Python programming exercises

Giuseppe Profiti (University of Bologna)

TT Teresa Tavella (University of Bologna)

adjust: 1. basic 2. string 3. list 4. matrix 5. reading matrix from file 6. expression matrices (4.15)

1 Basics

1.1 Roots of equation

Write a Python function that given the coefficients a,b and c, evaluates the roots, if any, of a second degree equation:

$$ax^2 + bx + c = 0$$

Please remember that the roots of the equation are evaluated as:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

1.2 Scalar by point

Write a Python function that given a number a and a 2 dimensional point $\mathbf{p}$ returns their product:

$$f(a, \mathbf{p}) = a \cdot \mathbf{p} = a \cdot < p_x, p_y > = < a \cdot p_x, a \cdot p_y >$$

Please clearly state your choice for the data structure used for p.

1.3 Distance

Write a Python function that returns the distance of two points in a 3-dimensional space.

$$\text{distance}(\mathbf{a}, \mathbf{b}) = \|a - b\| = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2}$$

Please clearly state your choice for the data structure used for $\mathbf{a}$ and $\mathbf{b}$.

1.4 Bernstein basis

Write a Python function that evaluates the Bernstein basis polynomial $b_{i,n}(t)$ given an index i, a number n and a parameter t,

$$b_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

where $\binom{n}{i}$ is the binomial coefficient. Assume that there is a function computing the factorial $fact(n)$.

1.5 Even or odd

Write a Python function evaluating if a number is even or odd. Motivate your choice of the return value.

2 Parsing file

Having a *gff* (an example of file is shown below, where each uniprot ID identifies a new line)
Q9C000 UniProtKB Mutagenesis 339 340 . . . Note=Loss of ATP binding. GK− >EA;Ontology_term=ECO:0000269;evidence=ECO:0000269—
PubMed:15212762;Dbxref=PMID:15212762GKEA;Ontology_term=ECO:0000269;evidence=ECO:0000269—PubMed:15212762;Dbxref=PMID:15212762
Q9C000 UniProtKB Mutagenesis 340 340 . . . Note=No effect. K− >L

1. Find the lines containing the field *'Mutagenesis'*.
2. Identify the position where only single point mutation occur. This information is contained after the word *'Mutagenesis'*, you have *'340 340'* the start and the end of the mutation.
3. in the field *'Note=[...]'* of the same line, identify the mutation generally written as *'K− >L'*, where *'K'* is the wild type and *'L'* is the mutated residue. Note: Consider only the single point mutations.

3 Lists and loops

3.1 Scalar by vector

Write a Python function that given a number a and a list of values $\mathbf{p}$ returns their product:

$$f(a, \mathbf{p}) = a \cdot \mathbf{p} = a \cdot < p_1, \dots, p_n > = < a \cdot p_1, \dots, a \cdot p_n >$$

3.2 Bernstein polynomial

Write a Python function that evaluates the Bernstein polynomial $B_n(x)$ given a polynomial degree n, a list of coefficients β and a point x:

$$B_n(x) = \sum_{i=0}^{n} \beta_i b_{i,n}(x)$$

3.3 Frequency (DNA)

Write a Python function that, given a list containing characters representing DNA bases (A,C,G,T), evaluates the frequency of each character.

3.4 Frequency

Write a Python function that, whose content is unknown, evaluates the frequency of its elements.

3.5 Mode

Write a Python function that evaluates the mode of a vector. (Hint: it's the most frequent value/s)

3.6 Average

Write a Python function computing the mean of a vector of values ($\mathbf{x} = x_1, \ldots, x_n$).

$$\overline{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)$$

3.7 Factorial

Write a Python function that given a number n computes its factorial $n!$. Please remember that:

$$n! = \prod_{t=1}^{n} t$$

Try to solve this problem once with recursion and once with iteration.

3.8 Median

Write a Python function computing the median of a vector of values ($\mathbf{x} = x_1, \ldots, x_n$).

(Hint: check the order of the vector; n is the length)

$$\widetilde{x} = \begin{cases} x_{[\frac{n+1}{2}]}, & \text{if } n \text{ is odd} \\ \\ \frac{1}{2}\left(x_{[\frac{n}{2}]} + x_{[\frac{n}{2}+1]}\right), & \text{if } n \text{ is even} \end{cases}$$

3.9 Variance

Write a Python function computing the variance of a list of values.

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x-\overline{x})^2$$

3.10 Standard deviation

Write a Python function computing the standard deviation of a list of values.

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2}$$

3.11 Z-score

Write a Python function computing the z-scores of a vector of values ($\mathbf{x} = x_1, ..., x_n$).

$$z_{x_i} = \frac{x_i - \overline{x}}{\sigma}$$

3.12 Sum of two vectors

Write a Python function that given two vectors $\mathbf{a}$ and $\mathbf{b}$ returns their sum $\mathbf{c}$, where

$$c_i = a_i + b_i$$

3.13 Dot product

Write a Python function that given two vectors $\mathbf{a}$ and $\mathbf{b}$ returns their dot product, where

$$c = \sum_{i=1}^{n} a_i \cdot b_i$$

3.14 Root mean square

Write a Python function that, given two sets of points in a 3-dimensional space, evaluates the root mean square deviation between them. (i.e. the RMSD of an alignment of two structures from PDB)

$$\mathrm{RMSD}(\mathbf{v},\mathbf{w}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|v_i - w_i\|^2}$$

3.15 Entropy

Write a Python function that, given a string s and a probability distribution of characters, evaluates the entropy H of s.

$$\mathrm{H}(s) = -\sum_{i=1}^{n} p(s_i)\log p(s_i)$$

3.16 Covariance

Write a Python function that given two vectors $\mathbf{X}$ and $\mathbf{Y}$, evaluates their covariance.

$$\sigma_{X,Y} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{N}\sum_{i=1}^{N} x_i y_i - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)\left(\frac{1}{N}\sum_{i=1}^{N} y_i\right)$$

3.17 Independent sample t-test

Assuming you have two independent populations, a control(A) and case group(B), write a Python function computing the t-value.

$$t = (\overline{x}_A - \overline{x}_B)\left(\frac{\sigma_A}{\sqrt{n_A}} + \frac{\sigma_B}{\sqrt{n_B}}\right)^{-1}$$

3.18 Chi-square

Write a Python function computing the chi-square test given a vector of observed (o) and expected (e) values.

$$\chi^2 = \sum_{i=1}^{n}\frac{(o_i - e_i)^2}{e_i}$$

3.19 Clustering with threshold

Having a vector of points ($\mathbf{p} = p_1, \ldots, p_n$) in 1D space, write a Python function computing the set of closest points for each p$_i$ with threshold w.

4 Matrices

4.1 Sum of matrix row

Write a Python function that, given a matrix $A \in \mathbb{R}^{n,m}$, evaluates the sum of the elements in each row:

$$s_i = \sum_{j=1}^{m} a_{i,j}$$

4.2 Sum of two matrices

Write a Python function that, given two matrices $A \in \mathbb{R}^{n,m}$ and $B \in \mathbb{R}^{n,m}$, evaluates their sum $C \in \mathbb{R}^{n,m}$

$$c_{i,j} = a_{i,j} + b_{i,j}$$

4.3 Determinant

Write a Python function that given a square matrix evaluates its determinant. Please remember that given a matrix A with elements a_{ij} the determinant is given by

$$\det(A) = |A| = \sum_{i=1}^{n} \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \, |M_{ij}|$$

where M_{ij} is the minor of the matrix, i.e. the matrix A without row i and column j.

4.4 Read substitution matrix

Write a Python function that reads a substitution matrix from a file and returns a convenient data structure containing it. The file is formatted as elements separated by comma, with the first row and the first column as labels. An example:

```
,A,B,C,D
A,3,-2,0,1
B,-2,3,1,-1
C,0,1,3,2
D,1,-1,2,3
```

Note: the substitution matrix is square and it can have any size and labels.

4.5 Threshold

Write a function that given a matrix (stored in a data structure of your choice) and a threshold value, returns how many values are greater or equal to the threshold per each row.

Example:

$$f\left(M = \begin{vmatrix} 3 & 1 & -5 \\ 4 & 7 & 18 \\ -5 & 6 & 4 \end{vmatrix}, \text{threshold} = 3\right) \rightarrow [1, 3, 2]$$

4.6 Alignment score

Write a Python function that given two aligned sequences and substitution matrix, evaluates the score of the alignment between the two sequences. Gaps are marked with the - (minus) character. The score for any character aligned to a gap is by default 0.

Example: s1 = “AABA-A”, s2=“AAABBA”

$$M = \begin{array}{ccc} & \text{A} & \text{B} \\ \text{A} & 7 & -3 \\ \text{B} & -3 & 7 \end{array}$$

```
A  A  B  A  -  A
A  A  A  B  B  A
7 +7 -3 -3  0 +7 = 15
```

The result score is then 15.

Note: the substitution matrix and the sequences can have any number of characters (i.e. genomic bases, residues etc). The sequences contain only letters in the matrix or the gap character.

4.7 Frobenius Inner Product

Write a Python function calculating the Frobenius Inner Product of two matrices (A, B).

$$\langle A, B \rangle_F = \sum_{i,j} a_{i,j} b_{i,j} \tag{1}$$

5 Text

5.1 Palidrome

Check if a string is palindrome without using predefined methods.
Example ***ATGCGTA***

5.2 Anagram

Having two strings, check if the first one is the anagram of the second.
Example:
$seq_1 = \text{ATGACG}$
$seq_2 = \text{GATCAG}$

5.3 Sequence type

Having a sequence, check whether it is a ribonucleic acid sequence or nucleic acid. If it is RNA, return ’RNA’, if it is DNA, return ’DNA’.

5.4 Translation

Write a function translating a sequence into a protein sequence until the first stop codon (use the first frame). (The DNA codon table is stored in a dictionary)

- DNA stop codons are “taa”, “tag”, “tga”
- RNA stop codons are “uag”, “uaa”, “uga”

5.5 FASTA file reading

Write a Python function that given a file name reads its content. Assume that it should contain a sequence in FASTA format.

5.6 FASTA file reading (2)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. Compute the length of the sequences and store them (describe the data structure you are choosing).

5.7 FASTA file reading (3)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. Calculate the average length of the sequences in the dataset and return the headers of the shortest and the longest sequences.

5.8 FASTA file reading (4)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. The header of each sequence is in the form: $>$’accession number’|’Genus species’| ’protein name’. Split the header and retrieve the ”Genus species” field. Store this information in a list without duplicates.

5.9 Fasta file manipulation (5)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. The header of each sequence is in the form: $>$’accession number’|’Genus species’| ’gene name’. Store the information on the genus in a dictionary and count how many sequences for each genera there are in the dataset.

5.10 Fasta file manipulation (6)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. The header of each sequence is in the form: $>$’accession number’|’Genus species’| ’gene name’. Split the header and retrieve the ”gene name” field. Store this information in a list without duplicates.

5.11 Fasta file manipulation (7)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. The header of each sequence is in the form: $>$’accession number’|’Genus species’| ’gene name’. For each Genera in the dataset, return a list of gene names associated to them.

5.12 Fasta file manipulation (8)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. The header of each sequence is in the form: $>$’accession number’|’Genus species’| ’gene name’. Check if there are sequences of the species ’D. melanogaster’ and how many.

5.13 Fasta file manipulation (9)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. Write the $\%$GC content of the dataset.

5.14 Fasta file manipulation (10)

Write a Python function that given a file name reads its content. Assume that it contains multiple sequences in FASTA format. Count the number of sequences starting with ’ATGC’.

5.15 Fasta file manipulation

You have a multifasta sequence file. Write functions for:

1. Counting how many fasta sequences there are in the file.
2. For each sequence store the length (describe the data structure you are choosing).
3. Calculate the average length of the dataset, return the shortest and the longest sequence and header.
4. Count how many sequences there are for each organism: the header is in the form: $>$'accession number'$|$'Genus species'$|$ 'gene name'. example: $>$P04412$|$Drosophila melanogaster$|$EGFR
5. Check if there are sequences of the species 'melanogaster' and how many.
6. Write the $\%$GC content of the dataset.
7. Count the number of sequences starting with 'ATGC'.

5.16 Expression matrix

Having a matrix with N rows representing genes and M column representing samples.
Write a Python function to normalize the expression matrix with quantile normalization.
Algorithm steps:

1. From lowest to highest value in each column assign a rank.
   Example:

$$G = \begin{pmatrix} 1 & 3 & 4 & 5 \\ 7 & 8 & 1 & 2 \\ 3 & 4 & 5 & 6 \\ \end{pmatrix},$$

$$R = \begin{pmatrix} i & i & ii & ii \\ iii & iii & i & i \\ ii & ii & iii & iii \\ \end{pmatrix}$$

   Save the rank translated matrix, you will use it later.

2. Sort the values in each sample/column from the lowest to the highest.

$$G_{sort} = \begin{pmatrix} 1 & 3 & 1 & 2 \\ 3 & 4 & 4 & 5 \\ 7 & 8 & 5 & 6 \\ \end{pmatrix}$$

3. Compute for each gene/row the mean value. Rank the means from the lowest to the highest.
   mean$_{row1}$ = (1+3+1+2)/4 = 1.75 $- >$ i
   mean$_{row2}$ = (3+4+4+5)/4 = 4 $- >$ ii
   mean$_{row3}$ = (7+8+5+6)/4 = 6.5 $- >$ iii

4. Take the ranked matrix (R) as reference and substitute the ranks with the mean values corresponding to that rank, as calculated in point '3'. Return the normalized matrix.

$$G_{qnorm} = \begin{pmatrix} 1.75 & 1.75 & 4 & 4 \\ 6.5 & 6.5 & 1.75 & 1.75 \\ 4 & 4 & 6.5 & 6.5 \\ \end{pmatrix}$$

5.17 Motif

Write a Pyhton function returning all the number of occurrences and the starting positions of a motif in a word of length N.

6 Operations on Dictionaries, lists and strings

6.1 Nucleotide frequencies

Write a Python function evaluating the frequency of the four nucleotides in a string of DNA.

6.2 Intersection

Write a Python function performing the intersection of two lists.

6.3 Range

Write a Python function returning the sum of elements in the range 1 to 10.

6.4 Duplicates

Given a list write a Python function returning the list without duplicates.

6.5 search in string

Write a Python function checking for the word 'happy' is in the string 'Bioinformaticians are HAPPY'.

6.6 String joining

Write a Python function returning a string obtained from the elements of the following list ['This', 'is', 'an', 'exercise'].

6.7 String splitting

Write a Python function splitting the string x = 'split;the;string', by ';' and returning the 1st character of the splitted string.

6.8 Summing elements in list

Write a Python function returning the sum of the elements in the list ['1','10','5','6','7','8'].

6.9 Returning elements in list

Write a Python function returning the first three elements in the list ['1','10','5','6','7','8'].

6.10 List slicing

Write a Python function slicing a list every three positions and return the sublists.

6.11 Summing dictionaries values

Write a Python function returning the sum of the values of keys A and B in the dictionary d = {A : ’2’, B: ’4’, C:’7’}.

6.12 Multiplication of dictionary values

Write a Python function returning the product of the values in the previous dictionary (d) and join the keys in a string interspaced by ’*’.

6.13 Empty strings

Write a Python function checking if a string is empty.

6.14 Reversing a string

Write a Python function returning a string in reverse order.

6.15 Digits and letters

Write a Python function returning the numbers of digits and letters in a string. (Describe the data structure you decided to use)

6.16 Upper and lower cases

Write a Python function returning the numbers of lower and upper cases in a string.

6.17 Square of odd values

Write a Python function returning the square of the odd values in a list.

6.18 Check value of key in dictionary

Write a Python function returning the value of key ’year’ in a dictionary.

6.19 Dictionary updating

Write a Python function returning a dictionary updated by adding 3 to the value of key ’A’. (What if ’A’ is not a key of the dictionary?) Add key ’D’ with value 4 to the dictionary.

6.20 Dictionary looping

Write a Python function to iterate through the key and the values of the dictionary.

6.21 Summing values in dictionary

Write a Python function returning the sum of all the values in a dictionary.

6.22 Multiplying values in dictionary

Write a Python function returning the product of all the values in a dictionary.

6.23 Square of dictionary values

Write a Python function returning the square of all the values in a dictionary.

6.24 Sorting

Write a Python function returning the sorted list of the keys of a dictionary.

6.25 Merge dictionary

Write a python function merging two dictionaries by summing up the values if the key is the same.

6.26 Unique values in dictionary

Write a Python function returning a list of unique values in a dictionary.

6.27 Highest and lowest values in dictionary

Write a Python function returning the highest and the lowest values in a dictionary.

6.28 Square of keys

Write a Python function generating a dictionary where the keys are numbers from 1 to 10 and the values are the square of the keys.

6.29 Tuples

Write a Python function generating the squares of values between 1 and 10, store the values in a list of tuples, each tuple store a value and its square.

7 More

7.1 Body Mass Index

Write a Python function evaluating the BMI from a set of individuals. The data are stored in a dictionary where each individual is the key of the dictionary and its value is a tuple of height (h) and weight (w).

$$BMI_i = \frac{w_i}{h_i^2} \tag{2}$$

7.2 MA plot

Having two lists one for a control sample (x = 1,..n) and one for a case sample (y = 1,..n), where each position in the lists is the expression value of a gene, write a Python function calculating the M-values (the ratio of the expression levels of a gene between two cases) and the A-values (the average of the expression levels of a gene between cases) for every gene 'i'. (instead of two separated list you can think of a matrix)

$$M_i = \log_k\left(\frac{xi}{yi}\right); A_i = \frac{1}{2}\left(\log_k xi + \log_k y_i\right) \tag{3}$$

6.3 Range

Define the output of the following ranges (5,0,-1), (1,10), (5,-5,-1), (0,10,2).