

# Bioinformatik

## Substitutionsmatrizen

Ulf Leser

Wissensmanagement in der  
Bioinformatik



# Alignment mit linearem Platzbedarf

---

- Bisherige Algorithmen haben  $O(m \cdot n)$  Zeit- und  $O(m \cdot n)$  Platzbedarf
- Platzbedarf ist ein Problem für Alignierung großer Sequenzen (Genom-Genom)
- Gesucht: **Speicherplatzeffizienterer Algorithmus**

# Strings und reverse Strings

---

- Definition

- Sei  $A^r$  das *Reverse des Strings A*
- Für Strings  $A, B$  sei  $v^r(i, j) = v(A^r[1..i], B^r[1..j])$ 
  - $A^r[1..i]$  bezeichnet (wie immer) die ersten  $i$  Zeichen von  $A^r$ , also die letzten  $i$  Zeichen von  $A$

- Bemerkung

- Es gilt :  $v^r(i, j) = v(A[n-i+1..n], B[m-j+1..m])$ 
  - Denn:  $v(A^r, B^r) = v(A, B)$
- **Berechnung von  $v^r$**  kann exakt wie die von  $v$  erfolgen

**A** ..... **ATGCGGT**  
**B** ..... **GGTCGTAG**  
  
**A<sup>r</sup>** **TGGCGTA** .....  
**B<sup>r</sup>** **GATGCTGG** .....

# Problemhalbierung

---

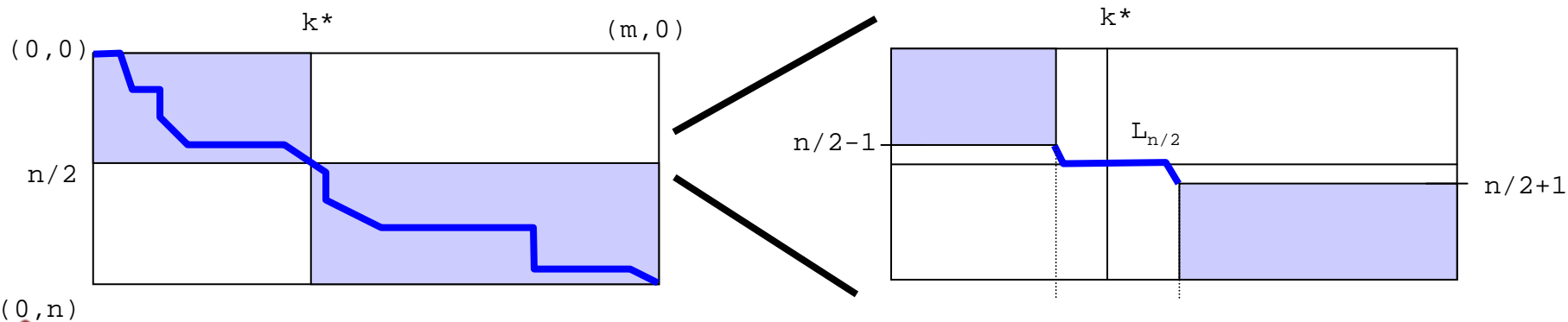
- Lemma  
*Gegeben zwei Strings A,B. Dann gilt*

$$v(n, m) = \max_{0 \leq k \leq m} \left( v(n/2, k) + v^r(n/2, m - k) \right)$$

- Beweisidee
  - Wir alignieren
    - A[1..n/2] mit B[1..k] **vorwärts**
    - A[n/2+1..n] mit B[k+1..m] **rückwärts**
  - Im optimalen Alignment muss irgendein Präfix von B mit A[1..n/2] alignieren
  - Der Rest muss dann mit A[n/2+1..n] alignieren
  - Durch das laufende k erwischen wir auf jeden Fall den richtigen Trennpunkt zwischen Präfix und Suffix
- Bemerkung
  - Das **Problem wird damit bzgl. |A| halbiert**

# Teilpfade

- Definition
  - Sei  $k^*$  das  $k$ , für das das Maximum  $v(n,m)$  erreicht wird
  - Sei  $L$  der Pfad von  $(0,0)$  bis  $(n,m)$
  - Sei  $L_{n/2}$  der Pfad zwischen dem letzten Knoten in der Zeile  $n/2-1$  und dem ersten Knoten in  $n/2+1$
- Lemma
  - $L$  und  $L_{n/2}$  müssen  $k^*$  enthalten
- Beweisidee:  $L$  muss irgendwo die Zeile  $n/2$  passieren
  - Die aufwendigen Fallunterscheidungen für gerade / ungerade Zahlen sparen wir uns



# Folgerungen

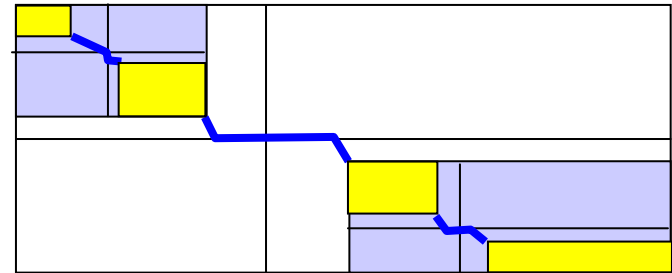
---

- Lemma
  - $k^*$  kann in *Zeit  $O(m \cdot n)$  und Platz  $O(m)$*  berechnet werden
  - $L_{n/2}$  kann ebenfalls in *Zeit  $O(m \cdot n)$  und Platz  $O(m)$*  berechnet werden
- Beweisidee
  - Berechne Matrix *zeilenweise vorwärts von 0 bis  $n/2$* ; speichere jeweils nur die letzte Zeile plus die Tracebackpfeile
    - $O(m \cdot n)$  Zeit,  $O(m)$  Platz
  - Berechne Matrix *zeilenweise rückwärts von  $n$  bis  $n/2$* ; speichere jeweils nur die letzte Zeile plus die Tracebackpfeile
    - $O(m \cdot n)$  Zeit,  $O(m)$  Platz
  - Mit den Werten  $v(n/2, 1..m)$  und  $v^r(n/2, 1..m)$  *kann man  $k^*$  finden* (Summe maximieren)
    - $O(m)$  Zeit, kein Platzverbrauch
  - Der Pfad  $L_{n/2}$  wird gefunden durch Traceback von Zelle  $(n/2, k^*)$  bis zu einer Zelle in Zeile  $n/2-1$  bzw.  $n/2+1$ 
    - $O(m)$  Zeit, kein Platzverbrauch

# Rekursion

---

- Wo sind wir?
  - Wir halten die gewünschten Komplexitätsschranken
  - Wir haben das Mittelstück von L berechnet
- Damit können wir **rekursiv absteigen**
  - Löse rekursiv die Probleme
    - für  $A[1..n/2] / B[1..k^*]$  und
    - für  $A[n/2+1..n] / B[k^*+1..m]$
- Platzkomplexität steigt nicht
- **Zeitkomplexität** steigt auch nicht
  - Zeitbedarf steigt um Faktor 2



# K-Band Algorithmus

---

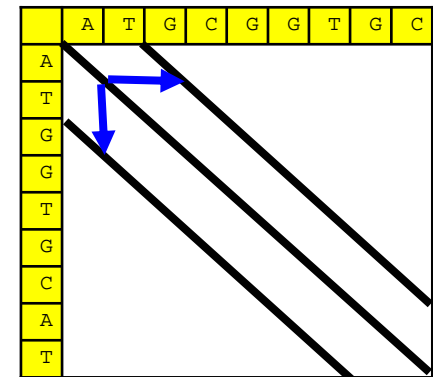
- Im Folgenden
  - Wir maximieren globale Ähnlichkeit
  - Sei  $s$  ist der Score eines Matches,
  - Sei  $b$  der negative Score einer Insertion oder Deletion
  - Kosten für einen Mismatch seien kleiner als  $b$
  - $|A|=|B|=n$
- Folgerung: Für das Alignment von  $A$  mit  $B$  gilt
  - Der bestmögliche Score ist  $n*s$ ,
  - Der schlechtestmögliche Score ist  $2*b*n$

# K-Band Algorithmus

- Ansatz: Wir erlauben nur **Abweichungen um maximal  $+k/-k$  Schritte**
- Algorithmus
  - Berechnet das beste globale Alignment innerhalb des Bandes der Breite  $2*k$

```
for i= 1 to n do
  for j= i-k to i+k do
    if (j<1) or (j>n) continue;
    M[i,j]= M[i-1,j-1] + t(A[i],B[j]);
    if inband(i-1,j) then
      M[i,j]= max( M[i,j], M[i-1,j]+b);
    if inband(i,j-1) then
      M[i,j]= max( M[i,j], M[i,j-1]+b);
  end for;
end for;
return M[n,n]
```

Beispiel:  $k=2$



# Eigenschaften

---

- Ein Gap im  $k$ -Band kann höchstens  $2k$  Leerzeichen lang sein
  - Wenn wir also optimale Alignments mit höchstens  $z$  langen Gaps suchen, finden wir die mit  $k=z/2$
- Komplexität des  $K$ -Band Algorithmus?
  - Für jede Zelle sind maximal 3 Zugriffe, 1 Addition und ein paar Vergleiche notwendig
  - Wir berechnen  $O(2k \cdot n)$  Zellen
  - Also:  $O(k \cdot n)$
- Können wir vielleicht noch wissen, wie weit weg wir vom Optimum gelandet sind?

# Optimalität

---

- Theorem

*Gegeben Strings  $A, B$  mit  $n=|A|=|B|$ . Sei  $v_K(A,B)$  der optimale  $K$ -Band Score für  $A$  und  $B$ . Wenn gilt*

$$v_K(A,B) \geq s^*(n-k-1) + 2b^*(k+1),$$

*dann ist  $v_K(A,B) = v(A,B)$ .*

- Beweis

- Wenn das optimale Alignment im  $k$ -Band verläuft, gilt auf alle Fälle  $v_K = v(A,B)$
- Wenn nicht, dann muss es irgendwo mindestens einmal aus dem  $K$ -Band laufen.
- Im besten solchen Fall haben wir also
  - $n-k-1$  Matches,
  - $k+1$  Insertions (Verlassen des Bandes)
  - $k+1$  Deletions (um am Ende noch  $(n,n)$  zu erreichen)

# Iteratives K-Band

---

- Das können wir ausnutzen, um das optimale Alignment **iterativ** zu finden
- Wir verdoppeln  $k$  in jedem Schritt, solange, bis wir garantiert das optimale Alignment haben

```
k = 1;
while (true) do
    compute  $v_k$ ;           // Costs  $O(k*n)$ 
    if  $v_k \geq s(n-k-1)+2b(k+1)$  then
        return  $v_k$ ;
    else
        k = 2*k;
    end if;
end while;
```

# Zusammen

---

- K-Band Algorithmus hat Komplexität  $O(sn^2 - vn)$
- Setzen wir z.B.  $s=1$ 
  - Dann kann  $v$  maximal  $n$  sein
  - Sehr ähnliche Sequenzen erreichen Wert nahe bei  $v$
  - Dann läuft der Algorithmus auch sehr schnell
- K-Band also umso besser, je **ähnlicher die Sequenzen** sind
  - Gut, um das schnell festzustellen (Algorithmus mit kleinen  $k$  laufen lassen)
  - Schlecht, um „irgendwelche“ Alignments zu berechnen

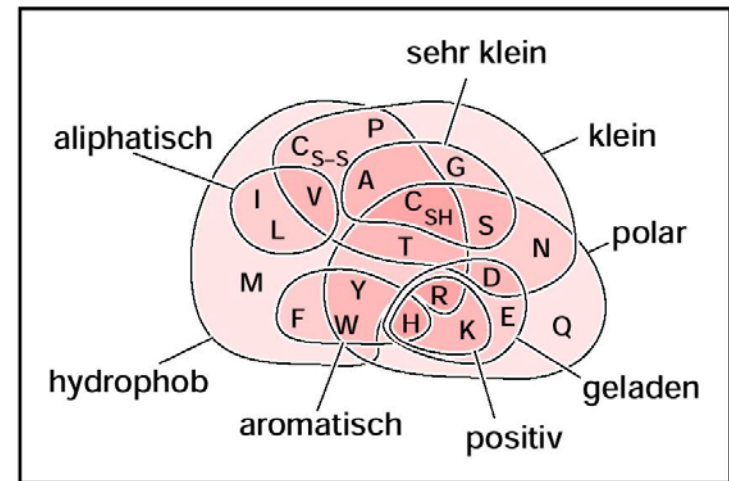
# Inhalt dieser Vorlesung

---

- Substitutionsmatrizen
  - PAM: Point-Accepted Mutations
  - BLOSUM: Blocks Substitution Matrices

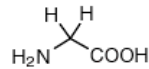
# Hintergrund

- Schon öfters angesprochen ...
  - Ähnlichkeitsmatrizen, Substitutionsmatrizen, Scorefunktionen, ...
- Ersetzung einer Base/Aminosäure durch eine andere hat **unterschiedliche Bedeutung**
  - Aminosäuren
    - Ersetzung mit „sehr ähnlichen“ Aminosäuren ändert Proteinstruktur in der Regel kaum
    - Ersetzung mit „wenig ähnlichen“ Aminosäuren kann Struktur vollkommen ändern

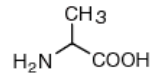


# Unterschiedliche Aminosäuren

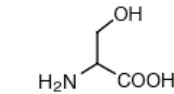
## Small



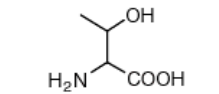
Glycine (Gly, G)  
MW: 57.05



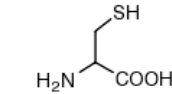
Alanine (Ala, A)  
MW: 71.09



Serine (Ser, S)  
MW: 87.08, pK<sub>a</sub> ~ 16

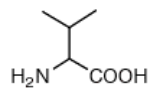


Threonine (Thr, T)  
MW: 101.11, pK<sub>a</sub> ~ 16

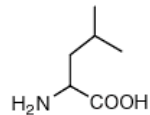


Cysteine (Cys, C)  
MW: 103.15, pK<sub>a</sub> = 8.35

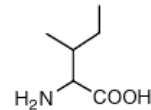
## Hydrophobic



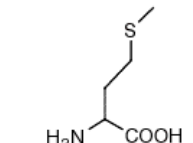
Valine (Val, V)  
MW: 99.14



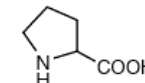
Leucine (Leu, L)  
MW: 113.16



Isoleucine (Ile, I)  
MW: 113.16

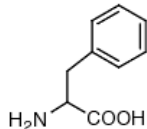


Methionine (Met, M)  
MW: 131.19

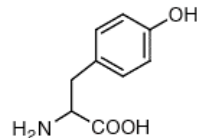


Proline (Pro, P)  
MW: 97.12

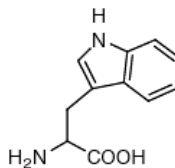
## Aromatic



Phenylalanine (Phe, F)  
MW: 147.18

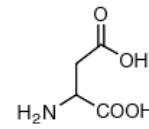


Tyrosine (Tyr, Y)  
MW: 163.18

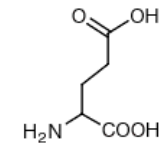


Tryptophan (Trp, W)  
MW: 186.21

## Acidic

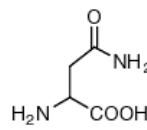


Aspartic Acid (Asp, D)  
MW: 115.09, pK<sub>a</sub> = 3.9

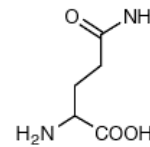


Glutamic Acid (Glu, E)  
MW: 129.12, pK<sub>a</sub> = 4.07

## Amide

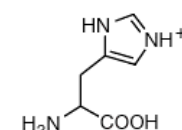


Asparagine (Asn, N)  
MW: 114.11

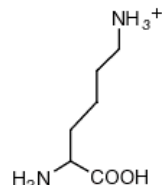


Glutamine (Gln, Q)  
MW: 128.14

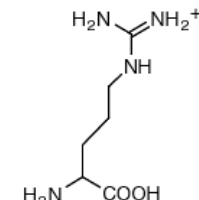
## Basic



Histidine (His, H)  
MW: 137.14, pK<sub>a</sub> = 6.04



Lysine (Lys, K)  
MW: 128.17, pK<sub>a</sub> = 10.79



Arginine (Arg, R)  
MW: 156.19, pK<sub>a</sub> = 12.48

# Substitutionsmatrizen

- Individuelle Bewertung von Replacements/Mismatches
- Alignmentalgorithmus bleibt unberührt
  - Das Ergebnis kann sich aber vollkommen ändern
- Beispiele: Blosom62, Identitätsmatrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	2	2	0	-3	-2	-1	-2	-1	1	-4	-3
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4

	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

# Ist das notwendig?

Code	Häufigkeit	Mutierbarkeit
L	0.091	54
A	0.077	100
G	0.074	50
S	0.069	117
V	0.066	98
E	0.062	77
K	0.059	72
T	0.059	107
I	0.053	103
D	0.052	86
P	0.051	58
R	0.051	83
N	0.043	104
Q	0.041	84
F	0.040	51
Y	0.032	50
M	0.024	93
H	0.023	91
C	0.020	44
W	0.014	25

- Häufigkeiten der Ersetzung einer Aminosäure durch irgendeine andere AA im Verhältnis zu allen Ersetzungen
- Alanin (A) willkürlich als 100% gesetzt
- **Keine Gleichverteilung**
- Mutationen sind mehr oder weniger erfolgreich, je nachdem, welche AA ersetzt wird
  - Besser: Mutationen werden durch Selektion mehr oder weniger geduldet
  - Tryptophan (W) sehr selten (25)
  - Serin (S) sehr häufig (117)

# Woher nehmen?

---

- Wie kann man sinnvolle Werte für die Matrix bestimmen?
  - Wir wollen **Ähnlichkeit der biologischen Bedeutung** messen
  - Für Aminosäuren benötigt man ~200 (verschiedene) Werte
- Möglichkeit 1: Chemische Eigenschaften
  - Ladung, Größe, Polarität, ...
  - Viele Faktoren mit unklaren Gewichten
  - Wie soll man das durch **ein Bewertungsschema** ausdrücken?
  - Keine Verwendung in der Praxis
- Möglichkeit 2: Beobachtung
  - **Beobachtung der Evolution** statt analytischer Vorhersage
  - Lernen aus Beispielen, also „tatsächlich“ vorgekommener Mutationen
  - Benötigt Beispieldaten: Paare von homologen Sequenzen

# PAM: Point-Accepted Mutations

---

- Dayhoff, M. O., R. V. Eck, C. M. Park. 1972. A model of evolutionary change in proteins. Pp. 89–99. *in* M. O. Dayhoff, ed., Atlas of Protein Sequence and Structure Vol. 5.
- PAM: Zwei Bedeutungen
  - 1 PAM – **Einheit** für den Abstand von Proteinsequenzen
  - PAM-X Matrix – Berechnete **Substitutionsmatrix** für zwei Sequenzen, die X PAM entfernt sind

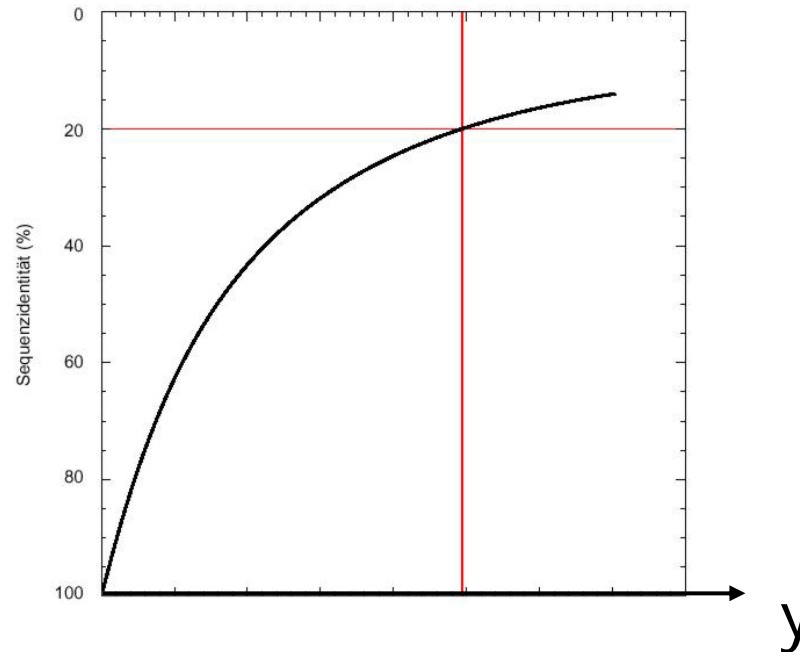
# Vorbetrachtung

---

- Annahme
  - Sequenz von 100 Aminosäuren
  - Mit jeder Generation ändert sich eine zufällig ausgewählte Position
    - Achtung: Das ist eine starke Vereinfachung
    - Molecular Clock theory
- Betrachten wir  $y$  Generationen
  - Menge der stattgefundenen Änderungen =  $y$
  - Die Menge der A-Posteriori beobachtbaren Änderungen korreliert mit  $y$
  - Wenn man Ausgangs- und Ergebnissequenz kennt, kann man  $y$  abschätzen
    - Mit welcher Wahrscheinlichkeit hat sich eine gegebene Position verändert?
    - Wie viele Positionen haben sich wahrscheinlich verändert?

# Mutationshäufigkeit und Sequenzidentität

---



- Kein linearer Zhg: Rückmutationen, Doppelmutationen, etc.
- Mit steigendem  $y$  wächst der Abstand zur Ausgangssequenz ab einem bestimmten Punkt kaum noch
- Ab diesem Punkt kann man über **Homologie** kaum noch eine Aussage machen

# PAM als Sequenzabstand

---

- Definition

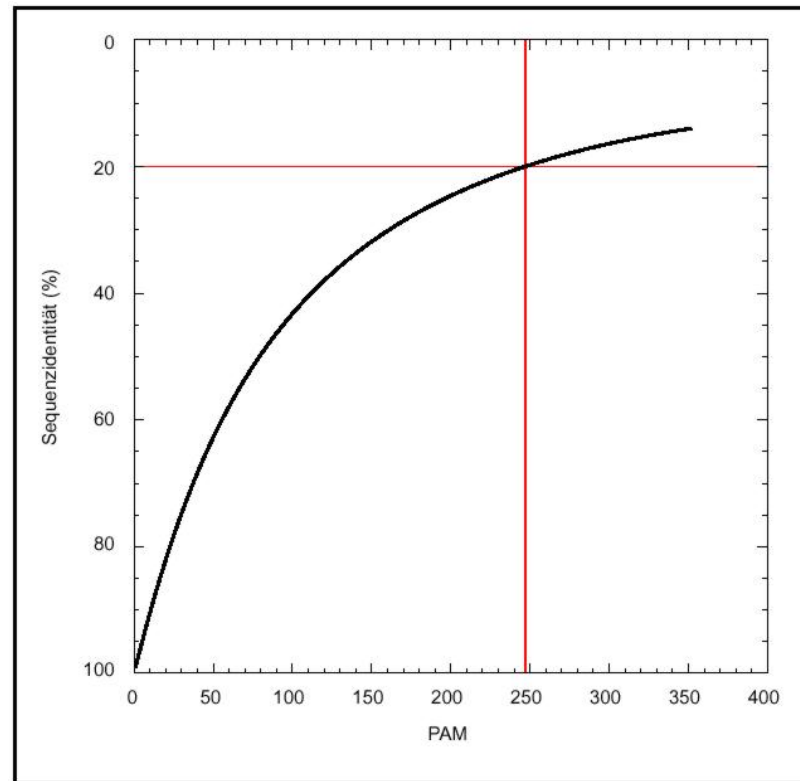
*Seien  $S_1$  und  $S_2$  zwei Proteinsequenzen mit  $|S_1|=|S_2|$ .  $S_1$  und  $S_2$  heißen  $x$  PAM entfernt, wenn  $S_1$  in  $S_2$  überführt wurde mit  $x$  Punktmutationen pro 100 Aminosäuren*

- Eigenschaften

- PAM beachtet keine Inserts und Deletions
- $x$  schätzt man aus den Unterschieden von  $S_1$  und  $S_2$ 
  - Kann man analytisch berechnen oder durch Simulation bestimmen
- 50 PAM Abstand heißt nicht 50 Veränderungen pro 100 Aminosäuren

# PAM Abstand und Sequenzidentität

---



- Jenseits von PAM 250: Rauschen

# PAM Matrizen

---

- Definition

Seien  $(S_{1,1}, S_{2,1}), \dots, (S_{1,n}, S_{2,n})$  Paare von Sequenzen, die jeweils  $x$  PAM entfernt sind. Dann ist die **PAM- $x$  Matrix  $M_x$**  wie folgt definiert:

- Sei  $f(A_i)$  die absolute Häufigkeit der Aminosäure  $A_i$
- Seien alle Paare optimal aligniert
  - Sei  $S_{k,i}$  das  $S_{k,l}$  mit den durch das Alignment eingefügten Leerzeichen
- Sei  $f(i,j)$  die **relative Übergangshäufigkeit** von  $A_i$  zu  $A_j$  in allen alignierten Paaren
  - Anzahl von Positionen  $k$  mit  $S_{1,z}[k]=A_i$  und  $S_{2,z}[k]=A_j$  oder umgekehrt
    - Übergang ist „richtungslos“:  $f(i,j) = f(j,i)$
    - Paare  $(A_x, -)$  werden ignoriert
  - $f(i,j)$  wird auf die Gesamtzahl aller Paare ohne INSDEL normiert
- Damit:

$$M_x(i, j) = \log \left( \frac{f(i, j)}{f(i) * f(j)} \right)$$

# Erläuterung

---

- **Log-Odds Ratio**
- Numerischer Trick: Logarithmus zur Ersetzung von Multiplikation mit Addition
- Bruch
  - Verhältnis der beobachteten Ersetzung zu den erwarteten Übersetzungen bei statistischer Unabhängigkeit

$$M_x(i, j) = \log\left(\frac{f(i, j)}{f(i) * f(j)}\right)$$

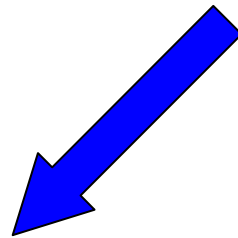
- $M(i, j) = 0$  (Bruch = 1)
  - **Keine Selektion** - Anzahl Übergänge entspricht statistischer Erwartung
- $M(i, j) < 0$  (Bruch < 1)
  - **Negative Selektion** – Übergang wird unterdrückt
- $M(i, j) > 0$  (Bruch > 1)
  - **Positive Selektion** – Übergang wird bevorzugt

# Beispiel

$S_{1,1}$ : ACGGTGAC  
 $S_{2,1}$ : AGG\_TGCC  
 $S_{1,3}$ : GTT\_AGCTA  
 $S_{2,4}$ : TTTCAG\_TA  
 $S_{1,2}$ : GGTCAA  
 $S_{2,2}$ : AGTC\_A

Absolute Häufigkeiten

A: 11/42	C: 8/42	G: 12/42	T: 11/42
----------	---------	----------	----------



Übergangshäufigkeiten



	A	C	G	T
A	4/19	1/19	1/19	0/19
C		2/19	1/19	0/19
G			4/19	1/19
T				5/19

Substitutionsmatrix

	A	C	G	T
A	0,48	0,02	-0,15	-
C		0,46	-0,01	-
G			0,41	-0,15
T				0,58

# Fehlerquellen

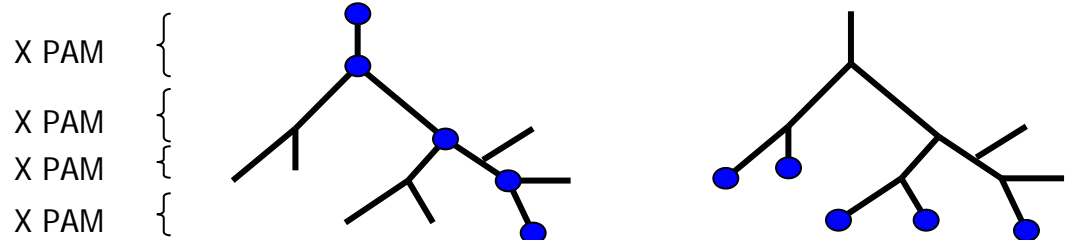
- Es kann **viele optimale Alignments** geben
  - Wahl eines Alignments besonders bei großem Abstand (also auch großen x) schwierig und subjektiv

Einfach: **FMM\_IYVVYL**  
**FMMUIYV\_YL**

Schwierig: **FMMF\_YV\_VYL**  
**\_\_UFPHVYLYL**

**\_FMMFYVVYL**  
**UFPHVYL\_YL**  
**FMMFYVVYL\_\_**  
**\_\_UFPHVYLYL**

- PAM Abstand kann man immer nur **schätzen**
  - Schätzung wieder fehleranfälliger, wenn x groß ist
  - Die beobachteten Sequenzen sind gar nicht direkt auseinander hervorgegangen



# Reale PAM Matrizen

---

- Annahme der „ **Molecular Clock Theory**“
  - Evolution verläuft gleichmäßig in der Zeit und in den Sequenzpositionen
  - Damit möglich: Hochrechnung der Matrixeinträge für große  $x$  aus der Matrix für kurze  $x$
- Vorgehen von Dayhoff et al.
  - Paare eng verwandter Sequenzen auswählen
    - >85% Identität, 34 Proteinfamilien
  - Manuell alignieren
  - PAM-1 Matrix  $M_1$  aus Häufigkeiten berechnen
  - PAM- $x$  Matrizen berechnen durch:  $M_x = (M_1)^x$



# Verwendung

---

- Welche PAM Matrix soll man zur Alignierung zweier Sequenzen verwenden?
  - Die, die dem PAM-Abstand der Sequenzen entspricht
  - Den kennt man aber nicht – schätzen
  - Schätzung benötigt Alignments
    - Zur Berechnung der Sequenzidentität
  - Alignments basieren auf Substitutionsmatrizen
  - **Henne – Ei Problem**
- Also
  - Iteratives Verfahren einsetzen und auf Konvergenz warten
  - Verschiedene Matrizen testen
  - Externes Wissen (Chemie, Domänen, etc.) hinzuziehen

# BLOSUM Matrizen

---

- Hauptkritikpunkte am PAM Ansatz
  - Nur Verwendung sehr ähnlicher Sequenzen
    - Realistische Zahlen für evolutionär weiter entfernte Sequenzen?
  - Einbeziehung kompletter Proteinsequenzen
    - Unterliegen alle Positionen der selben Mutationsrate?
  - PAM-x vervielfältigen Fehler in PAM-1
- Anderer (neuerer) Ansatz: BLOSUM
  - Henikoff, S. and Henikoff, J. G. (1993). "Performance evaluation of amino acid substitution matrices." *Proteins* **17**(1): 49-61.
  - **BLO**cks **SU**bstitution **M**atrix
  - **Multiplen Alignments evolutionär entfernt, aber homologer Proteinsequenzen**
  - Benutzung nur der konservierten Blöcke
  - Heute populärer als PAM Matrizen

# Vorarbeiten

---

- PROSITE

- Beschreibung funktionaler (=konservierter) Bereiche in [homologen Proteinsequenzen](#) durch reguläre Ausdrücke
- Expertenwissen - manuelle Pflege der Datenbank am EBI

- BLOCKS

- Alignierung von durch PROSITE Ausdrücke gematchten Sequenzen in [Multiple Alignments](#) (MSA)
  - Multiple Sequence Alignment (später mehr)
  - Heute: Verwendung weiterer Domänen aus PRINTS, PFAM, ...
- Ein [BLOCK](#) ist ein zusammenhängendes und hochgradig konserviertes Stück aus einem MSA

```
FMYMFYVPL PQ QVY
FYQDF VQLYP MFQV
FMY YUVQQP UMUQ
```

# BLOSUM Matrizen

---

- Berechnung der BLOSUM Matrizen verläuft identisch zur Berechnung der PAM-1 Matrix
  - Nur die BLOCKS werden betrachtet
  - Absolute Häufigkeiten aller Aminosäuren
  - Häufigkeiten aller Übergänge in allen Paaren
- BLOSUM–x Matrizen
  - Bias in BLOCKS durch viele sehr ähnliche Sequenzen
  - Zur Berechnung der BLOSUM-x Matrix werden in jedem Block alle Sequenzen mit >x% Identität zu einer Sequenz zusammengefasst
  - **Gänzlich andere Bedeutung** des „x“ als in PAM-x
  - Aber ähnliche Verwendung: x ~ evolutionärem Abstand

$$M_1(i, j) = \log\left(\frac{f(i, j)}{f(i) * f(j)}\right)$$

# Unterschiede PAM - BLOSUM

---

- BLOSUM verwendet nur hochkonservierte Bereiche, PAM komplette Alignments
- PAM rechnet große evolutionäre Abstände nur hoch, BLOSUM verwendet gezielt entfernte Sequenzen
- BLOSUM basiert auf deutlich mehr Sequenzen
- Heutige BLOSUM-Matrizen sind heuristisch verbessert
  - „Feedback-Schleife“: Mit initialer BLOSUM 62 Matrix erneute Alignierung
  - Bestimmung der BLOCKS verwendet BLOSUM Matrix
- BLOSUM-62 heute meist der Default in Alignmentprogrammen

# BLOSUM 45 Matrix

- Ausblendung Sequenzen >45% Sequenzidentität

Gly	7																			
Pro	-2	9																		
Asp	-1	-1	7																	
Glu	-2	0	2	6																
Asn	0	-2	2	0	6															
His	-2	-2	0	0	1	10														
Gln	-2	-1	0	2	0	1	6													
Lys	-2	-1	0	1	0	-1	1	5												
Arg	-2	-2	-1	0	0	0	1	3	7											
Ser	0	-1	0	0	1	-1	0	-1	-1	4										
Thr	-2	-1	-1	-1	0	-2	-1	-1	-1	2	5									
Ala	0	-1	-2	-1	-1	-2	-1	-1	-2	1	0	5								
Met	-2	-2	-3	-2	-2	0	0	-1	-1	-2	-1	-1	6							
Val	-3	-3	-3	-3	-3	-3	-3	-2	-2	-1	0	0	1	5						
Ile	-4	-2	-4	-3	-2	-3	-2	-3	-3	-2	-1	-1	2	3	5					
Leu	-3	-3	-3	-2	-3	-2	-2	-3	-2	-3	-1	-1	2	1	2	5				
Phe	-3	-3	-4	-3	-2	-2	-4	-3	-2	-2	-1	-2	0	0	0	1	8			
Tyr	-3	-3	-2	-2	-2	2	-1	-1	-1	-2	-1	-2	0	-1	0	0	3	8		
Trp	-2	-3	-4	-3	-4	-3	-2	-2	-2	-4	-3	-2	-2	-3	-2	-2	1	3	15	
Cys	-3	-4	-3	-3	-2	-3	-3	-3	-3	-1	-1	-1	-2	-1	-3	-2	-2	-3	-5	12
Gly	Pro	Asp	Glu	Asn	His	Gln	Lys	Arg	Ser	Thr	Ala	Met	Val	Ile	Leu	Phe	Tyr	Trp	Cys	

# Zusammenfassung

---

- Willkommen im Reich der Heuristiken
- Substitutionsmatrizen
  - Beobachtung tatsächlicher, evolutionär entstandener Unterschiede
  - Schätzung des evolutionären Abstands aus der Zahl der Unterschiede
  - Implizite Beachtung vielfältiger Faktoren
    - Ladungen, Nachbarschaft, Brücken über entfernte Aminosäuren durch 3D-Struktur, ...
  - Berechnung von Matrizen immer mit Hilfe „bekanntere“ echter Homologien – Henne-Ei Problem