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

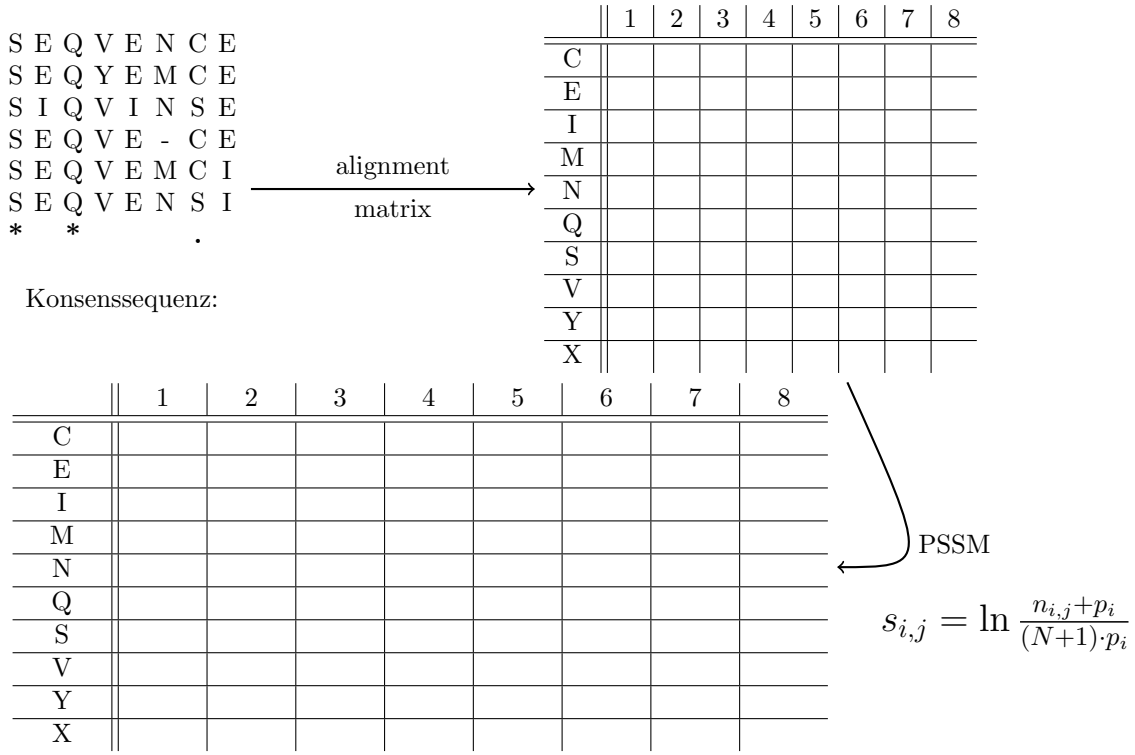
Übungsblatt 4

Sequenzanalyse: Motive und Transmembranhelices

Lernziel: *In dieser Übung lernen Sie zwei Datenbanken für Sequenzmotive kennen und ein Vorhersagetool für Transmembranhelices. Weiterhin berechnen Sie eine PSSM und die entsprechende Konsenssequenz per Hand und mit Python.*

Aufgabe 4.1: Positions–spezifische Gewichtsmatrix (PSSM)

Berechnen Sie aus dem folgenden Alignment die PSSM und geben Sie die Konsenssequenz an. (X steht für alle nicht gelisteten Aminosäuren.)



Parameter:

- $N = 6$: Sequenzanzahl
- $n_{i,j}$: Anzahl Buchstabe i an Position j
- $p_i = 0.05$: A–Priori–Wahrscheinlichkeit des Buchstabens i. Dies ist das einfachste Modell, da alle Buchstaben die gleiche Wahrscheinlichkeit haben.

Aufgabe 4.2: PRINTS

PRINTS (http://bioinf.man.ac.uk/dbbrowser/PRINTS/index.php) ist eine Datenbank für Protein–Fingerprints, also für Gruppen von kleineren, konservierten Motiven.

a) Suchen Sie das Mausprotein FosB (**P13346**) in PRINTS. Hierfür ist es meist einfacher, das Protein zuerst in UniProtKB zu suchen, dort auf *Family & Domains* und dann auf den PRINTS Link zu klicken.

b) Was ist die Accession des Fingerprints? Wie viele Motive beinhaltet es?

c) Aus wie vielen Proteinen wurden die Motive erstellt? Lesen Sie dafür entweder den Abschnitt *Documentation* oder zählen Sie für eines der Motive die Zeilen im Abschnitt *Initial Motifs*.

d) Enthält **P13346** alle Motive? Hierfür können Sie mit der alternativen UniProt Accession (**FOSB_MOUSE**) im Abschnitt *Final Motifs* suchen.

e) Die Startposition eines Motivs wird auf der PRINTS Seite für jedes Protein in der Spalte *st* angegeben. Suchen Sie die Motive von **P13346** in der Aminosäuresequenz auf der UniProt Seite.

Aufgabe 4.3: PROSITE

PROSITE (http://www.expasy.org/prosite) ist eine Datenbank für größere Motive, die eher Proteinfamilien, Domänen und funktionellen Bereichen zugeordnet werden können.

Aufbau von PROSITE–Signaturen:

[ ]	Aminosäuren, die an dieser Position vorkommen können
–	nächste Position
x	beliebige Aminosäure
( )	Anzahl der Wiederholungen
{ }	Aminosäuren, die an dieser Position nicht vorkommen dürfen

a) Um die PROSITE–Signatur für **P13346** zu finden, gehen Sie wieder zu *Family & Domains* auf der UniProt Seite von **P13346**. Dort finden Sie 3 PROSITE–Links, die alle zu der gesuchten Information führen, wobei der erste Link nie direkt. Für **P13346** wird ein *Domain Profile* und ein dazugehöriges *Domain Pattern* gefunden. Das *Domain Pattern* hat die gesuchte Signatur (*Consensus Pattern*).

b) Notieren Sie sich die Accession und das Consensus Pattern der Signatur.

c) Welches PRINTS-Motiv von **P13346** stimmt annähern mit der PROSITE–Signatur überein?

Aufgabe 4.4: TOPCONS

TOPCONS https://topcons.cbr.su.se/pred/ ist ein Vorhersagetool für die Topologie von Transmembranproteinen.

a) Suchen Sie die Sequenz des menschlichen Proteins **Q6PML9** auf der UniProt Seite und geben Sie sie auf der TOPCONS Seite ein.

b) Welche Vorhersagetools werden von TOPCONS verwendet? Was ist **3j1zP**?

c) Wie viele Transmembranhelices werden von den jeweiligen Tools vorhergesagt?

Aufgabe 4.5: PSSMs mit Pandas DataFrames

In dieser Aufgabe benutzen wir Python, um die PSSM und Konsenssequenz aus der vorherigen Aufgabe zu berechnen.

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) **Libraries:** Bibliotheken (auch *Modules* oder *Packages* genannt) bieten zusätzliche Funktionalität und können nach Bedarf in einem Python Skript mit dem Schlüsselwort *import* geladen werden.

```
import math
import pandas
```

Kopieren Sie den Code in eine leere Zelle und führen ihn aus. Dies lädt die Bibliothek *Math* für weitere mathematische Funktionen sowie *Pandas*, eines der populärsten Bibliotheken für Datenanalyse und -manipulation.

c) Laden Sie Datei **alignment_matrix.tsv** von der Vorlesungsseite herunter. Gehen Sie zu Ihrem Notebook und klicken Sie rechts auf **+ Add data** und dann auf **Upload** oben rechts. Ziehen Sie die Datei in das sich öffnende Fenster oder benutzen Sie den Dateienbrowser. Nennen Sie den Datensatz **Alignment** und clicken Sie auf **Create**.

Sie können nun die Datei oben rechts unter **input** → **alignment** → **alignment_matrix.tsv** finden. Wenn Sie die Maus über den Dateinamen bewegen, bekommen Sie die Option, den Dateipfad zu kopieren.

d) **DataFrames:** DataFrames sind Datentabellen, für die in Pandas bereits viele Analysefunktionen implementiert sind. Um die Alignment-Matrix in einen DataFrame zu laden, benutzen wir die Pandas-Funktion *read_csv()* und geben ihr den Dateipfad. Weiterhin benutzen wir den Parameter für den Spaltenseparator (*sep*), um Pandas mitzuteilen, dass die Spalten der Matrix durch Tabs und nicht durch Kommas getrennt sind. Mit *index_col=0* machen wir die Aminosäuren zum Index der Tabelle, wodurch nur die Zahlen in den anderen Spalten als Datenpunkte verwendet werden.

```
alignment_matrix = pandas.read_csv('../input/alignment/alignment_matrix.tsv',
                                   sep='\t', index_col=0)
alignment_matrix
```

e) Man kann sich nun leicht die Summe aller Spalten und Zeilen mit der *sum()*–Funktion anzeigen lassen. Standardmäßig wird die Summe der Spalten (Achse 0) gebildet, aber man kann die Zeilenachse (Achse 1) angeben, um die Summe der Zeilen zu bekommen.

```
print('Column sums:')
print(alignment_matrix.sum())
print('\nRow sums:')
print(alignment_matrix.sum(axis=1))
```

f) In dem Beispiel wissen wir, dass wir ein Alignment aus 6 Sequenzen haben. Damit unser Code aber auch funktioniert, wenn wir eine Alignment-Matrix aus mehr oder weniger Sequenzen hochladen, berechnen wir die Anzahl der Sequenzen mit dem DataFrame. Dafür zählen wir einfach die erste Spalte zusammen:

```
number_sequences = alignment_matrix['Pos 1'].sum()
print('Number of sequences, N:', number_sequences)
```

g) Nun erstellen wir eine Funktion, die den positionsspezifischen Score einer Zelle in der Matrix mit der Formel von oben berechnet. *Math.log* ist dabei der natürliche Logarithmus *ln* und *round(score, 2)* rundet den Score auf 2 Dezimalstellen nach dem Komma.

Damit wir die Funktion nicht explizit von Hand auf jede Zelle unseres DataFrames anwenden müssen, benutzen wir die Funktion *applymap()*, die dies automatisch für uns macht. Dabei wird ein neuer DataFrame erstellt, den wir in einer neuen Variable speichern und der die fertige PSSM enthält.

```
def pssm_score(cell):
    counter = cell + 0.05
    denominator = (number_sequences + 1) * 0.05
    score = math.log(counter / denominator)
    return round(score, 2)

pssm_matrix = alignment_matrix.applymap(pssm_score)
pssm_matrix
```

h) Mit dem PSSM–DataFrame können wir uns nun die Konsenssequenz berechnen lassen. Die *max()*–Funktion berechnet den maximalen Wert in jeder Spalte. Für die Konsenssequenz möchten wir aber natürlich nicht nur den maximalen Wert wissen, sondern brauchen die dazugehörige Aminosäure. Da wir die Aminosäuresspalte ganz am Anfang zum Index gemacht haben, können wir die *idxmax()*–Funktion dafür benutzen. Genau wie *max()* berechnet sie den maximalen Wert, gibt aber nicht den Wert sondern den dazugehörigen Index zurück.

```
print('Maximum scores:')
print(pssm_matrix.max())
print('\nConsensus sequence:')
print(pssm_matrix.idxmax())
```

Viel Spaß!