

## Genomics bei Dr. Jung. 2 Sequenzalignment und Phylogenetische Analysen

In dieser VL machen wir weiter mit Sequenzanalysen.

### S.3. Sequenzvergleiche –Ziele

Was ist das Ziel, wenn man genomische Sequenz miteinander vergleicht? Prinzipiell gibt es zwei Oberziele. Entweder ich möchte eine **Funktionsvorhersage für Proteine** machen. Das bedeutet, man versucht aus der Sequenz ein bisschen auf Die Funktion des Proteins zu schließen. Das kann man, weil ähnliche Proteine auch eine ähnliche DNA-Struktur besitzen. Das zweite Ziel sind **phylogenetische Analysen**. Man untersucht also Ähnlichkeiten oder Unähnlichkeiten von Spezies, indem man die Genome und Gene miteinander vergleicht. Bei der Funktionsvorhersage gibt es das Paradigma, dass, wenn eine hohe Sequenzähnlichkeit auf DNA-Ebene vorliegt zwischen zwei Spezies, dann haben die entsprechenden Proteine auch eine ähnliche Funktion. Wenn eine hohe Sequenzähnlichkeit vorliegt, dann ist also bedingt eine Funktionsvorhersage des Proteins möglich. Wenn ich zum Beispiel Sequenzen von zwei Spezies habe und wenn ihre Sequenzen ganz ähnlich sind, dann hat auch das gebildete Protein bei beiden eine ähnliche Funktion, auch wenn die zwei Spezies nicht nah miteinander verwandt sind. Umgekehrt muss man aufpassen, wenn man nur die Funktion eines Proteins kennt. Wenn man die Funktion eines Proteins kennt, dann heißt es noch lange nicht, dass die dahintersteckende DNA-Sequenz dieselbe ist! Das liegt daran, dass sich einige Sachen im Laufe der Evolution konvergent entwickelt haben. Was jetzt den Vergleich von Spezies betrifft, so muss man sagen, dass man den Vergleich früher nicht auf Sequenz-Ebene gemacht hat, sondern man verglich den Phänotyp und damit die morphologischen Eigenschaften und Ähnlichkeiten. Ein klassisches Beispiel wären die Galapagos-Finken, die Charles Darwin klassifiziert hat. Je ähnlicher die Finken zueinander in ihrer Morphologie waren, desto enger zueinander wurden sie auf dem phylogenetischen Baum geclustert. Seit der Entdeckung der DNA nutzt man den genetischen Code und Sequenzvergleiche, um Spezies und Gene zu clustern.

### S.4 -6. Dotplot.

Abbildung 1: Seite 6. Definition zum Dotplot+ Beispiel (Siehe Erklärung im Text)

#### Dotplot

- Definition:
  - Sequenz A:  $a_1 \dots a_n$ , Sequenz B:  $b_1 \dots b_m$
  - $(n \times m)$ -Matrix **M**
  - $M[i,j] = 1$  falls  $a_i = b_j$   $i = 1, \dots, n$  und  $j = 1, \dots, m$
  - $M[i,j] = 0$  falls  $a_i \neq b_j$   $i = 1, \dots, n$  und  $j = 1, \dots, m$

	E	S	T	R	A	G	O	N
E	1	0	0	0	0	0	0	0
G	0	0	0	0	0	1	0	0
O	0	0	0	0	0	0	1	0
N	0	0	0	0	0	0	0	1

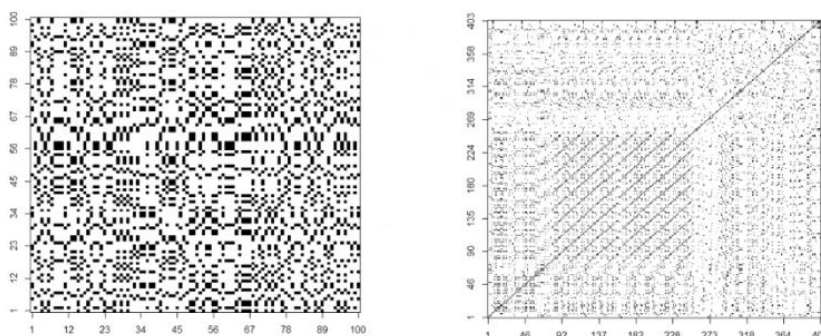
Wir gucken uns zu Beginn ein ganz einfaches graphisches Verfahren an, mit dem man zwei Sequenzen miteinander vergleichen kann. Das wäre der so genannte **Dotplot**. (Ab hier bezieht sich der Text auf Abbildung 1) Mal angenommen, wir haben zwei Sequenzen A und B, die wir miteinander vergleichen wollen. Sequenz A besteht aus Basenpaaren, die von 1 Basenpaar bis n-Basenpaare lang ist ( $a_1 \dots a_n$ ). Die Sequenz B besteht ebenfalls aus Basenpaaren, die bis zu m-Basenpaarlänge hat ( $b_1 \dots b_m$ ). So haben wir für die Basen und ihre jeweilige Sequenz die Buchstaben  $a_1$  bis  $a_n$  und  $b_1$  bis  $b_m$ . Das heißt, dass die zwei Sequenzen, die ich miteinander vergleichen will, auch unterschiedlich groß sein dürfen. Wenn ich die Sequenz von zwei Genen vergleichen möchte, dann ist es auch ziemlich wahrscheinlich, dass beide nicht dieselbe Länge haben. Dann würde ich mir eine **Matrix M** bauen, die n-Zeilen (Sequenz A) hat und m-Spalten (Sequenz B). Wir können es auch eine Tabelle nennen. Wenn jetzt in der i-ten Zeile und in der j-ten Spalte der gleiche Buchstabe in Sequenz A und B ist, dann trage ich eine 1 in die Matrix ein. Wenn ich für die i-te Zeile und j-ten Spalte unterschiedliche Basen in beiden Sequenzen habe, dann trage ich eine 0 ein. Also eigentlich ein relativ einfacher Logarithmus. Ein Logarithmus ist ja eine Vorgabe an den Computer, der ihm sagt, was er zu tun hat. Die Vorgabe beim Dotplot wäre jetzt eine 1 einzutragen, wenn die Sequenzen eine gemeinsame Base haben, und eine 0, wenn beide unterschiedliche Basen aufweisen. Ich habe zwei Beispiele aus dem Proteinhilfsbereich mitgenommen. Diese Sequenzen werden in der Natur so wahrscheinlich nicht vorkommen, aber die eine Sequenz wäre "Estragon" und die zweite Sequenz wäre "Egon". Wenn wir die eine Sequenz/ Wort in die Zeile eintragen und die andere Sequenz/ Wort in die Spalte, dann können wir uns anschauen, an welcher Stelle die zwei Sequenzen/Worte den gleichen Buchstaben besitzen. Das können wir dann in der Datenmatrix mit einer "1" für den gleichen Buchstaben und mit einer "0" für unterschiedliche Buchstaben notieren. Wenn ich die "1-sen" schwarz einfärbe, dann erhalte ich über die ganze Matrix verteilt schwarze Punkte oder "Dots" und deshalb nennt man dieses Verfahren auch Dotplot.

### s.7.Weitere Beispiele (für Dotplot).

Abbildung 2: Weitere Beispiele für Dotplot.

#### Dotplot

- Weitere Beispiele



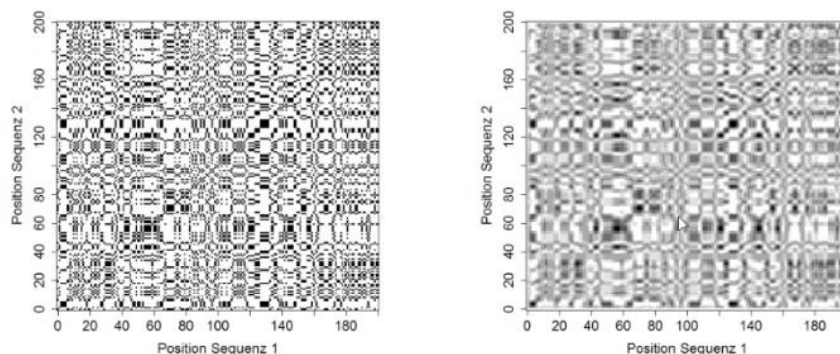
(Der folgende Text und Abbildung 2 sollen gemeinsam gelesen und verglichen werden, da der Text die Abbildung erklärt) Hier haben wir zwei Beispiele, die verdeutlichen sollen, wie ein Dotplot aussieht, wenn wir größere Sequenzen haben. Man sieht auch, dass Dotplot ein Verfahren für die

visuelle Analyse ist. Es ist deshalb eher subjektiv, aber es unterstützt einen doch beim Vergleich von Sequenzen. Wenn sich jetzt die zwei Graphiken anschauen auf Abbildung 2: welche Unterschiede sehen Sie? Auf der linken Graphik sehen wir kein strukturiertes Muster. Das deutet darauf hin, dass die beiden Sequenzen, die man auf dem linken Graphen verglichen hat, nicht viele Ähnlichkeiten zueinander haben. Auf dem rechten Graphen sieht man sofort die diagonale Linie als ein Muster. Dort, wo die diagonale Linie verläuft, hat man eine sehr hohe Sequenzähnlichkeit. Man sieht aber auch Punkte außerhalb der Diagonale und man sieht vor allem in der Mitte des Graphen, dass sich ein Block gebildet hat, wo rechts und links von der diagonalen Linie noch ein paar Linien verlaufen. Was haben diese Linien um die Diagonale herum zu bedeuten? Das wären Sequenzwiederholungen. Das heißt, wir haben in beiden Sequenzen repetitive Elemente. Wir haben also Motive, die sich immer wieder wiederholen und dadurch entstehen diese Linien. Das heißt, wenn Sie sowas im Dotplot sehen, dann wissen Sie, dass es repetitive Elemente gibt. Wenn Sie ganz genau hinschauen, sehen sie auf dem linken Graphen in der Hauptdiagonale noch einen kleinen Sprung (Diagonale ist nicht durchgängig verlaufend). Was könnte das bedeuten? Das zeigt uns die Gaps an. Das bedeutet, dass eine Insertion oder Deletion stattgefunden hat, weshalb eine Sequenz mehr Basen hat als die andere Sequenz. Gap bedeutet, dass ein Sequenzabteil an dieser Stelle ausgelassen wird. Kleinere Sprünge würde man im Dotplot nicht sehen. Wenn man einen Sprung sieht, dann deutet es auf größere Insertionen bzw. Deletionen in einer der beiden Sequenzen hin. Wir haben also beim Dotplot verschiedene Muster, die uns zeigen, welche Unterschiede und Ähnlichkeiten zwischen beiden Sequenzen existieren.

s.8.

Abbildung 3: Glättung von Matrix, wenn Sequenzen sehr lang werden.

- Glättung: Mittelwert aneinandergrenzender Zellen von **M**



Wenn man **sehr lange Sequenzen** hat, hilft es einem, wenn man den Dotplot glättet. Dadurch kann man die Muster dann besser erkennen. Wir haben in unserer Dotplot-Matrix ja nur 1 und 0 stehen. Wir glätten, indem wir die benachbarten Zahlen nehmen und einen Mittelwert bilden. Ich bilde Mittelwerte über die gesamte Matrix hinweg. So glättere ich meine Matrix und erkenne das Muster besser.

### s.9. Dotplot – Berechnungsalgorithmus.

In der Informatik – nicht nur Bioinformatik- verwendet man häufig **Pseudocode**. Pseudocode bedeutet, dass der Code nicht in einer bestimmten Programmiersprache geschrieben ist, sondern man hat einen Code, der sehr allgemein gehalten ist und man muss den Code dann an die

gewünschte Programmiersprache anpassen. Grundsätzlich sind bestimmte Elemente zwischen den Programmiersprachen sehr ähnlich. Das bedeutet, dass man nach dem Erlernen einer Sprache die anderen Sprachen auch relativ einfach erlernen kann, da viele Elemente gleich sind. Ich habe hier mal für den Dotplot den Pseudocode aufgeschrieben (siehe Abbildung 4). Hier werden zwei so genannte Vorschleifen entwickelt. Ich habe schon mal erwähnt, dass ein Computer sehr einfach denkt. Deshalb braucht der Computer auch sehr einfache Vorgaben und eine Möglichkeit für leichte Vorgaben besteht darin, dass ich dem Computer sage, er solle eine bestimmte Aktion mehrere Male wiederholen. Wir wollen ja beim Dotplot die Tabelle/Matrix füllen und das bedeutet für den Computer, dass er alle Zeilen und Spalten durchlaufen muss. Dieses Durchlaufen aller Zeilen und Spalten macht man über die Vorschleife. Ich gebe den Computer den Befehl "for i=1, ...,n {" , damit er alle Zeilen durchläuft. Mit "for j=1, ..., m {" durchläuft der PC alle Spalten. Mit "if (A(i)==A(j))M(i,j)=1" überprüfe ich, ob alle Buchstaben gleich sind und wenn das der Fall ist, kriegt es den Wert 1. Wenn das nicht der Fall ist und man hat unterschiedliche Buchstaben, dann setzt "else M(i,j)=0" eine 0 an die Stelle in der Matrix. Das ist eine Möglichkeit die Tabelle auszufüllen. Es ist aber sehr aufwendig, da ich genauso viele Rechenoperationen brauche, wie ich Zeilen und Spalten habe. Wenn man in der Informatik wissen will, wie schnell ein Algorithmus ist, dann nimmt man sich nicht eine Stoppuhr und schaut an, wie viele Sekunden er braucht, weil es ja auch von der Leistung des Computers abhängt, wie schnell es abläuft. Ein leistungsstarker PC wird schneller sein als ein leistungsarmer PC. Stattdessen gibt man die **Anzahl der Rechenoperationen** für die Geschwindigkeit des Algorithmus an. Hier wäre die Geschwindigkeit **n\*m**: Also Anzahl Zeilen \* Anzahl Spalten, die der Computer in meiner Matrix ausfüllen muss mit einer 1 oder einer 0. Man spricht dann oft von der **Ordnung** des Algorithmus. Das Zeichen für Ordnung ist  $O(nm)$  und damit gibt man die Laufzeit von einem Algorithmus an. Ziel der Informatik und damit auch der Bioinformatik ist es, Algorithmen zu finden, die eine kürzere Laufzeit haben. Gerade bei der Sequenzanalyse, wo wir Datenbanken durchsuchen müssen, brauchen wir Algorithmen, die sehr schnell sind. Deshalb versucht ein Bioinformatiker Algorithmen zu finden, die besonders schnell sind.

Abbildung 4: Pseudocode für Dotplot.

## Dotplot – Berechnungsalgorithmus

- Pseudocode:
 

```

      for i=1,...,n {
        for j=1,...,m {
          if (A[i]==A[j]) M[i,j] = 1
          else M[i,j] = 0
        }
      }
      
```
- Laufzeit: Ordnung  $O(nm)$

### s.10. Dotplot für ganze Genome.

Abbildung 5: Dotplot für ganze Genome.

#### Dotplot für ganze Genome

- Definition:

- Sequenz von Genen A:  $a_1 \dots a_n$
- Sequenz von Genen B:  $b_1 \dots b_m$
- $(n \times m)$ -Matrix **M**
- $M[i,j] = 1$  falls  $a_i = b_j$        $i = 1, \dots, n$  und  $j = 1, \dots, m$
- $M[i,j] = 0$  falls  $a_i \neq b_j$        $i = 1, \dots, n$  und  $j = 1, \dots, m$

Ich habe jetzt Beispiele gezeigt, wo man DNA-Sequenzen oder Protein-Sequenzen miteinander vergleicht. Wenn ich jetzt **große Genome mit dem Dotplot vergleichen** würde, dann wäre das Ganze nicht ganz adäquat, weil wir dann Millionen bis Milliarden Zeilen und Spalten in unserer Matrix hätten. Man kann da aber etwas dagegen machen. Man kann überprüfen, ob ich innerhalb zweier Genome an einer bestimmten Position das gleiche Gen habe. Das heißt, der Dotplot schaut nicht nach, ob eine Übereinstimmung zwischen den Basen existiert, sondern es schaut nach, ob man das gleiche Gen hat. Wenn das Gen auf den zwei Sequenzen eine geringfügig andere Basenabfolge besitzt, aber für dasselbe Protein codiert, dann würde man in der Matrix eine "1" setzen. Wenn das nicht der Fall ist, dann setzt man eine "0" in die Matrix. Insofern kann man den Dotplot in reduzierter Fassung auch für große Sequenzen verwenden.

### s.11. Dotplot

Fassen wir es kurz zusammen. Dotplot ist ein bioinformatisches Verfahren zur visuellen Analyse, wenn man zwei Sequenzen miteinander vergleichen möchte. Wir haben gesehen (s.7 Abb. 2), dass es bestimmte Muster für Insertionen, Deletionen, Inversionen oder repetitive Elemente gibt. Man kann sowohl DNA- als auch Aminosäure-Sequenzen im Dotplot vergleichen. Über Gen-Vergleiche kann man dann ganze Genom-Sequenzen vergleichen. Natürlich hat so ein graphisches Verfahren mehrere Einschränkungen. Zum einem ist es ein subjektives Verfahren: jede Person, die ihren Dotplot anschaut, wird etwas anderes sehen. Der Dotplot liefert auch kein Maß. Es gibt also keine Maßzahl, die Ihnen sagt: "ihre Sequenzen stimmen mit so viel Prozent zueinander überein". Das heißt, die Ähnlichkeit oder Unähnlichkeit zweier Sequenzen wird nicht quantifiziert.

### s.12. Paarweises Sequenzalignment (Global).

Hierfür hat die Bioinformatik weitere Algorithmen entwickelt. Mit diesen Algorithmen kann man auch ein Maß angeben und kann damit herleiten, wie ähnlich sich zwei Sequenzen sind. Das sind die Verfahren des Sequenzalignments. Wir schauen uns zuerst das paarweise Sequenzalignment an. Wir vergleichen also immer noch zwei Sequenzen und später kommt das multiple Sequenzalignment, wenn man mehrere Sequenzen gleichzeitig vergleichen möchte. Die Anforderungen für so ein Sequenzalignment sind, dass Ähnlichkeit bzw. Unähnlichkeit quantifiziert werden müssen. Das Alignment muss Deletionen und Insertionen berücksichtigen können. Gaps ("Lücken") müssen in die Sequenz einbaubar sein. Wenn ich in einer Sequenz eine Insertion habe, dann müsste ich ja in der anderen Sequenz an der Stelle eine Lücke oder Gap einbauen und die Sequenzen so auf die gleiche




Länge bringen, um die Sequenzen weiterhin vergleichen zu können. Man möchte so ein Alignment für einen präzisen Vergleich zweier Sequenzen nutzen oder ich mache es weniger Präzise, wenn ich viele Sequenzen absuchen muss, was bei einem Durschscreening von Datenbanken der Fall ist. Dann würde ich auch wenige Mismatches zwischen zwei Sequenzen erlauben.

### s.13-15. Paarweises Sequenzalignment (Global)

Schauen wir uns ein paar Maße an. Für die Übereinstimmung von Sequenzen gibt es mehrere Maßzahlen, um die Distanz zwischen zwei Sequenzen zu messen. Wir schauen uns dafür die **Hamming-Distanz** und **Levenshtein-Distanz** an. Beide sind Distanzmaßen für zwei Sequenzen. Bei der Hamming-Distanz schaue ich mir die Anzahl übereinstimmender Positionen an. Die Sequenzen "AAGTCTA" und "AAGTCCA" sind zum großen Teil identisch. Bei Hamming misst man jetzt also wie viele Positionen übereinstimmen. Bei der Levenshtein-Distanz ist es jetzt so, dass man aus zählt, wie viele Änderungen man vornehmen muss an einer Sequenz, bis aus ihr die andere Sequenz wird. Man misst also die Anzahl an notwendigen Editierungsoperationen (Löschen, Ergänzungen und Austausch), die notwendig sind, bis beide Sequenzen identisch sind. Wie viel muss ich in einer Sequenz ergänzen, löschen oder austauschen, um die andere Sequenz zu erhalten. Bei der Hamming-Distanz muss man je nach Software unterscheiden, ob man die Gleichheit oder die Ungleichheit zwischen zwei Sequenzen abbilden möchte. Man zählt also die Anzahl nicht übereinstimmender Positionen zwischen den Sequenzen "AAGTCTA" und "AAGTCCA" und man kommt auf eine Distanz =1 (der vorletzte Buchstabe in den Sequenzen stimmt nicht überein). Man hat also eine Position, die nicht übereinstimmt und deshalb ist die Hamming-Distanz =1. Das kann man aber auch normieren durch die Längen der beiden Sequenzen. Bei Levenshtein wird ein Score entwickelt (siehe Abbildung 6). Wenn beide Sequenzen übereinstimmen, dann gibt es keinen Punktabzug. Wenn ich einen Mismatch habe, dann gibt es 1 Punktabzug und bei einem Gap 2 Punkte Abzug. Bei den Sequenzen "AAGTCTAGTA" und "AAGTCCA" müsste ich bei "AAGTCTAGTA" gleich an der ersten Position einen Gap einbauen, damit die ähnlichen Basen übereinander liegen. Dann hätte ich aber immer noch ein Mismatch zwischen einem "T" und einem "C" und müsste 3 Basen deletieren. Ein Mismatch und eine Lücke ergeben insgesamt einen Score von 3. Eigentlich müsste ich für das Löschen der 3 Basen noch Gaps einsetzen und dann wäre die Distanz noch größer. Die Take-Home-Botschaft ist aber, dass bei Levenshtein die notwendigen Editierungsoperationen gewichtet werden und je nach Software wird das Distanzmaß unterschiedlich definiert. Es gibt auch gewichtete Levenshtein Distanzen.

Abbildung 6: Beispiel für Levenshtein

- **Beispiel Levenshtein-Distanz**
- 
- Match=0, Mismatch=1, und Lücke=2
- Distanz D=3

Nochmal die Bitte. Achten Sie auf die Software. Berechnet die Software die Ähnlichkeit oder die Unähnlichkeit? Ähnlichkeitsmaßen arbeiten dann mit einem Score, während Distanzmaßen Belohnungen und Bestrafungen nutzen.

### S16-23. Paarweises Sequenzalignment (Global)

Schauen wir uns an, wie man dem Computer so ein Alignment beibringen kann. Dazu habe ich zwei Beispielsequenzen genommen: "TTIHO" und "TIHMO". Die Frage ist jetzt, wie können wir diese zwei Sequenzen alignen? Wir brauchen ja für den Computer wieder eine möglichst einfache Vorschrift. Ähnlich wie beim Dotplot geht man so vor, dass man die eine Sequenz in die Zeile schreibt und die andere Sequenz in die Spalte einträgt (Abbildung 7). Dann muss man nach einer bestimmten Vorgabe die Matrix ausfüllen. Dieses Mal nutzt man aber nicht "0" und "1", sondern man nutzt natürliche Zahlen. Man baut noch zusätzlich noch eine Zusatzspalte und Zusatzzeile ein, die man mit Nullen füllt. Dann hat man nach dem so genannten Needleman-Wunsch Algorithmus eine Zuordnungsvorschrift.

Abbildung 7: Matrix und Needleman-Wunsch Algorithmus.

#### Paarweises Sequenzalignment (Global)

		j=1	j=2	j=3	j=4	j=5
		T	I	H	M	O
i=1	T	0				
i=2	T	0				
i=3	I	0				
i=4	H	0				
i=5	O	0				

- Needleman-Wunsch Algorithmus
 
$$x_i = y_j: \quad F_{i,j} = \max \begin{cases} F_{i-1,j} \\ F_{i,j-1} \\ F_{i-1,j-1} + 1 \end{cases}$$

Man fängt dann an, von links oben nach rechts unten die Tabelle zu füllen. Man schaut auch hier, ob eine Übereinstimmung zwischen den Buchstaben der i-ten Zeile und j-ten Spalte vorliegt. Schauen wir uns dafür i=1 und j=1 an (Abbildung 7). Man schaut sich jetzt die Zeile darüber an, die Zelle links daneben und die Zelle diagonal links von der gewünschte Position. Wenn eine Gleichheit vorherrscht wie bei unserem Beispiel, wo wir in beiden Sequenzen an selber Stelle ein "T" (i=1 und j=1) haben, dann nehmen wir die größte Zahl und addieren eine 1 dazu. Sollte es nicht gleich sein, dann addieren wir nichts hinzu. So füllt man die Tabelle auf und die Zahlen werden nach unten rechts hin immer größer, weil wir bei Gleichheit immer eine 1 dazu addieren (siehe Abbildung 8).

Abbildung 8: Ausgefüllte Matrix nach Needleman-Wunsch (globales paarweises Alignment).

### Paarweises Sequenzalignment (Global)

		j=1	j=2	j=3	j=4	j=5
		T	I	H	M	O
i=1	T	0	1	1	1	1
i=2	T	0	2	2	2	2
i=3	I	0	2	3	3	3
i=4	H	0	2	3	4	4
i=5	O	0	2	3	4	5

Das Ausfüllen des Graphen kann man dem PC beibringen. Jetzt haben wir beide Sequenzen aber noch nicht aneinander aligniert. Alignieren heißt, dass wir prinzipiell in die eine oder die andere Sequenz Lücken einbauen. Beide Sequenzen sind nämlich typischerweise unterschiedlich lang. Deshalb wollen wir Sie durch das Einfügen durch Lücken strecken oder stauchen, damit Sie gut übereinander passen. Dazu fängt man in der Matrix rechts unten an und arbeitet sich zurück nach links oben hin. Man fängt also bei der letzten Zeile an (hier "5") und schaut sich die drei Nachbarzeilen an. Man geht dann zurück zur Nächstkleinere Zahl. In dem Fall (Abbildung 8) haben wir neben 5 drei Mal die Zahl "4". Das sind gleichgroße Zahlen. In solchen Fällen geht man zur Diagonalliegenden Zeile. Bei der nächsten Zahl (Abbildung 9) hat man dann die Zahl "4" und sie ist umgeben von einer "4" zu ihrer Linken, einer "3" über Ihrer und einer "3" diagonal links. Die Nächstgrößere Zahl wäre jetzt die 4 auf der linken Seite und deshalb würde man links gehen. Die Idee ist, dass man einen Pfad durch die Tabelle findet, indem man sich die drei linken Nachbarn einer Zahl anschaut und zur Nächstgrößeren Zahl geht.

Abbildung 9: Das Alignen der Matrix

### Paarweises Sequenzalignment (Global)

		j=1	j=2	j=3	j=4	j=5
		T	I	H	M	O
i=1	T	0	1	1	1	1
i=2	T	0	2	2	2	2
i=3	I	0	2	3	3	3
i=4	H	0	2	3	4	4
i=5	O	0	2	3	4	5

T	T	I	H	-	O
T	-	I	H	M	O



Es entsteht so ein Pfad, der horizontale, diagonale und vertikale Pfeile aufweist. Bei einem horizontalen Pfeil hat die Sequenz in der Zeile einen Gap (siehe Abb. 9). Bei einem vertikalen Pfeil hat die Sequenz in der Spalte an der Stelle einen Gap. Man sieht es auch in Abbildung 9, wo ich für das doppelte T in "TTIHO" eine Lücke einbauen muss und bei der anderen Sequenz muss ich für das zusätzliche "M" eine Lücke einbauen. Diese Lücken entstehen durch den vertikalen und horizontalen Pfeil. Man füllt also die Tabelle von links oben nach rechts unten auf. Dann legt man den Pfad zurück von rechts unten nach links oben. Dieser Pfad sagt mir dann, wo ich Lücken einbauen sollte. Es gibt verschiedene Varianten dieser Verfahren, je nachdem welche Software man nutzt und ich habe Ihnen hier die leichteste Variante gezeigt (nicht wundern, wenn man bei Dr. Reinard in Bioinformatik diese Verfahren anders kennengelernt hat).

#### s.24 -27. Paarweises Sequenzalignment (Lokal)

Das, was ich Ihnen bis jetzt gezeigt habe, würde man globales Sequenzalignment nennen. Das Problem beim globalem Sequenzalignment ist Folgendes: Das erste Problem tritt auf, wenn beide Sequenzen unterschiedlich groß sind. Sagen wir mal, ich habe eine kleine Sequenz A, die ich an eine viel größere Sequenz B alignen möchte. Dann ist es ja sehr wahrscheinlich, dass in Sequenz B Bestandteile von A mehrfach auftreten, schon allein, weil Sequenz B so groß sein kann, dass es statistisch sehr wahrscheinlich sein kann. Der Needleman-Wunsch Algorithmus würde versuchen das Problem zu umgehen, indem er sehr große Lücken in Sequenz A einbaut. Das hätte aber zur Folge, dass ich Sequenz A damit quer über Sequenz B verteile (siehe Abbildung 10).

Abbildung 10: Probleme des globalen Sequenzalignments

- Problemfall 1: eine kleine und eine viel größere Sequenz, A und B
- Globales Alignment: Symbole von A werden sehr breit über B verteilt



Sagen wir mal, Sequenz A ist ein Gen, welches in Sequenz B auftaucht. In diesem Fall würde mir die Verteilung der Sequenz A über die ganze Sequenz B wie in Abbildung 10 nicht weiterhelfen. Bei Genen würde ich wollen, dass das Gen auf Sequenz A nur lokal auf Sequenz B auftaucht wie in Abbildung 11.

Abbildung 11: Sequenz A lokal aligns auf Sequenz B



In Abbildung 11 wird Sequenz A auf ein kleines Gebiet in Sequenz B fokussiert und man aligns damit lokal zwischen Sequenz A und B.

Ein zweites Problem tritt auf, wenn ich zwei ähnlich große Fragmente habe, die aber nur an ihren Enden eine sehr hohe Übereinstimmung haben. Das zweite Problem tritt also auf, wenn ich nur lokal an einer oder wenigen Stellen eine hohe Übereinstimmung zwischen zwei Sequenzen habe. Wenn ich da den Needleman-Wunsch Algorithmus anwende, kann es auch sein, dass beide Sequenzen mit

sehr vielen Gaps versehen werden. Das entspricht aber nicht der biologischen Realität. Das findet man häufig, wenn Reads aus der Hochdurchsatz-Sequenzierung aneinander alignen möchte.

### s.28. Paarweises Sequenzalignment (Lokal)

Um diese beiden Probleme des globalen Alignments zu lösen, hat man das lokale Sequenzalignment entwickelt. Hier wird überwiegend der so genannte **Smith-Waterman-Algorithmus** verwendet. Auch hier gibt es aber eine ganze Menge Varianten. Das Prinzip ist hier ganz ähnlich. Auch hier wird wieder über Tabellen gearbeitet. Hier hat man allerdings einen flexiblen Start- und Endpunkt, um diesen Pfad in der Tabelle zu bilden. Dadurch, dass ich einen flexiblen Startpunkt zulasse, kann ich an verschiedenen Stellen überprüfen, wo Sequenz A gut an Sequenz B passt. In der Regel ist es dann auch so, dass mehrere Pfade durch die Tabelle möglich sind. Hier kommen dann die Bestrafungs- und Belohnungssysteme ins Spiel. Wenn man jetzt beim Smith-Waterman mehrere Pfade hat, dann haben diese Pfade auch unterschiedene Distanzen. Das Alignment mit der geringsten Distanz zwischen zwei Sequenzen wird dann verwendet.

### s.29. Multiple Sequenzalignment (MSA).

Die beiden besprochenen Algorithmen (Needleman-Wunsch und Smith-Waterman) sollten Sie vom Namen her kennen. Sie werden wahrscheinlich nie in die Situation kommen, dass Sie beide per Hand ausführen müssen. Glücklicherweise gibt es viele Softwares, die es für Sie übernehmen werden. Es wäre aber gut, wenn man das Prinzip im Kopf behält. Die beiden Sequenzen dienen nur dem Vergleich zweier Sequenzen.

Wenn wir jetzt eine phylogenetische Analyse machen wollen, müssen wir mehrere Sequenzen gleichzeitig vergleichen. Dafür brauchen wir ein multiples Sequenzalignment. Das ist natürlich deutlich Rechenaufwendiger und typischerweise gibt es dafür keine eindeutige Lösung. Es gibt für das mehrere Lösungsansätze: Man kann Lücken einbauen und zulassen, dass man an einigen Stellen einen Mismatch hat. Das hat aber Einfluss auf den finalen Score. Aus diesem Grund macht MSA nur bei wenigen Sequenzen wirklich sinn. 50 bis 100 Sequenzen gehen noch. Wenn man 1000e Sequenzen hat, dann macht MSA keinen Sinn mehr, da man dann zu viele Mismatches hat. Dann dauert es auch ewig, weil der Rechenaufwand enorm wird.

### s.30 Multiples Sequenzalignment (MSA) -iteratives Verfahren.

Es gibt auch hier verschiedene Herangehensweisen. Viele Verfahren folgen einer Iterativen Struktur. Sprich, man startet mit zwei Sequenzen – idealerweise, die Sequenzen, die das beste Paarweise Alignment haben- und man aligniert diese zwei Sequenzen zuerst aneinander. Um diese zwei anfänglichen Sequenzen zu finden, vergleicht man jede Sequenz zueinander und man würde eine Distanzmaß berechnen. Die zwei Sequenzen, die am besten zueinander passen, aligniert man zuerst. Ab da würde man dann die dritte Sequenz hinzufügen und diese auf die ersten beiden Sequenzen alignieren. Dann würde man die vierte Sequenz hinzufügen und auf die ersten 3 Sequenzen alignieren und die 5. Sequenz würde man dann auf die ersten 4 alignieren und man würde so dann Stück für Stück alle Sequenzen aneinander alignen. Zwischendrin werden immer Distanzmaße und Alignment-Scores berechnet. Ausgangspunkt eines Iterativen Systems ist, dass man am Anfang jede Sequenz miteinander vergleicht und eine Distanzmatrix aus den Paarweisen Alignments bildet. Dabei kann es aber auch vorkommen, dass die ersten Alignments, die man bildet, nicht stabil bleiben und man an diesen auch erstmal “rüttelt”. Das bedeutet, dass man bei Ihnen zusätzliche Gaps einbaut,

damit am Ende alle Sequenzen aufeinanderpassen. Man muss also möglicherweise bestehende Gaps nochmal verlängern.

### s.31. Multiples Sequenzalignment (MSA)- ClustalW.

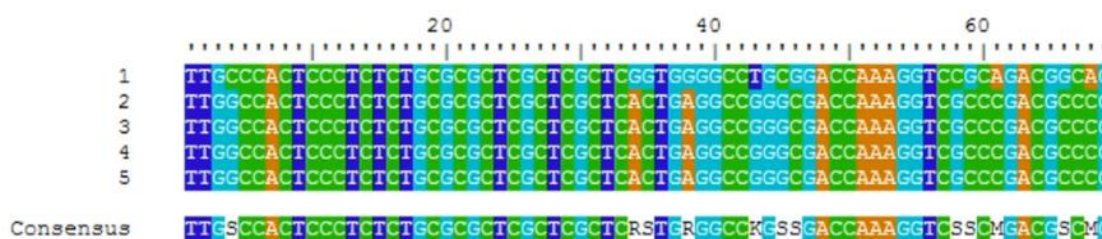
Hier möchte ich kurz den ClustalW-Algorithmus präsentieren. Das steht für “**Weighted Cluster Alignment**”. Das ist so ein Iteratives Verfahren. Auch hier startet man mit den zwei Sequenzen, die das beste Alignment haben. Die Reihenfolge der hinzufügenden Sequenzen ergibt sich nicht direkt aus der Distanz, weil man hier noch einen phylogenetischen Baum nutzt, um zu entscheiden, welche Sequenz man als Nächstes alignen sollte. Einen Nachteil hat dieser Algorithmus aber auch: Wenn die ersten Sequenzen bereits schlecht aufeinander aligniert sind, dann setze ich diesen Fehler beim weiteren Hinzufügen und Alignieren von Sequenzen fort. Ich habe also eine Fehlerfortpflanzung.

### s.32+33. Multiples Sequenzalignment (MSA)- Konsensussequenz

Es gibt kein eindeutiges multiples Sequenzalignment. Insofern hat jedes MSA Defizite. Was kann man jetzt aber mit einem Sequenzalignment machen? Man kann es als Grundlage für einen phylogenetischen Baum verwenden. Wir wollen aber auch mal schauen, was wir im Bereich der Infektionsbiologie alles machen kann. Ein Konsensusalignment kann auch Grundlage sein für eine Konsensussequenz. Viren haben ja kleine Genome und wenn ich jetzt eine Handvoll Viren aufeinander aligniere, die phylogenetisch sehr ähnlich sind, dann könnte ich für diese Viren eine gemeinsame Konsensussequenz ermitteln. Ich schaue also nach, an welcher Position diese Viren denselben Buchstaben haben. Eine Konsensussequenz folgt typischerweise dem so genannten IUPAC-Code. IUPAC steht für “International Union of Pure and Applied Chemistry”. Es kann ja sein, dass ich nach einem Alignment an bestimmten Stellen keine klaren Übereinstimmungen habe. Es gibt also Stellen, wo nicht eindeutig ein “A”, “G”, “C” oder “T” auftritt. Es kann sein, dass zur Hälfte A und zur Hälfte G auftaucht. Dann würde man den Buchstaben R nutzen. Bei der Dreier-Kombination C oder G oder T nutzt man den Buchstaben B und wenn alle 4 Basen vorkommen an einer bestimmten Position, dann nutzt man N. Es gibt also nach IUPAC noch viel mehr Buchstaben, die man verwenden kann, um in der Konsensussequenz darauf hinzuweisen, welche Basen man an einer Position vorfinden könnte. Lücken kann man durch einen Strich (-) oder durch einen Punkt (.) kennzeichnen. Wenn Sie in einer Software eine Konsensussequenz bilden, werden Sie typischerweise auch nach einem Schwellenwert gefragt. Der Schwellenwert gibt an, ab wann bei einer Nicht-Übereinstimmung einer der anderen Buchstaben an der Position verwendet werden soll. Wenn Sie 100 Sequenzen haben und da tritt 95-Mal ein “A” auf, dann würden Sie in die Konsensussequenz auch ein A reinschreiben. Diesen Threshold können Sie aber selbst festlegen und entscheiden, ab wie viel Prozent Ungleichheit ein anderer Buchstabe verwendet werden sollte.

In Abbildung 12 haben wir einen kleinen Ausschnitt von 5 Sequenzen, die über MSA aneinander aligniert wurden.

Abbildung 12. Konsensussequenz aus 5 Sequenzen



Unter den 5 Sequenzen sehen Sie die Konsensus-Sequenz.

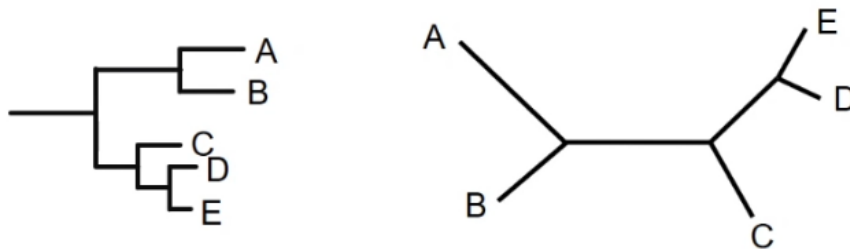
### s.34. Phylogenetische Bäume.

Dieses Alignment als Grundlage für phylogenetische Analysen schauen wir uns an. So ein phylogenetischer Baum ist hierarchisch aufgebaut. Zunächst ist es nur ein mathematischer Vergleich der Ähnlichkeiten. So basieren diese Vergleiche zum Beispiel auf Sequenzdaten. Ich kann eine Ähnlichkeitsmatrix aber auch auf morphologischen Ähnlichkeiten basieren lassen oder andere Eigenschaften nutzen. Es ist viel Vorsicht geboten bei der biologischen Interpretation dieser Bäume, weil die Ähnlichkeit unterschiedliche Gründe haben kann. Entweder es liegt tatsächlich ein gemeinsamer Vorfahre vor oder es hat horizontaler Gentransfer stattgefunden – also Genaustausch zwischen verschiedenen Arten.

### s.35. Phylogenetische Bäume.

Für diese phylogenetischen Bäume gibt es verschiedene Layouts. Es gibt Bäume mit Wurzeln, die man **Dendrogramm** nennt. Sie geben eine zeitliche Richtung an. Bäume ohne Wurzeln wäre zum Beispiel ein radialer Baum, wie man ihn rechts in Abbildung 13 sieht.

Abbildung 13. Phylogenetische Bäume



Beim Dendrogramm (links) ist eine Zeitachse noch möglich. Man kann also sagen, dass die Distanz der Äste eine zeitliche Entwicklung wiedergeben. Je länger die Äste desto länger ist auch die Zeitliche Entwicklung, bis der Unterschied und die Aufteilung in zwei neue Spezies aufgetreten ist.

Radiale Bäume nutzt man, wenn man viele Arten clustern will, da es sich räumlich einfach leichter aufteilen lässt auf einem DNA4-Blatt. Man kann hier aber nur schwer eine zeitliche Achse einzeichnen.

Dann unterscheidet man noch Bäume, wo alle Zweige gleich lang sind. Das wäre ein Cladogramm. Manchmal findet man auch Bäume mit unterschiedlichen Zweiglängen. Das wäre ein Phylogramm.

### s.36- 42. Phylogenetische Bäume - Neighbor-Joining-Algorithmus

Auch für das Zusammenführen von Spezies in einem phylogenetischen Baum gibt es sehr viele Algorithmen. Einer der bekanntesten ist der sogenannte Neighbor-Joining-Algorithmus. Den möchte ich hier nur kurz skizzieren. Dieser Algorithmus ist sehr rechenintensiv. Man lässt es auch vom Pc machen und macht es nicht selbst per Hand. Ausgangsdaten für so einen phylogenetischen Baum ist eine Distanzmatrix. Ich betone es noch einmal: **Je nach Software müssen Sie aufpassen, ob Sie vorher eine Distanz- oder eine Ähnlichkeitsmatrix berechnet haben!** Wenn Sie anstelle einer Distanz eine Ähnlichkeit ausdrücken wollen, müssen Sie das Ganze Kehrwert nehmen oder Minus 1 multiplizieren. Diese Distanzen können dann abgelesen werden aus einem Multiplen

Sequenzalignment, wie wir es kennengelernt haben. Typischerweise hat man dann eine quadratische Matrix. Wenn ich dann zum Beispiel 4 Sequenzen (S1-4) miteinander vergleichen möchte (Abbildung 14), dann würde ich für jede Sequenz eine Zeile und eine Spalte verwenden. Diese Matrix ist nicht nur quadratisch, sie ist auch symmetrisch. Quadratisch heißt ja: Gleichviele Spalten und gleichviele Zeilen. Symmetrisch heißt, dass im oberen Dreieck (Abgrenzung durch die Nullen) das gleiche steht, wie im unteren Dreieck, weil der Abstand zwischen Sequenz 1 und Sequenz 2 gleich so groß ist wie der Abstand zwischen Sequenz 2 zu 1. Deshalb hat man bei beiden Fällen eine "30" stehen (Siehe Abbildung 14). Wenn es wirklich eine Distanzmatrix ist, dann haben wir auf der Diagonale Nullen, weil jede Sequenz mit sich selbst natürlich keine Distanz aufweist, da sie identisch sind. Jetzt ist die Frage, welche zwei Sequenzen würde ich zuerst zusammenführen? Das ist auch ein Iteratives Verfahren. Ich muss also zuerst die beiden Sequenzen finden, die ich zuerst aneinander gruppiere in einem Ast, um dann weitere Äste hinzuzufügen.

Abbildung 14. Neighbor-joining-Algorithmus-Matrix

D =

	S1	S2	S3	S4
S1	0	30	10	20
S2	30	0	20	10
S3	10	20	0	5
S4	20	10	5	0

- Durchschnittliche Distanzen von jeder Sequenz zu allen anderen
- $r_1 = \frac{0+30+10+20}{4-2} = 30$ ,      entsprechend  $r_2=30$ ,  $r_3=17,5$ ,  $r_4=17,5$

Dazu verwendet man beim Neighbor-Joining-Algorithmus mehrere Schritte. Ausgehen von der Distanzmatrix berechnet man die Durchschnittliche Distanz von jeder Sequenz zu allen anderen Sequenzen. Schauen wir uns das mal für Sequenz 1 an (Abbildung 14). Es ist egal, ob wie für S1 die erste Spalte oder Zeile nehmen, da in beiden dasselbe drinnen steht. Normalerweise nimmt man nicht den genauen Mittelwert, sondern man gewichtet das Ganze. Hier haben wir deshalb im Nenner (Rechnung in Abbildung 14) die Minus 2 stehen. Ich errechne also jetzt die durchschnittlichen Distanzen für alle Sequenzen zu allen anderen Sequenzen. Wenn ich die Durchschnittlichen Distanzen berechnet habe, kann ich mir eine so genannte Zwischenmatrix bauen (siehe Abbildung 15). Diese wurde in Abbildung 15 mit D\* bezeichnet. Ich ziehe hier von den ursprünglichen Distanzen nochmal die Summe der durchschnittlichen Distanzen ab. Wenn die ursprüngliche Distanz (d<sub>ij</sub>) sehr klein ist – vielleicht sogar nah an 0-, aber die durchschnittliche Distanz von Sequenz i und j sehr groß ist, dann erhalten wir einen großen negativen Wert wie man in Abbildung 15 sehen kann. Wenn die ursprüngliche Distanz hingegen groß ist und die durchschnittlichen Distanzen der Sequenzen sehr klein sind, dann bleibt der Wert groß und im positiven Bereich. D\* bildet also erstmal ab, wie groß die Distanz zwischen zwei Sequenzen ist. Es wird aber noch berücksichtigt, wie groß die durchschnittliche Distanz beider zu allen anderen ist. Am Ende ist es so, dass sehr kleine Werte aussagen, dass zwei Sequenzen sich am Ähnlichsten sind. Wenn das d<sub>ij</sub> schon sehr klein ist und Sie zu den anderen Sequenzen eine sehr große Durchschnittliche Distanz haben, dann wird D\* sehr klein.

Die zwei Sequenzen mit den geringstem D\*-Wert (hier -37,5 in Abbildung 15) wären die Sequenzen, die ich am Anfang zu einem Ast zusammenführe, bevor ich die anderen anfangs hinzuzufügen.

Abbildung 15. Zwischenmatrix.

$D^* =$

	S1	S2	S3	S4
S1	0			
S2	-30	0		
S3	-37,5	-27,5	0	
S4	-27,5	-37,5	-30	0

- Matrix  $D^*$

- $d_{i,j}^* = d_{i,j} - (r_i + r_j)$

In unserem Beispiel würde ich jetzt S1 und S3 zusammenführen und ich hänge beide als eine neue Zeile bzw. Spalte in meine Distanzmatrix (Abbildung 16).

Abbildung 16: Distanzmatrix nach  $D^*$

- Sukzessives Zusammenführen der Äste

$D_{\text{neu}} =$

	<del>S1</del>	S2	<del>S3</del>	S4	S13
<del>S1</del>	0	30	10	20	
S2	30	0	20	10	20
<del>S3</del>	10	20	0	5	
S4	20	10	5	0	7,5
S13		20		7,5	0

- Matrix  $D_{\text{neu}}$

- $d_{m,k} = \frac{d_{i,k} + d_{j,k} - d_{i,j}}{2}$        $d_{5,4} = \frac{d_{1,4} + d_{3,4} - d_{1,3}}{2} = \frac{20 + 5 - 10}{2} = 7,5$

Nachdem ich sowohl S1 und S3 zusammengeführt habe, wird die Zeile und Spalte für S1 und S3 getrennt aus der Distanzmatrix gelöscht. Dann muss ich die Distanz zwischen S13 und S4 berechnen und kann eine reduzierte Matrix erstellen (Abbildung 17). In der reduzierten Matrix sind S2 und S4 noch enthalten, aber S1 und S3 wurden zu S13 zusammengeführt. So würde ich jetzt mit den anderen Sequenzen weitermachen und die Matrix so immer weiter reduzieren. Man sieht, dass es sehr aufwendig ist. Deshalb ist man froh, dass es der PC selber macht und er macht es auch sehr schnell, solange es wenige Sequenzen sind. So werden dann Stück für Stück Äste zusammengeführt.



Abbildung 17. Reduzierte Matrix mit S13.

- Sukzessives Zusammenführen der Äste

$$D_{\text{neu}} =$$

	S2	S4	S13
S2	0	10	20
S4	10	0	7,5
S13	20	7,5	0

- usw.

Wie ich bereits gesagt habe, ist Neighbor-Joining einer von vielen Algorithmen.

#### s.43. Phylogenetische Bäume - Bootstrap Verfahren.

Wenn ich weiß, welche Äste ich zusammenfügen muss und welche Äste einen Baum bilden, dann sollte mir bewusst sein, dass hinter jeder Datenanalyse eine hohe Unsicherheit steckt. Das liegt daran, dass ich oft mehrere Möglichkeiten habe, die Äste zusammenzufügen.

Daher möchte ich hier auf die sogenannten Bootstrap Verfahren zu sprechen kommen, die man bei vielen Fragen der Bioinformatik anwendet, inklusive Sequenzalignment.

Wenn ich eine Unsicherheit habe bei einer Datenanalyse, dann wiederhole ich diese Analyse mehrere Male – zum Beispiel 100-Mal- und bilde einen “Durchschnittsbaum” von einem phylogenetischen Baum. Die Unsicherheit entsteht ja zum großen Teil durch das multiple Sequenzalignment, die ja für den phylogenetischen Baum genutzt wurde. Was macht man jetzt aber? Mal angenommen, ich habe ein Sequenzalignment, wo ich durch das Setzen von Lücken alle Sequenzen auf eine Länge von 100 bp gesetzt habe. Dann würde ich zum Beispiel 90 % dieser Position heranziehen, um mir einen Baum daraus basteln zu lassen. Dann wiederhole ich es. Dieses Mal ziehe ich eine andere Stichprobe, die auch 90 % beträgt, und ich wiederhole es dann nochmal sehr oft. Jedes Mal wird der Baum also neu berechnet. Darüber kann ich dann schauen, wie stabil so ein Baum bleibt. An einen Baum sieht man dann an jeder Verzweigung Zahlen, die maximal 100 betragen können. Die Ergebnisse des Bootstrapping, wo ich den Baum mehrere Male mit verschiedenen Konfiguration bauen ließ, stehen an den Verzweigungen. Wenn da eine “100” steht, dann heißt es, diese Verzweigung ist in allen 100 Durchläufen aufgetreten. Ich habe dann so eine Gewissheit darüber, wie hoch die Unsicherheit ist für die Clusterung von bestimmten Sequenzen. Wenn da eine 73 steht, dann heißt es diese Verzweigung ist in 73 % aller Durchläufe aufgetreten. Man sollte diese Verzweigung also mit einer höheren Unsicherheit betrachten. Bootstrapping hilft also bei der Bewertung von einzelnen Verzweigungen. Es ist aber auch bei anderen Analysen der Bioinformatik der Fall, dass man eine Analyse sehr oft wiederholt und wenn etwas häufiger auftritt, dann hat es eine höhere Wissenschaftliche Sicherheit.

#### **s.44-49. Beispiele aus der Infektionsforschung.**

Dann schauen wir uns ein paar Beispiele aus der Infektionsforschung an. Im Moment treibt sich die Afrikanische-Schweine Pest herum. 1990 ist jetzt eine Studie erschienen, wo eine phylogenetische Analyse des Virus unternommen wurde, der für die Krankheit verantwortlich ist (ASFV = Afrikanischer Schweine-Fiber Virus). Der Virus befällt primär Haus- und Wildschweine. Zecken dienen als Vektoren. Dieses Virus selbst induziert dann eine bestimmte Thymidinkinase-Aktivität in den infizierten Zellen. Diese Kinase ist bekannt bei Viren und anderen Vertebraten wie zum Beispiel Mensch, Maus und Huhn. Ziel der phylogenetischen Analyse war es, herauszufinden, ob man den Virus von anderen Viren abgrenzen kann, die eine ähnliche Replikationsstrategie haben. Man wollte also sehen, ob diese Viren einen gemeinsamen evolutionären Ursprung haben. Man hat jetzt die Aminosäuresequenzen der Kinase verschiedener Spezies miteinander verglichen.

Dazu wurde ein MSA durchgeführt. Dann wurde ein phylogenetischer Baum gebildet.

Sie haben vermutet, dass die Kinase im ASFV unterschiedlich zu Pockenviren ist. Mensch, Maus und Huhn waren aufgrund ihrer Phylogenie nah aneinander geclustert. Weil die Pockenviren nah an Wirbeltieren geclustert waren, hat man den Schluss gemacht, dass die Pockenviren ihre Kinase aus ihrem Wirt erworben haben. ASFV hat Kinase-Sequenz möglicherweise früher erhalten als Pockenvirus.

Ein letztes Beispiel schauen wir uns noch an. Hier geht es um eine phylogenetische Analyse von Herpesviren in Säugetieren und Vögeln. Die Autoren mussten Teilbäume bilden, weil Sie nicht gesamte Genome zur Verfügung hatten, sondern nur max. 8 Gene. Dann haben Sie die Teilbäume zusammengeführt zu einem Baum.

Man sieht also, dass es zu Situationen kommen kann, wo man sich nicht ausschließlich auf Standard-Tools verlassen kann. Manchmal muss man sich überlegen, wie man aus vielen kleinen Bäumen einen großen Baum zusammenbastelt.