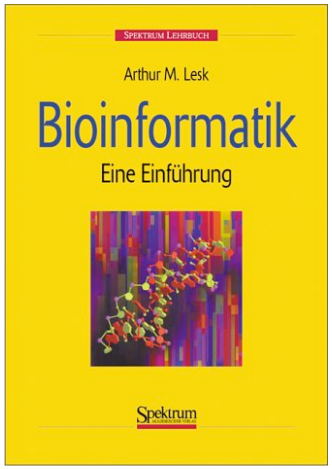
V3 - Multiples Sequenz Alignment und Phylogenie

Literatur: Kapitel 4 in Buch von David Mount



Thioredoxin-Beispiel heute aus Buch von Arthur Lesk



Leitfragen für V3

Frage1: 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 und leite daraus phylogenetische Stammbäume ab (zweiter Teil der Vorlesung heute).

Methode: (1) suche homologe Proteine in verschiedenen Organismen (BLAST bzw. Psiblast) (2) führe multiples Sequenzalignment durch (erster Teil)

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

Leitfragen für V3

Frage2: Können wir aus den evolutionären Veränderungen in einer Proteinsequenz etwas über die Struktur und Funktion des Proteins lernen? (erster Teil)

Ansatz : führe multiples Sequenzalignment durch (erster Teil der Vorlesung)

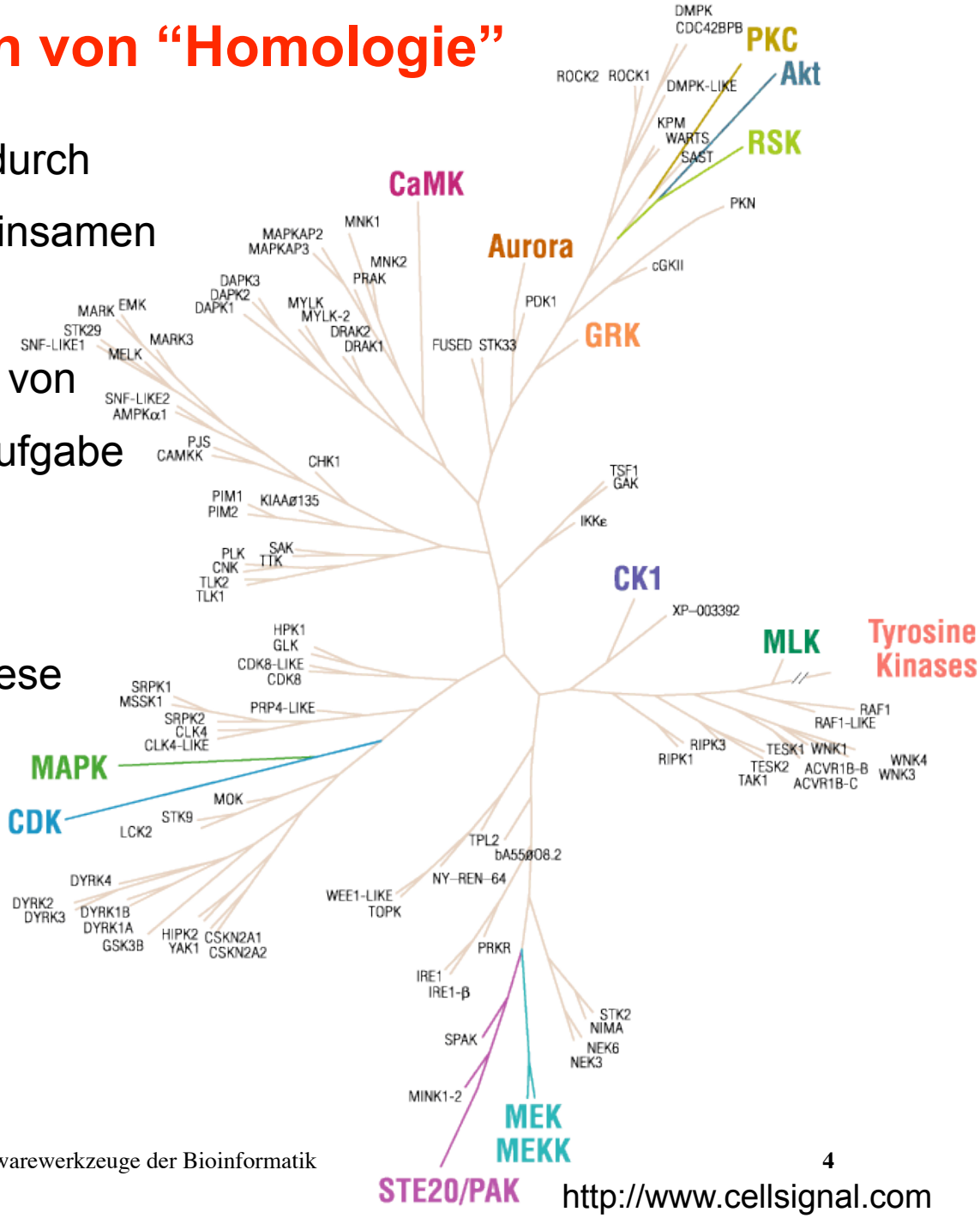
Exkurs: Evolution von Autos

- Welche Teile entsprechen dem aktiven Zentrum eines Proteins?
- Wird auch die Karosserie von Autos an Umgebungsbedingungen angepasst? (wo in Europa gibt es am meisten Cabrios?)
- Was entspricht dem Prozess der Proteinfaltung?
- Welchem Teil des Proteins entsprechen die Autotüren?

Definition von “Homologie”

- **Homologie**: **Ähnlichkeit**, 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.



http://www.cellsignal.com

Alignments können einfach oder schwer sein

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

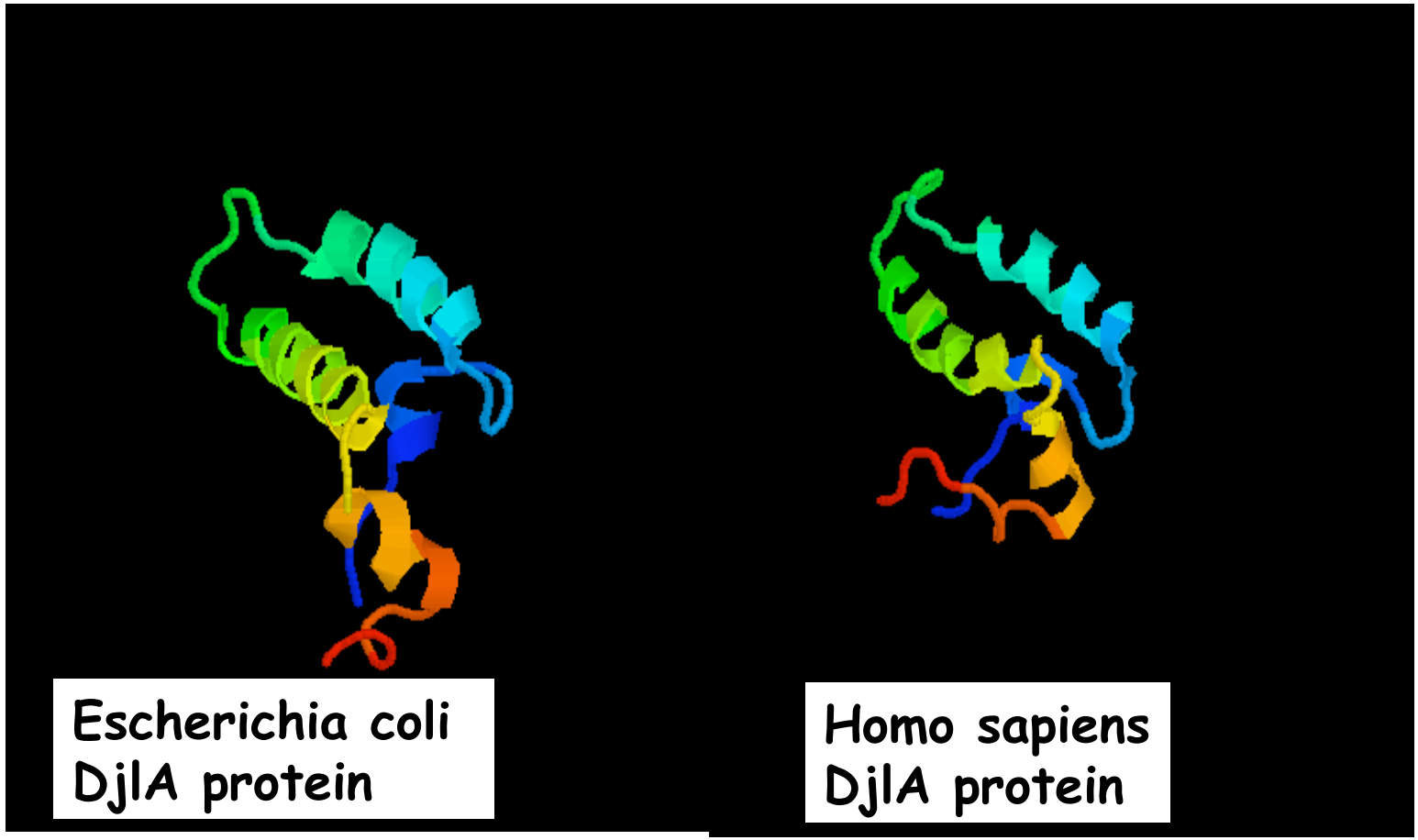
Einfach

```
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?

Protein-Alignment kann durch tertiäre Strukturinformationen geführt werden

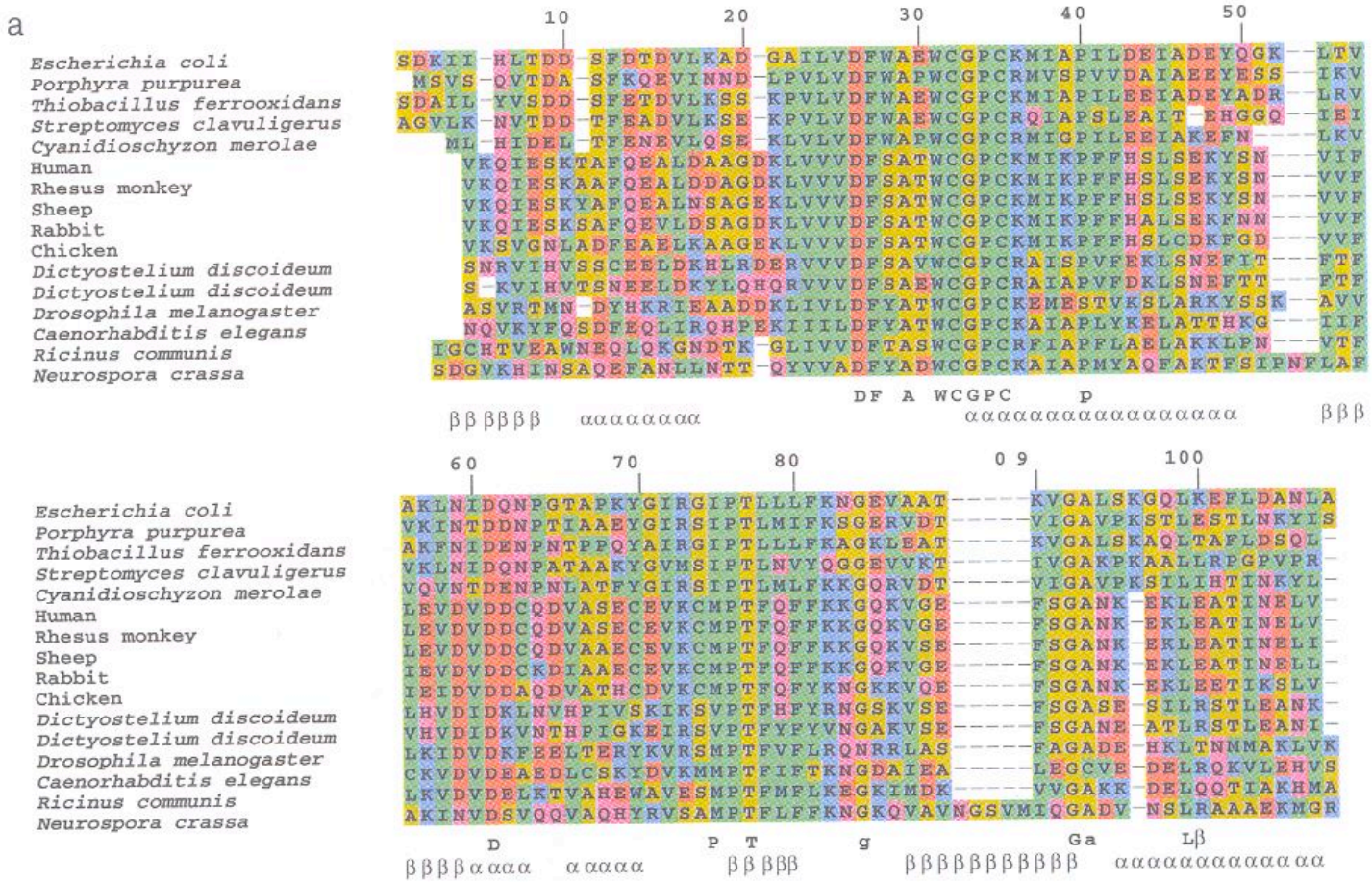


Gaps eines Alignments sollten vorwiegend in Loops liegen, nicht in Sekundär-struktur-elementen.

nur so kann man letztlich bewerten, ob ein Sequenzalignment korrekt ist. Beweisen im strikten Sinne kann man dies nie.

MSA für Thioredoxin-Familie

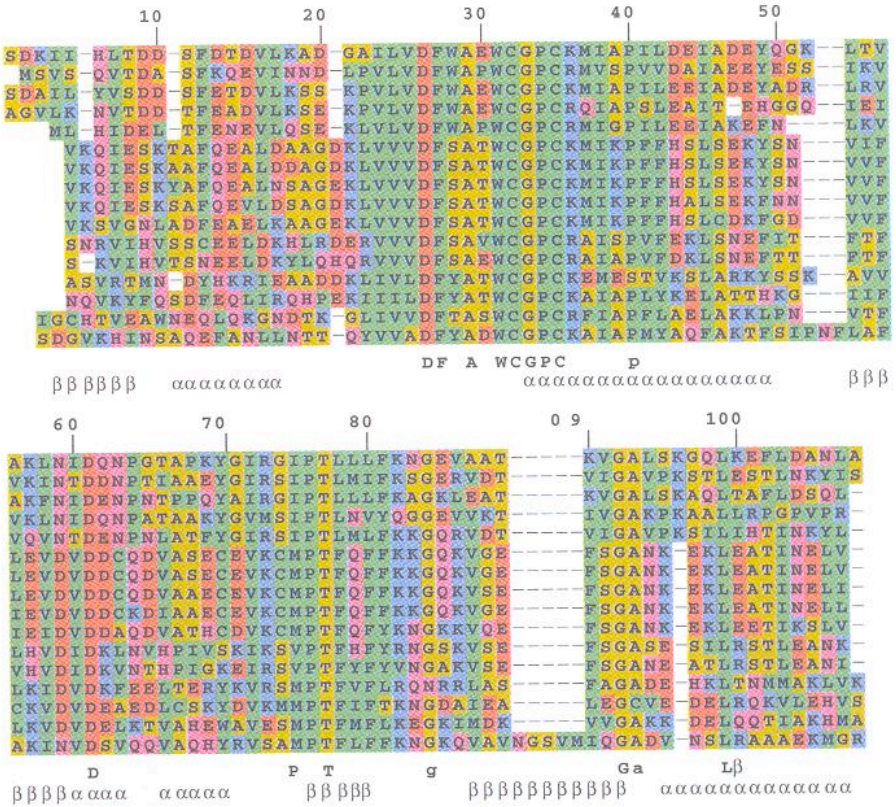
Farbe	Aminosäuretyp	Aminosäuren
gelb	klein, wenig polar	Gly, Ala, Ser, Thr
grün	hydrophob	Cys, Val, Ile, Leu
		Pro, Phe, Tyr, Met, Trp
violett	polar	Asn, Gln, His
rot	negativ geladen	Asp, Glu
blau	positiv geladen	Lys, Arg

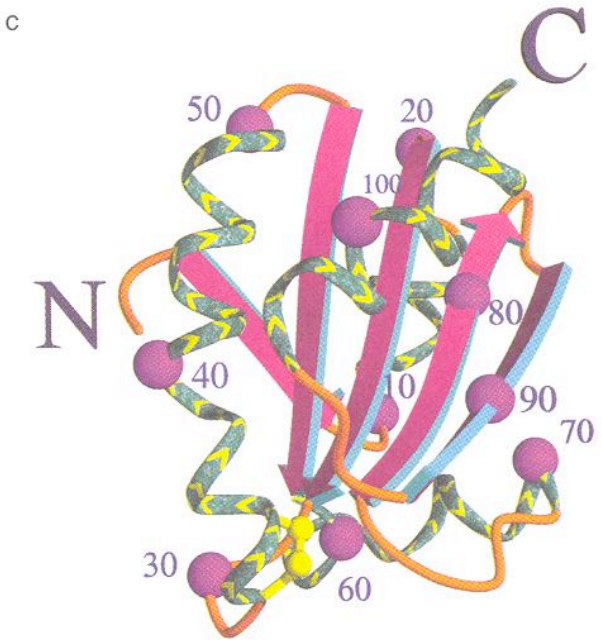


Infos aus MSA von Thioredoxin-Familie

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

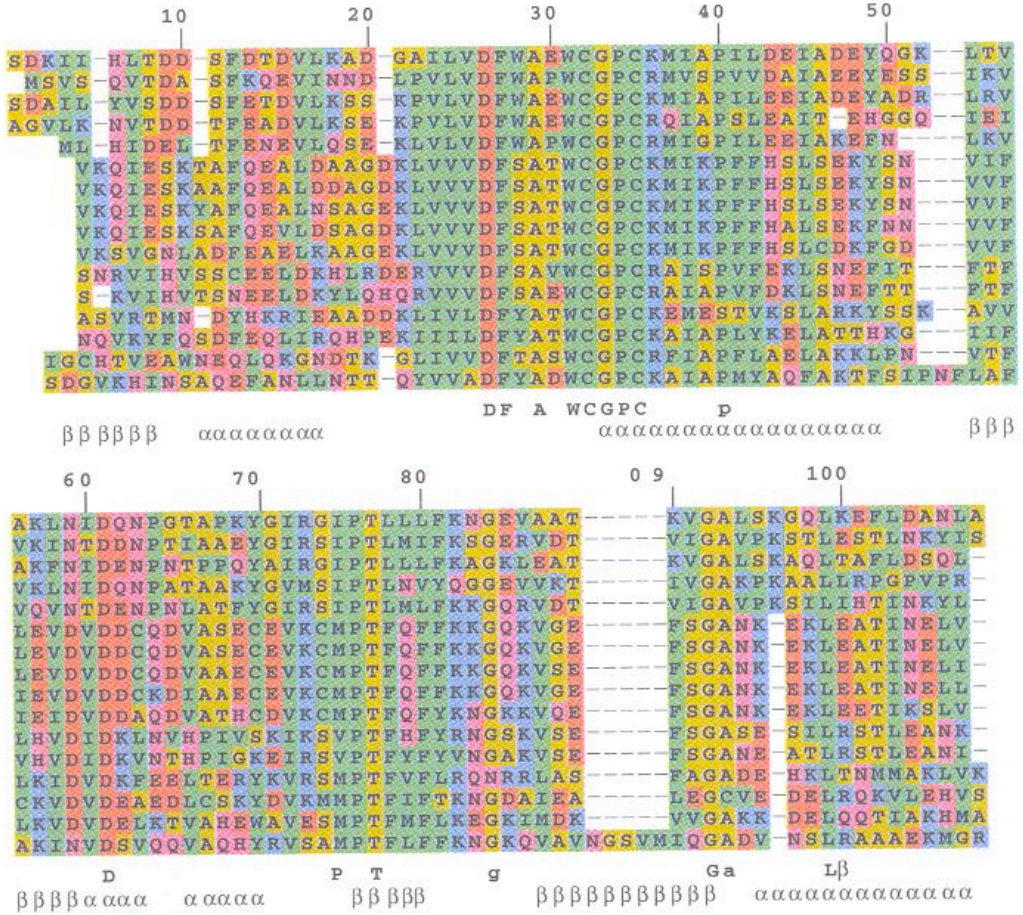
gemeinsamer Mechanismus: Reduktion von Disulfidbrücken in Proteinen

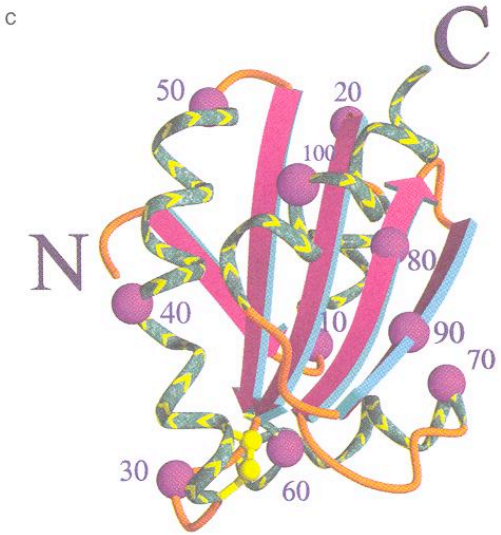




Infos aus MSA von Thioredoxin-Familie

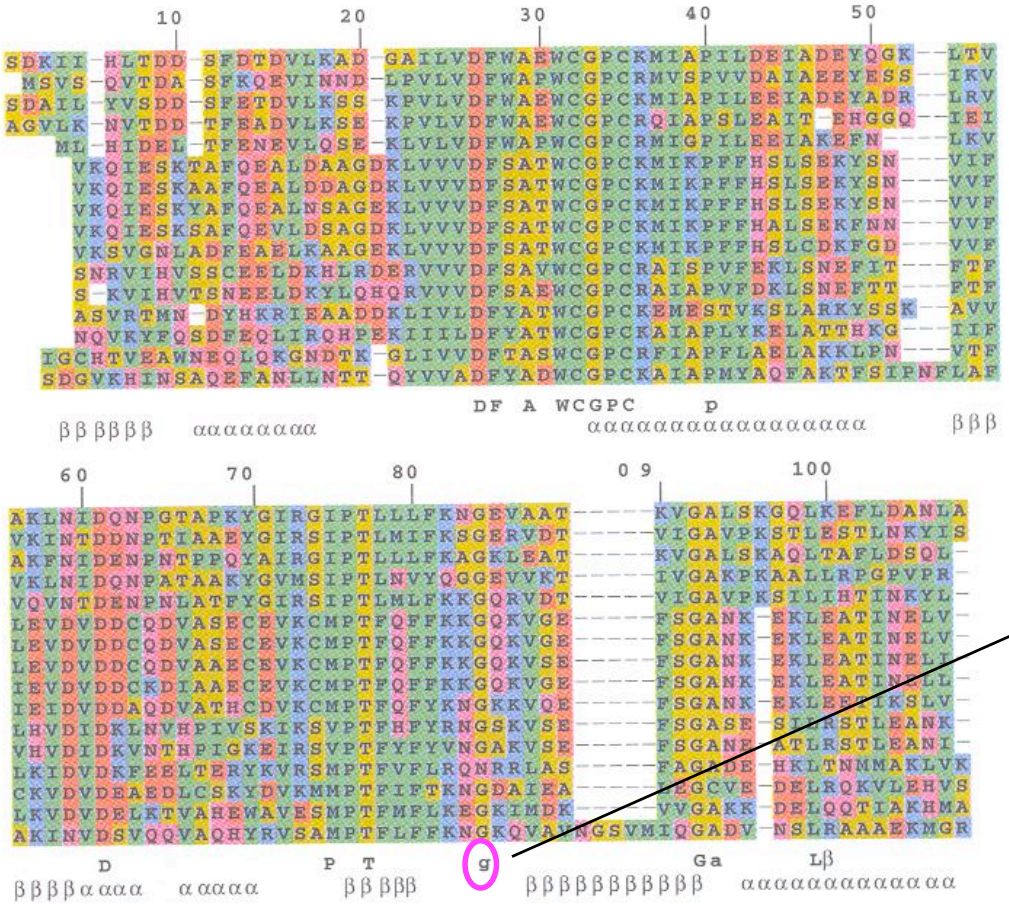
1) Die am stärksten konservierten Abschnitte entsprechen wahrscheinlich dem aktiven Zentrum. Disulfidbrücke zwischen Cys32 und Cys35 gehört zu dem konservierten WCGPC[K oder R] Motiv. Andere konservierte Sequenzabschnitte, z.B. Pro76Thr77 und Gly92Gly93 sind an der Substratbindung beteiligt.

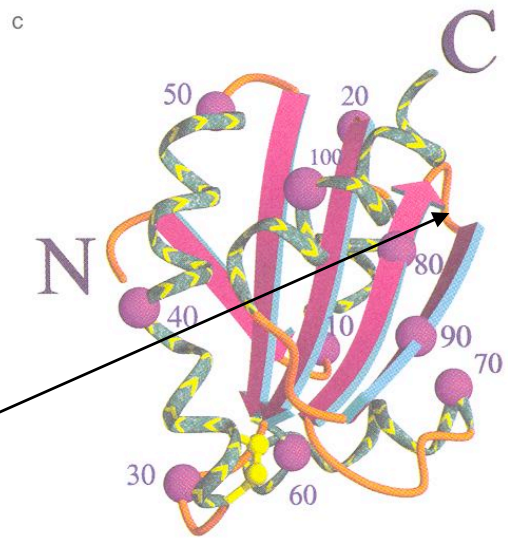




Infos aus MSA von Thioredoxin-Familie

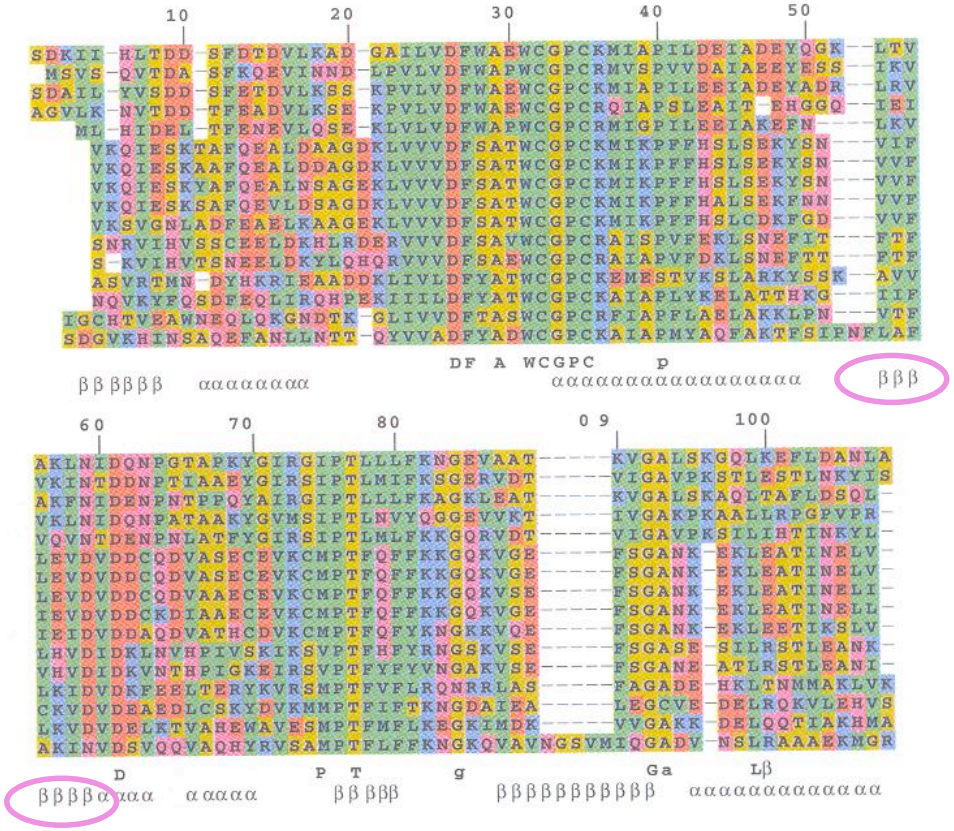
2) Abschnitte mit vielen Insertionen und Deletionen entsprechen vermutlich Schleifen an der Oberfläche. Eine Position mit einem konservierten Gly oder Pro lässt auf eine Wendung der Kette (‚turn') schließen.

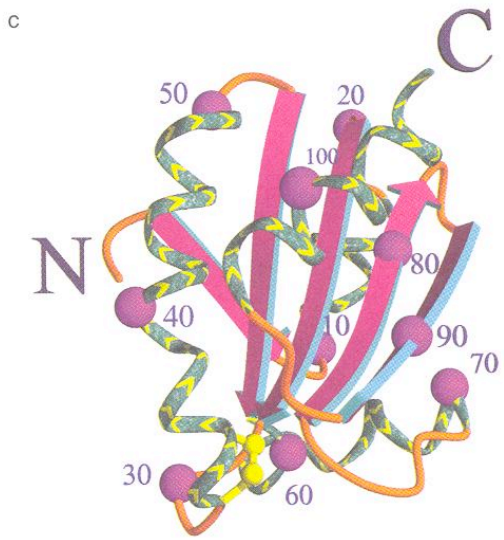




Infos aus MSA von Thioredoxin-Familie

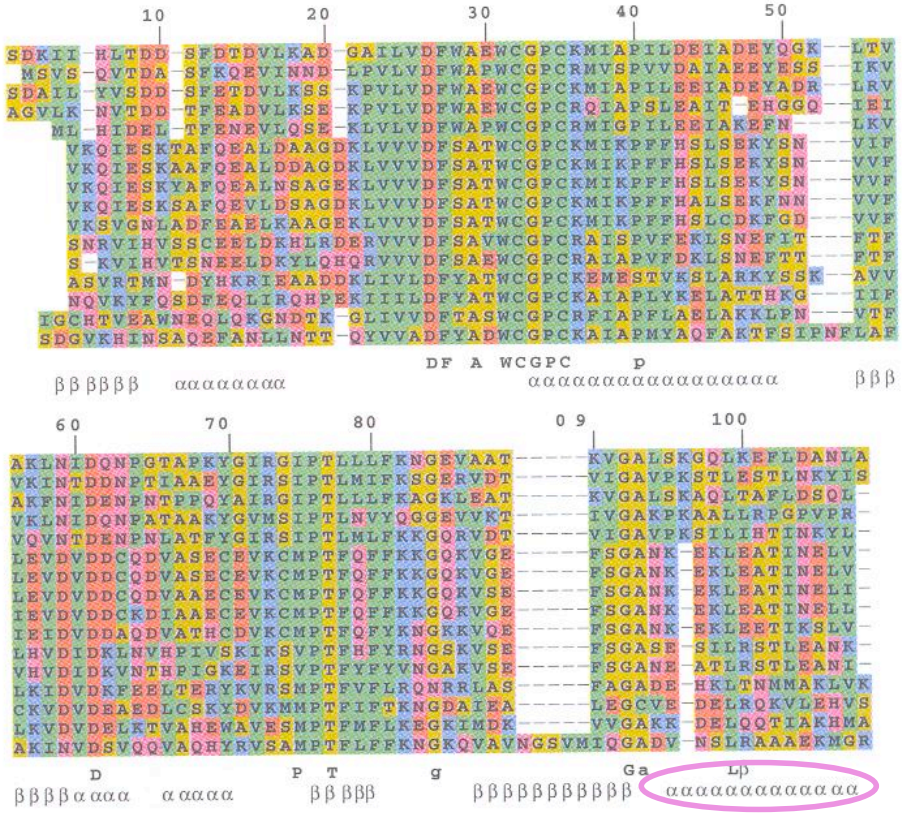
3) Ein konserviertes Muster hydrophober Bausteine mit dem Abstand 2 (d.h., an jeder zweiten Position), bei dem die dazwischen liegenden Bausteine vielfältiger sind und auch hydrophil sein können, lässt auf ein β-Faltblatt an der Moleküloberfläche schließen.

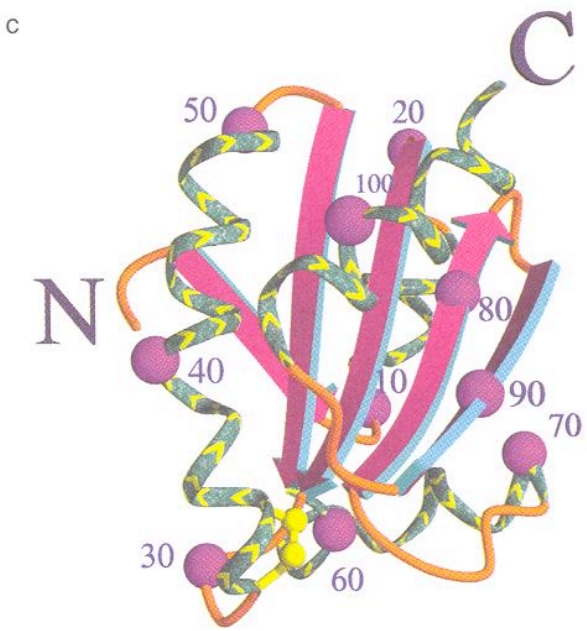




Infos aus MSA von Thioredoxin-Familie

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





Automatisches multiples Sequenzalignment

Hier gibt es vor allem folgende 2 Vorgehensweisen:

- **Dynamische Programmierung**
  - liefert garantiert das optimale Alignment!
  - aber: betrachte 2 Proteinsequenzen von 100 Aminosäuren Länge.

    wenn es 100^2 Sekunden dauert, diese beiden Sequenzen erschöpfend zu alignieren, dann wird es

    100^3 Sekunden dauern um 3 Sequenzen zu alignieren,

    100^4 Sekunden für 4 Sequenzen und

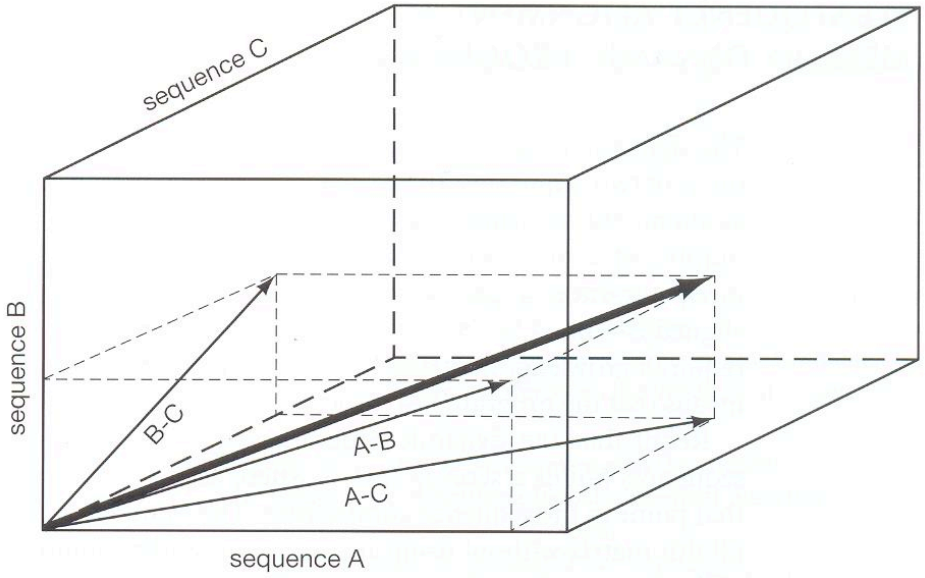
    1.90258×10^{34} Jahre für 20 Sequenzen.

- **Progressives Alignment**

dynamische Programmierung mit MSA Programm

berechne zunächst paarweise Alignments

für 3 Sequenzen wird Würfel aufgespannt:



D.h. dynamische Programmierung hat nun Komplexität n1 * n2 * n3
mit den Sequenzlängen n1, n2, n3.

Sehr aufwändig! Versuche, Suchraum einzuschränken und nur einen kleinen Teil des Würfels abzusuchen.

Progressives Alignment

- wurde von Feng & Doolittle 1987 vorgestellt
- ist eine **heuristische** Methode.

  Daher ist nicht garantiert, das “optimale” Alignment zu finden.

- benötigt (*n-1) + (n-2) + (n-3) ... (n-n+1)* paarweise Sequenzalignments als Ausgangspunkt.

- weitverbreitete Implementation in **Clustal** (Des Higgins)

- **ClustalW** ist eine neuere Version, in der Gewichte (**weights**) verwendet werden.

ClustalW- Paarweise Alignments

- Berechne alle möglichen paarweisen Alignments von Sequenzpaaren.
  Es gibt *(n-1)+(n-2)...(n-n+1)* Möglichkeiten.

- Berechne aus diesen isolierten paarweisen Alignments den “Abstand” zwischen jedem Sequenzpaar.

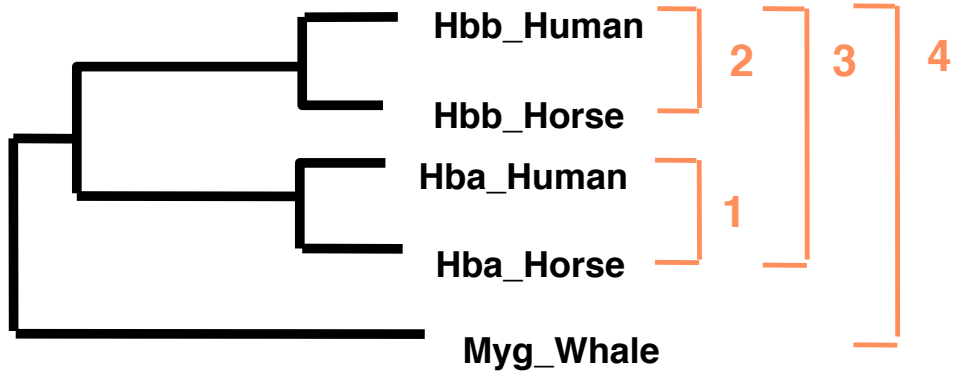
- Erstelle eine **Abstandsmatrix**.

- aus den paarweisen Distanzen wird ein **Nachbarschafts-Baum** erstellt

- Dieser Baum gibt die **Reihenfolge** an, in der das progressive Alignment ausgeführt werden wird.

Überblick der ClustalW Prozedur

Hbb_Human	1	-			
Hbb_Horse	2	.17	-		
Hba_Human	3	.59	.60	-	
Hba_Horse	4	.59	.59	.13	-
Myg_Whale	5	.77	.77	.75	.75	-



alpha-helices

```
1  PEEKSAVTALWGKVN--VDEVGG
2  GEEKAAVLALWDKVN--EEEVGG
3  PADKTNVKAAWGKVGAHAGEYGA
4  AADKTNVKAAWSKVGGHAGEYGA
5  EHEWQLVLHVWAKVEADVAGHGQ
```

2 3 4 1

CLUSTAL W

↓

Schnelle paarweise Alignments:
berechne Matrix der Abstände

↓

Nachbar-Verbindungs-
Baumdiagramm

↓

progressive Alignments
entsprechend dem
Baumdiagramm

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:

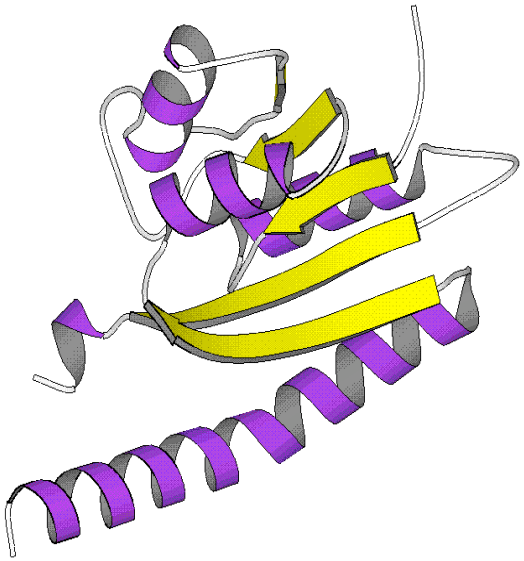
- Prozedur kann in ein lokales Minimum geraten.

  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.
- Zufälliges Alignment.

ClustalW- Besonderheiten

- Sollen all Sequenzen gleich behandelt werden?

  Obwohl manche Sequenzen eng verwandt und andere entfernt verwandt sind? ➔ Sequenzgewichtung



- Variable Substitutionsmatrizen

- Residuen-spezifische Gap-Penalties und verringerte Penalties in hydrophilen Regionen (externe Regionen von Proteinsequenzen), bevorzugt Gaps in Loops anstatt im Proteinkern.

- Positionen in frühen Alignments, an denen Gaps geöffnet wurden, erhalten lokal reduzierte Gap Penalties um in späteren Alignments Gaps an den gleichen Stellen zu bevorzugen

ClustalW- vom Benutzer festzulegende Parameter

- Zwei Parameter sind festzulegen (es gibt Default-Werte, aber man sollte sich bewusst sein, dass diese abgeändert werden können):

- Die **GOP- Gap Opening Penalty** ist aufzubringen um eine Lücke in einem Alignment zu erzeugen.

  Bevor irgendein Sequenzpaar aligniert wird, wird eine Tabelle von GOPs erstellt für jede Position der beiden Sequenzen.

  Die GOP werden positions-spezifisch behandelt und können über die Sequenzlänge variieren.

- Die **GEP- Gap Extension Penalty** ist aufzubringen um diese Lücke um eine Position zu verlängern.

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 *k* enthalten.

MSA mit MAFFT-Programm

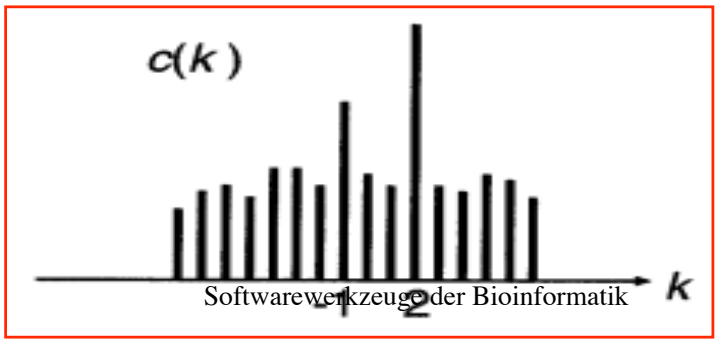
Berechne die Korrelation der beiden Vektoren v_1 , v_2 mit den Aminosäure-Volumina für jede mögliche Verschiebung k:

$$c_v(k) = \sum_{1 \le n \le N, 1 \le n+k \le M} \hat{v}_1(n)\hat{v}_2(n+k).$$

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

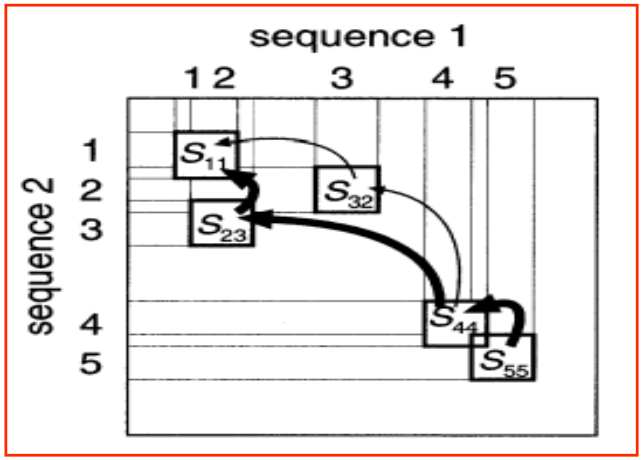
Bilde dann die Summe der beiden Korrelationen: $c(k) = c_v(k) + c_p(k).$

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



MSA mit MAFFT-Programm

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



Schritt 3: erstelle progressiv multiples Alignment:

- Schnelle Berechnung einer Distanzmatrix:

  → gruppiere 20 Aminosäuren in 6 physikochemische Gruppen

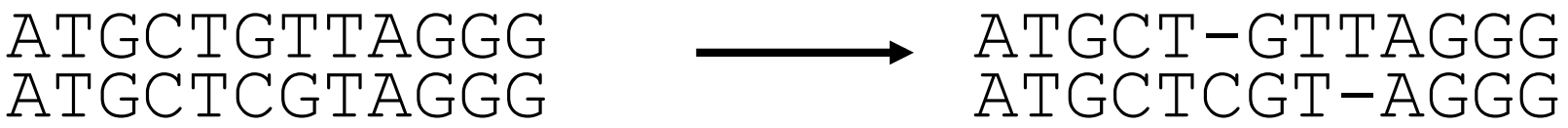
  → zähle 6-Tuples, die beide Sequenzen gemeinsam haben (vgl. Blast)

- 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

Alignment von Protein-kodierenden DNS-Sequenzen

- **Es macht wenig Sinn, proteinkodierende DNS-Abschnitte zu alignieren!**



Das Ergebnis kann sehr unplausibel sein und entspricht eventuell nicht dem biologischen Prozess.
Es ist viel sinnvoller, die Sequenzen in die entsprechenden Proteinsequenzen zu übersetzen, diese zu alignieren und dann in den DNS-Sequenzen an den Stellen Gaps einzufügen, an denen sie im Aminosäure-Alignment zu finden sind.

Zusammenfassung

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

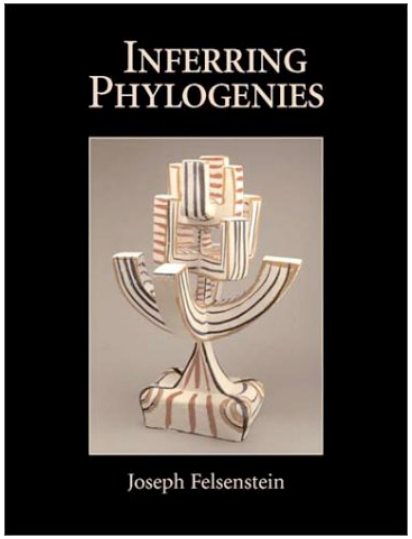
Sehr sensitive Methode ebenfalls: Hidden Markov Modelle (HMMer)

Multiples Sequenzalignment ist **nicht trivial**. Manuelle Nacharbeit kann in Einzelfällen das Alignment verbessern.

Multiples Sequenzalignment erlaubt **Denken** in **Proteinfamilien** und –**funktionen**.

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.

3 Hauptansätze für Phylogenien einzelner Gene

- **maximale Parsimonie**

- **Distanzmatrix**

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

Häufig verwendete **Programme**:
PHYLIP (phylogenetic inference package – J Felsenstein)
PAUP (phylogenetic analysis using parsimony – Sinauer Assoc

Parsimonie Methoden

Edwards & Cavalli-Sforza (1963):
derjenige evolutionäre Baum ist zu bevorzugen,
der „den minimalen Anteil an Evolution“ enthält.

Luca Cavalli-Sforza

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

(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.

Ein einfaches Beispiel

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

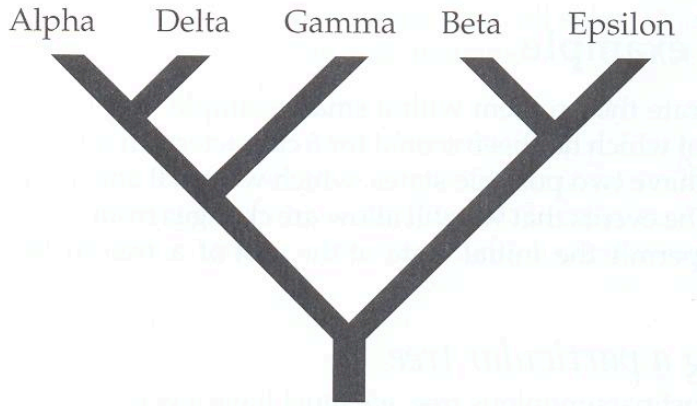
	Characters					
Species	1	2	3	4	5	6
Alpha	1	0	0	1	1	0
Beta	0	0	1	0	0	0
Gamma	1	1	0	0	0	0
Delta	1	1	0	1	1	1
Epsilon	0	0	1	1	1	0

Erlaubt seien Austausche 0 → 1 und 1 → 0.

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

Bewerte einen bestimmten Baum

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



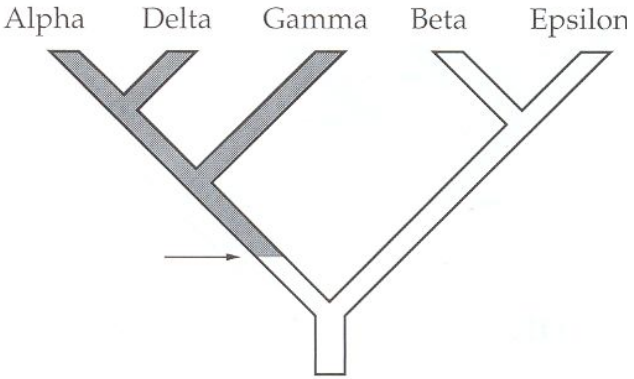
	Characters					
Species	1	2	3	4	5	6
Alpha	1	0	0	1	1	0
Beta	0	0	1	0	0	0
Gamma	1	1	0	0	0	0
Delta	1	1	0	1	1	1
Epsilon	0	0	1	1	1	0

Dieser Baum stelle die Phylogenie des ersten Buchstabens dar.

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

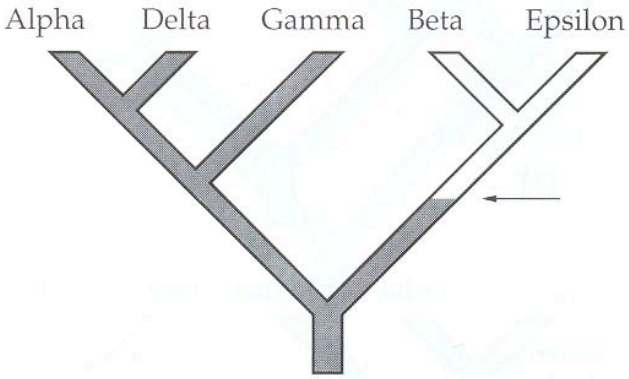


Figure 1.2: Alternative reconstructions of character 1 on the phylogeny of Figure 1.1. The white region of the tree is reconstructed as having state 0, the shaded region as having state 1. The two reconstructions each have one change of state. The changes of state are indicated by arrows.

Bewerte einen bestimmten Baum

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

Species	Characters					
	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

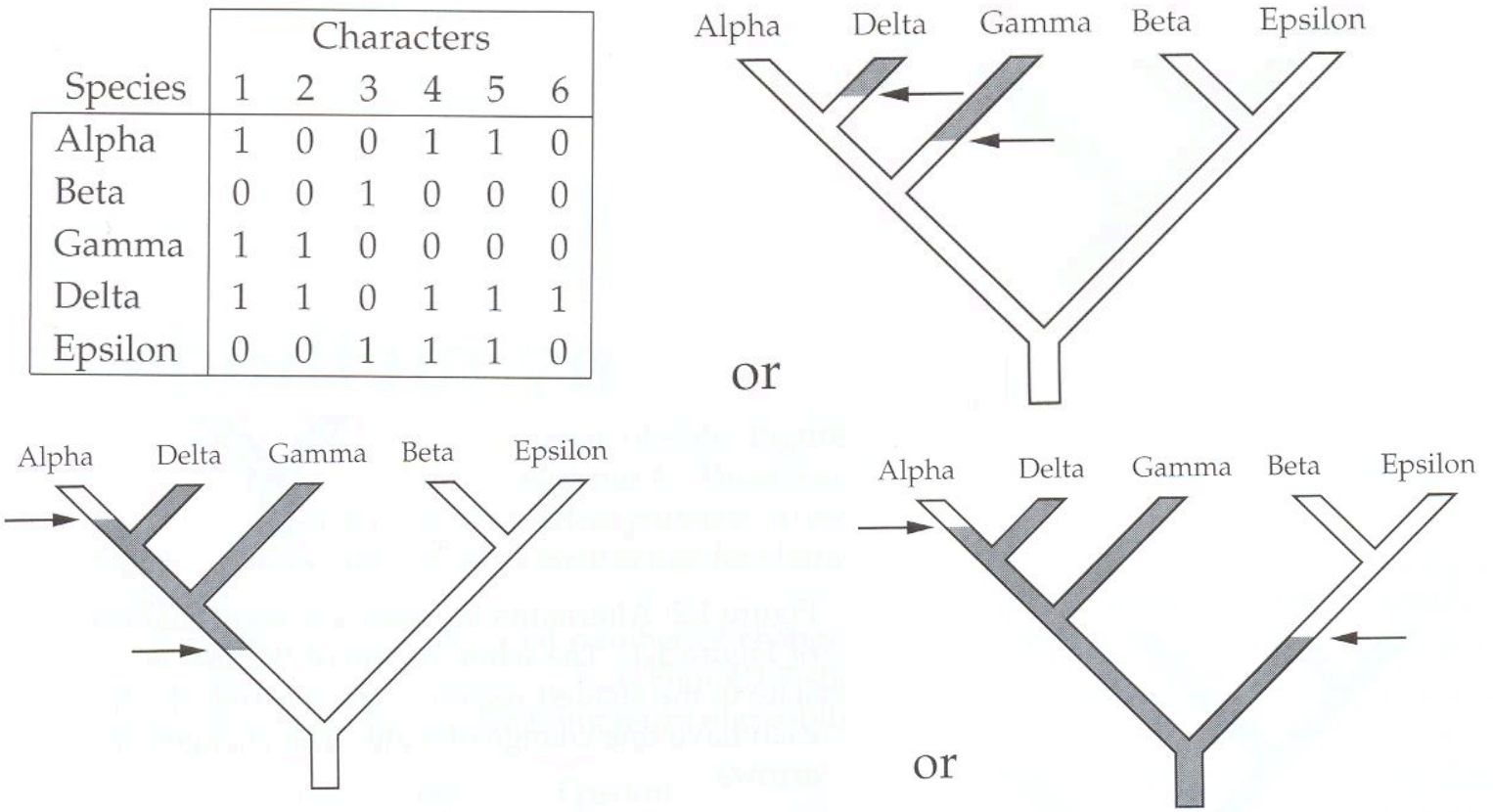


Figure 1.3: Reconstructions of character 2 on the phylogeny of Figure 1.1. The white regions have state 0, the shaded region state 1. The changes of state are indicated by arrows.

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

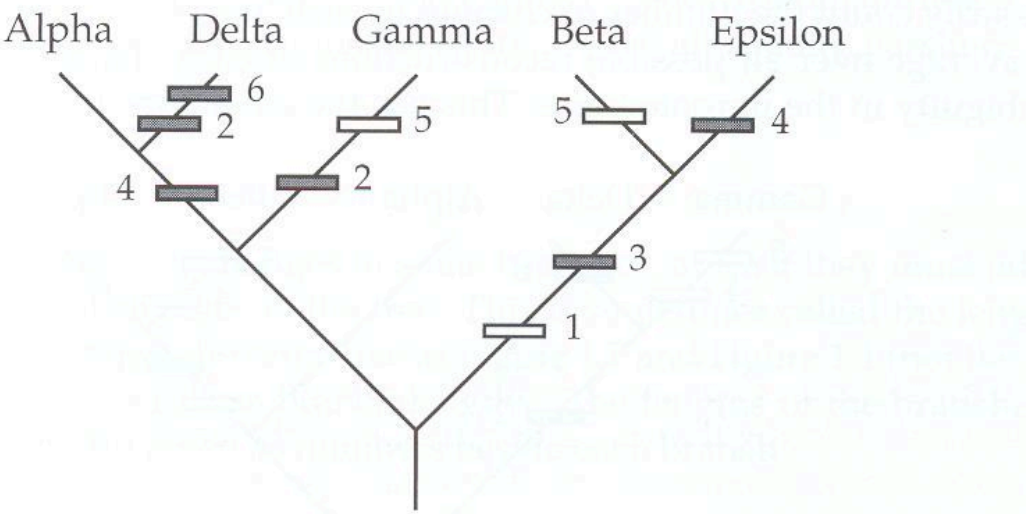


Figure 1.7: Reconstruction of all character changes on the phylogeny of Figure 1.1. The changes are shown as bars across the branches, with a number next to each indicating which character is changing. The shading of each box indicates which state is derived from that change.

Bewerte einen bestimmten Baum

Ein anderer Baum, der nur 8 Zustandsänderungen benötigt.

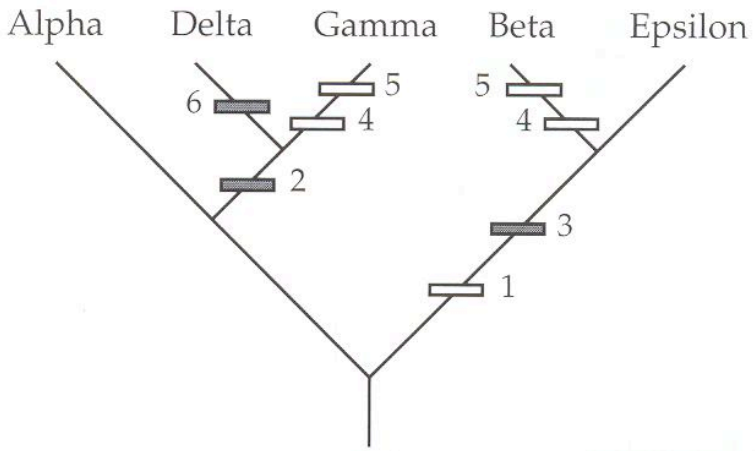


Figure 1.8: Reconstruction of all changes on the most parsimonious phylogeny for the data of Table 1.1. It requires only 8 changes of state. The changes are shown as bars across the branches, with a number next to each indicating which character is changing. The shading of each box indicates which state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

Die minimal Anzahl an Zustandsänderungen ist 6, da es 6 Buchstaben gibt, die jeweils 2 Zustände annehmen können.

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.

→ 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.

Zähle evolutionäre Zustandsänderungen als Modell für evolutionäre Kosten eines gegebenen Evolutionsbaums

Hierfür existieren zwei verwandte Algorithmen, 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 nach unten durch den Baum;
beim Erreichen der Blätter ist die Anzahl der Zustandsänderungen bekannt.

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

Sankoff Algorithmus

David Sankoff

Gesucht: Modell für Evolution einer Nukleotid-Position.

Konstruiere einen evolutionären Baum und wähle 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.

{C}	{A}	{C}	{A}	{G}
∞ 0 ∞ ∞	0 ∞ ∞ ∞	∞ 0 ∞ ∞	0 ∞ ∞ ∞	∞ ∞ 0 ∞
A	C	G	T	

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

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 $S_l(j)$.

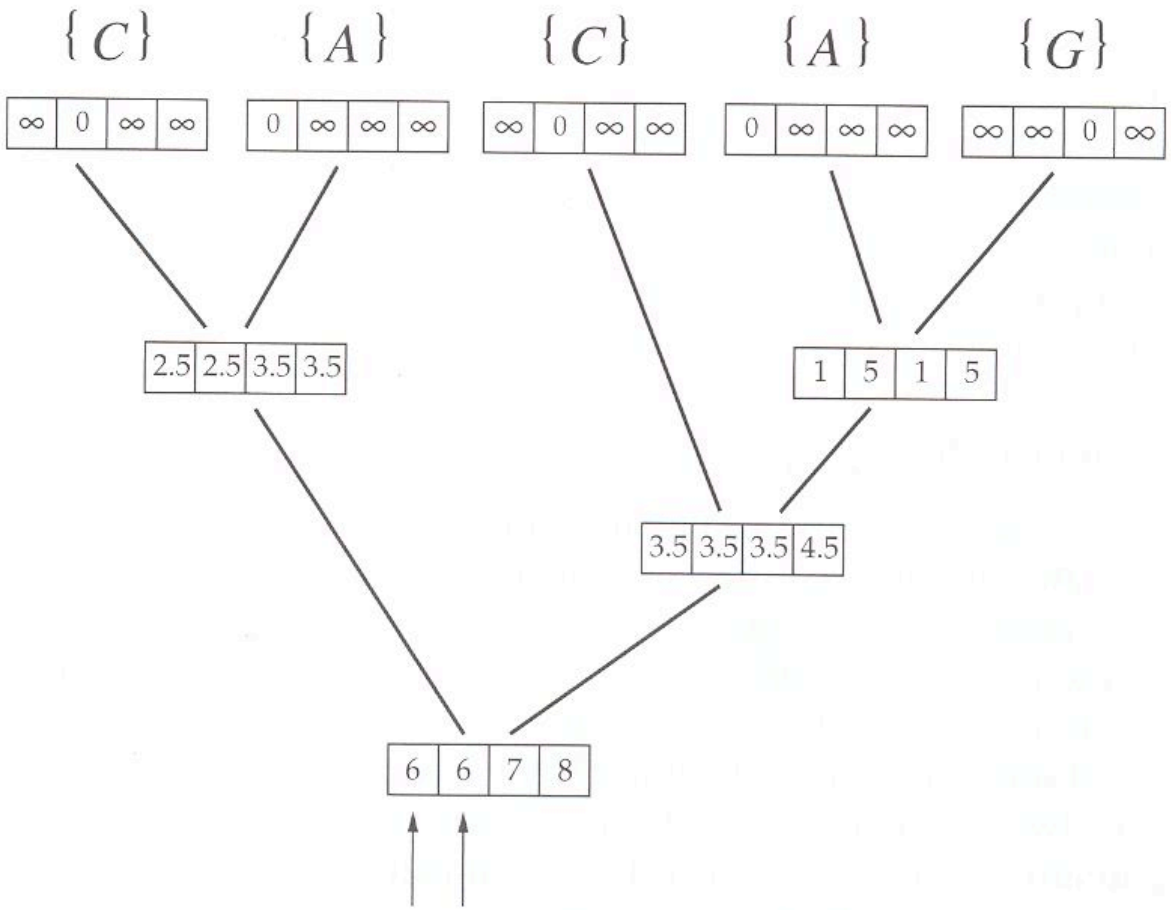
Wähle den Wert *j*, der diese Summe minimiert.

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: $S = \min_i S_0(i)$

Sankoff-Algorithmus



Cost matrix:

from \ to	A	C	G	T
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0

Der Vektor (6,6,7,8) an den Blättern besitzt ein Minimum von 6

= dies sind die minimalen Gesamtkosten dieses Baums für diesen Buchstaben.

Die Ur-Vorgängersequenz enthielt an dieser Position vermutlich „A“ oder „C“.

Konstruiere einen guten Baum: neighbor-joining Methode

durch Saitou und Nei (1987) eingeführt – der Algorithmus verwendet Clustering – eine molekulare Uhr wird nicht angenommen, aber das Modell minimaler Evolution.

„Modell minimaler Evolution“

wähle unter den möglichen Baumtopologien die mit minimaler Gesamtlänge der Äste.

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

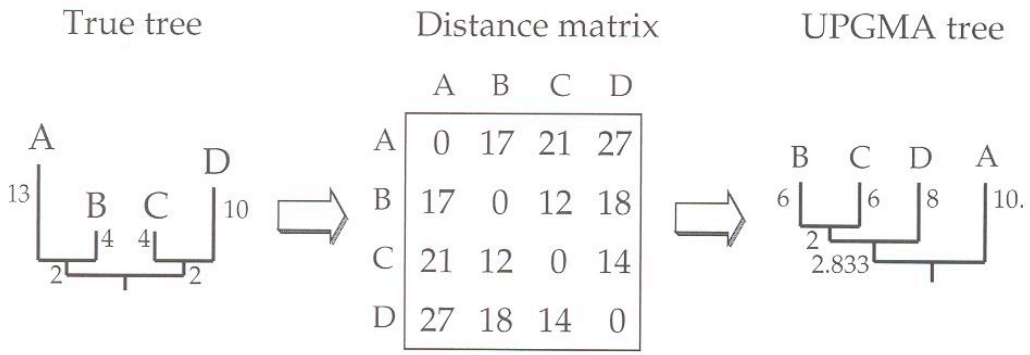


Figure 11.7: A four-species, nonclocklike tree and the expected data matrix it yields, when distances are the sums of branch lengths. The tree estimated by applying the UPGMA method to this distance matrix is shown—it does not have the correct tree topology. In both trees the branch lengths are proportional to the vertical length of the branches.

neighbor-joining Methode

(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}.

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: geleitet durch biologische Intuition; langsame Laufzeit.

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

Die Rekonstruktion von phylogenetische Bäumen beruht auf multiplen Sequenz-alignments.

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