Softwarewerkzeuge der Bioinformatik

Prof. Dr. Volkhard Helms
PD Dr. Michael Hutter, Markus Hollander,
Andreas Denger, Marie Detzler, Velik Velikov
Wintersemester 2021/2022

Universität des Saarlandes
Zentrum für Bioinformatik

Übungsblatt 3

Sequenzanalyse: Multiple Sequence Alignment (MSA) und Phylogenie

Lernziel: *In dieser Übung lernen Sie, wie man multiple Sequenzalignments erstellt, wie man diese beispielsweise bzgl. Konservierung interpretiert und wie man phylogenetische Bäume erstellt. Weiterhin wenden Sie den Sankoff Algorithmus an und lernen mehr über Python.*

Aufgabe 3.1: Homologe Sequenzen, gemeinsame Domänen und phylogenetischer Baum

a) Wenden Sie ProteinBLAST auf das Protein **Q38856** an. Denken Sie daran, die richtige Datenbank auszuwählen. Wählen Sie die ersten 10 Sequenzen aus und speichern sie im Multi–Fasta–Format, indem Sie auf "Download" → "FASTA (complete Sequence)" klicken. Speichern Sie analog die ersten 50 Sequenzen in einer anderen Datei.

b) Klicken Sie auf "Distance tree of results" und schauen Sie sich den phylogenetischen Baum der ersten 50 Sequenzen an. Wählen Sie "Taxonomic Name" unter "Sequence Label" aus, um die Spezies als Beschriftung zu erhalten. Finden Sie drei biologischen Gruppen (Pflanzen, Pilze, Tiere).

c) Auf http://www.ebi.ac.uk/Tools/msa werden verschiedene Tools für multiple Alignments angeboten. Wählen Sie ein Tool aus und wenden Sie es mit Standardeinstellungen auf die Datei mit den 10 Sequenzen an. Vollkommen konservierte Positionen sind mit "*" markiert, hohe Konservierung mit ähnlichen Eigenschaften mit ":", und weniger ähnliche Konservierung mit ".". Finden Sie zusammenhängende, **hoch** konservierte Bereiche in dem Alignment.

d) Angenommen Sie möchten das aktive Zentrum einen Proteins lokalisieren, haben aber nur die Proteinsequenz und keine Struktur. Wie kann ein multiples Sequenzalignment dabei helfen, dieses Problem zu lösen?

e) Erstellen Sie nun ein MSA von den 50 ähnlichsten Sequenzen mit demselben Tool. Welche Unterschiede stellen Sie zwischen den beiden Alignments fest?

Aufgabe 3.2: Outgroup

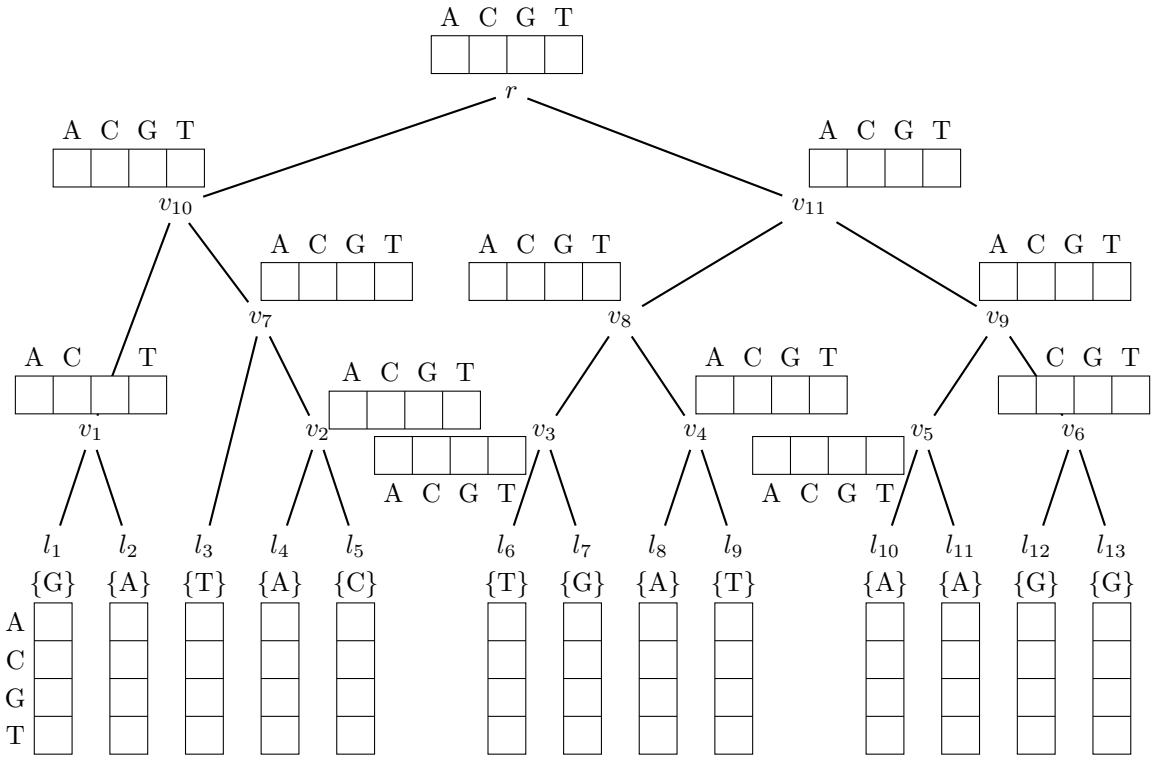
a) Erstellen Sie ein MSA aus den Sequenzen auf der Vorlesungsseite (sequences.fasta). Ist überall Konservierung zu erkennen?

b) Die Sequenz welcher Spezies unterscheidet sich am meisten von den anderen, wenn Sie sich nur das Alignment anschauen?

c) Benutzen Sie das Alignment Tool, um einen phylogenetischen Baum zu erstellen. Die Sequenz welcher Spezies unterschiedet sich hier am meisten von den anderen?

Aufgabe 3.3: Sankoff Algorithmus

Welche Base hatte die Urvorgängersequenz wahrscheinlich an der gegebenen Stelle eines Alignments? Verwenden Sie dazu den Sankoff Algorithmus und die folgende Kostenfunktion.



⟶ Base der Urvorgängersequenz:

Kostenfunktion:

	A	C	G	T
A	0	2	1	2
C	2	0	2	1
G	1	2	0	2
T	2	1	2	0

Aufgabe 3.4: Aminosäuresequenzen (Python)

a) Besuchen Sie kaggle.com und loggen Sie sich in ein. Erstellen Sie ein neues Notebook, benennen Sie es um, und löschen Sie die bereits existierende Zelle.

b) **Strings:** Strings sind Sequenzen von Buchstaben, Symbolen und Zahlen und repräsentieren somit Text. In der letzten Übung haben wir den String 'Hello World!' mit der *print*() Funktion angezeigt. Strings können Variablen zugewiesen und auch leicht mit "+" kombiniert werden:

```
part1 = 'MNPTLILAAFCLGIASATLTFDHSLEAQWTKWK'
part2 = 'AMHNRLYGMNEEGWRRAVWEKNMKMIELHNQEYREGK'
complete = part1 + part2
```

Kopieren Sie das Beispiel in eine leere Code Zelle. Fügen Sie eine Zeile hinzu, in der Sie mit der *print*() Funktion den Inhalt von *complete* anzeigen lassen.

c) Man kann auch einzelne Buchstaben eines Strings ansteuern, indem man dessen Position (*Index*) in eckigen Klammern angibt. Der Index ist dabei 0–basiert, d.h. der erste Buchstabe eines Strings hat Index 0 und nicht 1. Um den ersten Buchstaben der Sequenz aus b) zu bekommen, schreibt man also: $complete[0]$. Man kann auch die Position vom Ende aus angeben, z.B. um den letzten Buchstaben zu bekommen, schreibt man: $complete[-1]$.

Benutzen Sie die *print*() Funktion, um sich den ersten, zehnten, letzten und vorletzten Buchstaben von *complete* anzeigen zu lassen.

d) Man kann auch Abschnitte eines Strings ansteuern, indem man die Start- und Endindex des Abschnittes in eckigen Klammern angibt. Auch hier ist der Index 0–basiert. Dabei wird die angegebene Endindex nicht mehr inkludiert. Möchte man sich zum Beispiel Positionen 10 bis einschließlich 20 der Sequenz anzeigen lassen, schreibt man: $complete[9:20]$.

Benutzen Sie die *print*() Funktion, um sich die ersten 10 Buchstaben der Sequenz anzeigen zu lassen.

e) **Dictionaries:** Sogenannte "Wörterbücher" sind Datenstrukturen, die darauf ausgelegt sind, Werte (*values*) bestimmten Schlüsseln (*keys*) zuzuweisen. Mit den Schlüsseln können diese Werte dann wieder abgerufen werden. Die Werte können dabei einzelne Wörter, Texte oder Zahlen sein, aber auch komplexere Datentypen wie z.B. Listen.

Zum Beispiel, hier wird der Variable *symbols* ein (unvollständiges) Dictionary mit Aminosäure–symbolen zugewiesen. Die 1-Buchstabensymbole sind dabei die Schlüssel und die 3-Buchstaben–symbole die Werte:

```
symbols = {'A': 'Ala', 'R': 'Arg', 'N': 'Asn', 'D': 'Asp', 'C': 'Cys',
           'Q': 'Gln', 'E': 'Glu', 'G': 'Gly', 'H': 'His', 'I': 'Ile',
           'L': 'Leu', 'K': 'Lys', 'M': 'Met', 'F': 'Phe', 'P': 'Pro',
           'S': 'Ser', 'T': 'Thr', 'W': 'Trp'}
```

Kopieren Sie das Beispiel in eine leere Code Zelle und vervollständigen Sie es mit den fehlenden Symbolen von Tyrosin und Valin.

f) Um Werte aus einem Dictionary abzurufen, gibt man ähnlich wie bei Strings den Schlüssel in eckigen Klammern an. Zum Beispiel, $symbols['A']$ gibt den 3-Buchstabencode 'Ala' zurück.

Benutzen Sie die *print*() Funktion, um sich die 3-Buchstabencodes von Serin und Threonin anzeigen zu lassen.

g) **For–Loops:** Sogenannte For–Schleifen (*Loops*) sind nützlich, um eine Operation automatisch mehrfach ausführen zu lassen. In diesem Beispiel iterieren wir mit dem Schlüsselwort for über alle Aminosäuren (*aa*) in der Sequenz (*complete*) und benutzen unser Dictionary, um uns den 3-Buchstabencode anzeigen zu lassen.

```
for aa in complete:
    print(symbols[aa])
```

Dies ist auch das erste Beispiel, in dem wir sehen, dass Python mit Einrückung arbeitet, um Code-Abschnitte zu erkennen. Alles, was unter der For–Schleife nach rechts eingerückt ist, wird in jeder Iteration ausgeführt.

Kopieren Sie das Beispiel in eine leere Zelle und führen Sie sie aus.

h) **Funktionen:** Wir haben bereits die eingebaute *print*() Funktion kennen gelernt, die Dinge anzeigt. Funktionen sind sehr nützlich, wenn man etwas oft mit unterschiedlichen Eingaben ausführen möchte. In dem folgenden Beispiel erstellen wird mit dem Schlüsselwort *def* eine neue Funktion *transform*, die eine Sequenz *sequence* als Eingabe nimmt. Die Funktion iteriert dann mit einer For–Schleife über alle Aminosäuren in der Eingabesequenz und fügt den 3-Buchstabencode der Variable *new_sequence* hinzu. Am Ende gibt die Funktion diese neue Sequenz mit dem Schlüsselwort *return* zurück.

Alles, was nach der Funktion nicht mehr nach rechts eingerückt ist, gehört nicht mehr zur Funktion. Die Funktion kann dann genau wie *print*() aufgerufen und das Ergebnis auch Variablen wie z.B. *transformed_sequence* zugewiesen werden.

```
def transform(sequence):
    new_sequence = ''
    for aa in complete:
        new_sequence += symbols[aa]
    return new_sequence

transformed_sequence = transform(complete)
print(transformed_sequence)
```

Kopieren Sie das Beispiel in eine leere Zelle und führen Sie sie aus. Wenden Sie die Funktion auf eigene Aminosäuresequenzen an.

Viel Spaß!