Softwarewerkzeuge der Bioinformatik

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

Saarland University
Center for Bioinformatics

Exercise Sheet 4

Sequence Analysis: Motifs and Transmembranehelices

Learning objective: *In this tutorial you will get to know two databases for sequence motifs and a prediction tool for transmembrane helices. In addition, you are going to construct a PSSM and the corresponding consensus sequence by hand and with Python.*

Exercise 4.1: Position–specific scoring matrix (PSSM)

Compute the PSSM and consensus sequence of the following alignment. (X represents all amino acids not listed here.)

```
S E Q V E N C E
S E Q Y E M C E
S I Q V I N S E
S E Q V E - C E
S E Q V E M C I
S E Q V E N S I
*   *       .
```

consensus sequence:

alignment → matrix

	1	2	3	4	5	6	7	8
C								
E								
I								
M								
N								
Q								
S								
V								
Y								
X								

PSSM

$$s_{i,j} = \ln \frac{n_{i,j}+p_i}{(N+1)\cdot p_i}$$

Parameter:

- $N = 6$: number of sequences
- $n_{i,j}$: number of amino acid i at position j
- $p_i = 0.05$: A priori probability of amino acid i. This is the simple model in which all amino acids habe the same probability.

Exercise 4.2: PRINTS

PRINTS (http://bioinf.man.ac.uk/dbbrowser/PRINTS/index.php) is a database for protein fingerprints, meaning groups of smaller, conserved motifs.

a) Look up the murine protein FosB (**P13346**) in PRINTS. For this it is easier to first look up the protein in UniProtKB, to click there on *Family & Domains* and then the PRINTS link.

b) What is the accession of the fingerprint? How many motifs does it include?

c) How many proteins were used to create the motifs? To answer the question, either read the section *Documentation* or count the number of rows for one of the motifs in the section *Initial Motifs*.

d) Does **P13346** contain all motifs? For this you can search with the alternate UniProt accession (**FOSB_MOUSE**) in the section *Final Motifs*.

e) The start position of a motif on the PRINTS site is given for each protein in the column *st*. Find the motifs of **P13346** in the amino acid sequence on the UniProt site.

Exercise 4.3: PROSITE

PROSITE (http://www.expasy.org/prosite) is a database for larger motifs belonging to protein families, domains and functional sections.

Composition of PROSITE signatures:

[ ]	amino acids that can occur at this position
–	next position
x	any amino acid
( )	number of repititions
{ }	amino acids that are not allowed at this position

a) To find the PROSITE signature of **P13346**, return to *Family & Domains* on the UniProt site of **P13346**. There you can find 3 PROSITE links that all lead to the desired information, though the first link never directly. For **P13346** there is domain profile and a corresponding domain pattern. The domain pattern has the desired signature (consensus pattern).

b) Note down the accession and the consensus pattern of the signature.

c) Which PRINTS motif of **P13346** corresponds to the PROSITE signature?

Exercise 4.4: TOPCONS

TOPCONS https://topcons.cbr.su.se/pred/ is a prediction tool for the topology of transmembrane proteins.

a) Search for the sequence of the human protein **Q6PML9** on the UniProt site and input it on the TOPCONS site.

b) Which prediction tools are used by TOPCONS? What is **3j1zP**?

c) How many transmembrane helices are predicted by each tool?

Exercise 4.5: PSSMs with Pandas DataFrames

In this exercise we are going to use Python to compute the PSSM and consensus sequence from the previous exercise.

a) Go to kaggle.com and log in. Create a new notebook, rename it, and delete the already existing cell.

b) **Libraries:** Libraries (also called *modules* or *packages*) offer additional functionality and can be loaded in Python script as needed by using the keyword *import*.

```
import math
import pandas
```

Copy the code into an empty cell and run it. This will load the library *math* for additional mathematical functions as well as *pandas*, one of the most popular libraries for data analysis and data manipulation.

c) Download the file **alignment_matrix.tsv** from the lecture website. Go to your notebook and on the righthand side click on **+ Add data** and then on **Upload** in the top right. Drag and drop the file into the window or use the file browser. Call the data set textbfAlignment and click on **Create**.

You can now find the file in the top right under **input** → **alignment** → **alignment_matrix.tsv**. When you hover with your mouse over the file name, it will give you the option to copy the file path.

d) **DataFrames:** DataFrames are data tables for which pandas already implemented many analysis functions. To load our alignment matrix into a DataFrame, we use the pandas function *read_csv()* and give it the file path. In addition, we use the paramter for the column separator (*sep*) that tells pandas that the columns in the matrix are separated by tabs and not commas. With *index_col=0* we make the amino acid column the index of the table so that only the numbers in the other columns are going to be used as data points.

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

e) We can now easily display the sum of all columns and rows with the *sum()* function. By default, this will compute the sum of the columns (axis 0), but we can input the axis of the rows (axis 1) to get the sum of the rows.

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

f) In our example we know that the alignment consists of 6 sequences. To make our code also work when we upload a different alignment matrix with more or less sequences, we compute the number of sequences with the DataFrame. For this we simmply sum up the first column:

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

g) Now we define a function that computes the position-specific score of a cell in the matrix with the formula in the exercise above. *Math.log* gives the natural logarithm *ln* and *round(score, 2)* rounds the score to the 2 decimal places after the comma.

To avoid having to call the function explitely by hand on each cell of our DataFrame, we use the function *applymap()* that does automatically for us. This creates a new DataFrame that we assign to a new variable and that contains the finished PSSM.

```
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) With the PSSM DataFrame we can now compute the consensus sequence. The *max()* function computes the maximal value of each column. However, for the consensus sequence we need not just the maximum value but also the actual amino acid. Since we set the index of the table to the amino acid column at the very beginning, we can now use the *idxmax()* function. Just like *max()* it computes the maximum value but returns the corresponding index instead of the value.

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

Have fun!