LAB 4 – Probabilistic Models

Authors :

	Ricardo Rei	78047
	Luis Henriques	77919
	Luis Nunes	77940

Question 1:

CpG islands are found in certain regulatory regions of the genome, including the promoter regions of many housekeeping genes such as the phosphoglycerate kinase gene. Since DNA methylation is involved in the repression of gene expression, it is usually not seen in association with housekeeping genes, which are expressed in all tissues. In tissue-specific genes, CpG islands are much less common, probably because these genes are frequently methylated, resulting in potential loss of CpG dinucleotides due to mutation. There are about 30000 CpG islands in the human genome. The fact that many are associated with genes suggests that CpG islands might be useful in locating genes within DNA sequences. (M. Goldman 2001)

a) **CpGPlot**: Is an on-line tool that identifies and plots CpG Islands in nucleotide sequence(s).

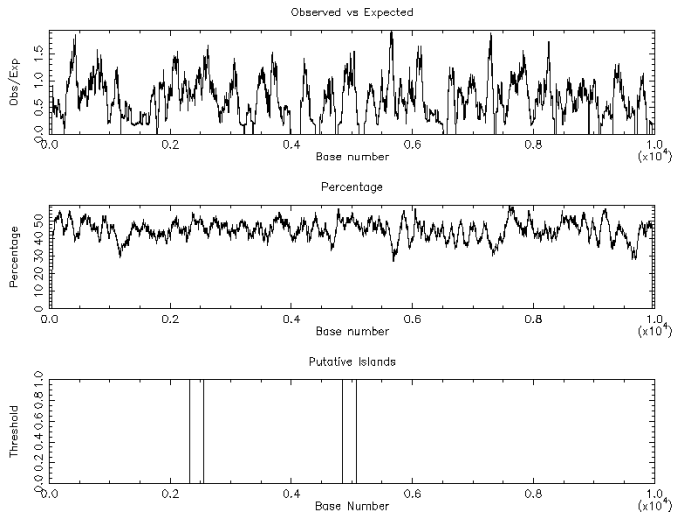
In order to do this calculation, a sequence (or a file) is uploaded. Then, the percentage of CG content and the Observed frequency of CG is calculated within a window whose size can be defined by us. The windows are then moved along the sequence in order to calculate the previous mentioned statistics at each position that the windows are moved to.

Beyond the window size there are three more important parameters:

- Minimum length of an island: This sets the minimum length that a CpG island has to be before it is reported.
- Minimum observed/expected: This sets the minimum average observed to expected ratio of C plus G to CpG in a set of 10 windows that are required before a CpG island is reported.
- Minimum percentage: This sets the minimum average percentage of G plus C a set of 10 windows that are required before a CpG island is reported.

When changing these parameters to obtain different results we have to take in consideration certain limitations. For example, in the window size, setting a too small size can cause two close CpG-Islands to be treated as one. And by other hand, setting a too large value can cause small CpG-Islands to be identified as part of larger CpG-Island. The minimum length of an island delineates the size that a genome has to have so it can be classified as being a CpG-Island. Since CpG-Islands are mostly found in rRNA and tRNA regions (with high replication rates), this parameter shouldn't be too high so that we can find more regions.

Below we present the graphical results of running the tool with the *genome.txt* file and with the standard parameters (Window size=100, Minimum length=200, Minimum observed=0.6, Minimum percentage=50):



Two CpG-Islands were identified. If we look closely we can see reflected in the third graph a match of the zones in the first graph where the *Obs/Exp* values are higher than 50 with the zones in the second graph where the *Percentage* is higher than 0.6.

```
CPGPLOT islands of unusual CG composition
EMBOSS_001 from 1 to 10009

     Observed/Expected ratio > 0.60
     Percent C + Percent G > 50.00
     Length > 200

 Length 221 (2326..2546)

 Length 232 (4840..5071)
```

Below we can see the results outputted that tell us the exact size of each CpG-Island (221bp and 232bp) and the where exactly they are in the sequence (2326..2546 and 4840..5071).

After changing the value parameters (images in annex) we conclude the following:

- Decreasing the window value (to 50) makes the number of CpG-Islands decrease to 0;
- Increasing the window value (to 200) makes the number of CpG-Islands increase to 4;
- Decreasing the length (to 100) makes the number of CpG-Islands increase to 10;
- Increasing the window value (to 300) makes the number of CpG-Islands decrease to 0;
- Increasing the observed/expected ratio to 1.0 makes the number of CpG-Islands decrease to 0.

All the results are self-explanatory and match what we expected to see based on the previous description of the parameters.

CpG Islands: Is an online tool that identifies potential CpG island regions. The calculation is performed using a 200 bp window moving across the sequence at 1 bp intervals. CpG islands are defined as sequence ranges where the Obs/Exp value is greater than 0.6 and the GC content is greater than 50%. The expected number of CpG dimers in a window is calculated as the number of 'C's in the window multiplied by the number of 'G's in the window, divided by the window length. CpG islands are often found in the 5' regions of vertebrate genes, therefore this program can be used to highlight potential genes in genomic sequences.

Bellow we present the initial result of running the *genome.txt* in the tool (the full result can be seen in annex):

```
Results for 10009 residue sequence "Untitled" starting "GCTATCGTAG"

CpG island detected in region 160 to 359 (Obs/Exp = 0.63 and %GC = 50.50)

CpG island detected in region 161 to 360 (Obs/Exp = 0.63 and %GC = 50.50)

CpG island detected in region 162 to 361 (Obs/Exp = 0.62 and %GC = 51.00)

CpG island detected in region 163 to 362 (Obs/Exp = 0.68 and %GC = 51.50)

CpG island detected in region 164 to 363 (Obs/Exp = 0.69 and %GC = 51.00)

CpG island detected in region 165 to 364 (Obs/Exp = 0.71 and %GC = 50.50)

CpG island detected in region 168 to 367 (Obs/Exp = 0.71 and %GC = 50.50)

CpG island detected in region 170 to 369 (Obs/Exp = 0.71 and %GC = 50.50)

CpG island detected in region 171 to 370 (Obs/Exp = 0.71 and %GC = 50.50)

CpG island detected in region 172 to 371 (Obs/Exp = 0.71 and %GC = 50.50)
```

In the full result we can see that the 1st result present in the preview tool (2326..2546) is split in multiple smaller CpG-Islands.

DNA Stats: DNA Stats returns the number of occurrences and percentage totals for each residue in the sequence. For certain groups of residues, the tool lets us quickly compare the results obtained for different sequences, which is really useful when we are trying to find CpG-Islands. Below we show the results, that are relevant for CpG-Islands, outputted from the tool (the full output can be seen in the annex):

DNA Stats results

Results for 10009 residue sequence "Untitled" starting "GCTATCGTAG"

Pattern:	Times found:	Percentage:
g	1682	16.80
a	2964	29.61
t	2550	25.48
c	2813	28.10

gg	367	3.67
ga	519	5.19
gt	328	3.28
gc	468	4.68
gn	0	0.00
ag	587	5.87
aa	948	9.47
at	670	6.69
ac	759	7.58

Above we can see that G and C are approximately 46% of the sequence, although, only 4.68% of the sequence contains CG residues.

ORF Finder: Is an online tool that searches for open reading frames (ORFs) in the DNA sequence. The program returns the range of each ORF, along with its protein translation.

An open reading frame (ORF) is an interval of genetic material that allows genetic machineries that receive it as input to produce a protein. These machineries need the genetic material (DNA and RNA) that is written in code to tell them what to do, for example add another protein building block, although others tell the machine to *start* or *stop* the protein-building process. An ORF is an interval of genomic "letters" that falls between the start and stop signals.

Researchers can use ORFs for experimental purposes. While RNA interference technology allows scientists to "turn off" or silence certain genes, ORFs can be used to "turn on" or "overexpress" particular genes. Flipping the switches on genes one at a time can help reveal the functions of individual genes, such as those that play a role in cancer.

Specifically related with CpG-Islands, ORF are important to find CpG-Islands in areas of the sequence that at first glance appear to have an absent of CpG dinucleotides (Hisano M, Ohta H, Nishimune Y, Nozaki M. 2003).

Bellow we present the initial result of running the *genome.txt* in the tool (the full result can be seen in annex):

ORF Finder results
Results for 10009 residue sequence "Untitled" starting "GCTATCGTAG"

```
>ORF number 1 in reading frame 1 on the direct strand extends from base 142 to base 249.
ATGCCCCACAGTTCCCTTCTTGGGAACAAGGAGCGGGTATCAGGCACAATTCAACGATTA
GCCCAAGACACCTTGCTTAGCCACACCCTCAAGGGAACTCAGCAGTGA

>Translation of ORF number 1 in reading frame 1 on the direct strand.
MPHSSLLGNKERVSGTIQRLAQDTLLSHTLKGTQQ*

>ORF number 2 in reading frame 1 on the direct strand extends from base 2329 to base 2433.
ATGCCTTTGGTTGGGGCGACCGCGGGGAAACAAAAAACCCCCACGTGGAATGGGATAACC
TGTCCTCCAAACAAGAGCCCCCGCTCTAAAAAACAGAACATCTGA

>Translation of ORF number 2 in reading frame 1 on the direct strand.
MPLVGATAGKQKTPTWNGITCPPNKSPRSKKQNI*

>ORF number 3 in reading frame 1 on the direct strand extends from base 2857 to base 3132.
ATGATCTCAACCCTTATTACCCACGTGATCTCCCCCCTGGCTTTCATCGTCCCTATTCTT
CTGGCAGTAGCCTTCCTCACCTTAGTTGAACGTAAAGTATTAGGCTACATGCAACTACGA
AAAGGGCCAAATGTTGTAGGTCCATATGGACTCTTACAACCAATCGCAGACGGGGTAAAA
CTTTTCATTAAAGAGCCCGTACGACCATCCACATCTTCCCCACTACTATTCTTATTAACC
CCGATACTTGCTCTAACACTAGCAATAACCCTATGA

>Translation of ORF number 3 in reading frame 1 on the direct strand.
MISTLITHVISPLAFIVPILLAVAFLTLVERKVLGYMQLRKGPNVVGPYGLLQPIADGVK
LFIKEPVRPSTSSPLLFLLTPILALTLAITL*
```

b) In order to compare the genomic sequence presented in the file *genome.txt* with the ones available at the Genbank Nucleotide database, we use the BLAST tool available at NCBI (National Center for Biotechnology Information).

The Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between sequences. Since we are working with Nucleotides we are using the specific tool BLASTn that receives nucleotides sequences in the FASTA configuration and compares it to the sequence databases and calculates the statistical significance of matches.

After using BLASTn with the *Nucleotide collection (nr/nt)* database and optimize the program for *Highly similar sequences (megablast)* we obtain the following results:

Sequences producing significant alignments:

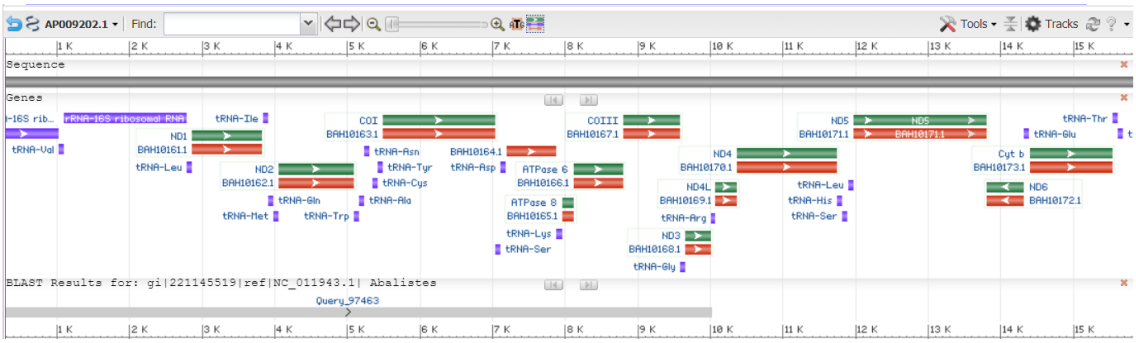
Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

Description	Max score	Total score	Query cover	E value	Ident	Accession
Abalistes stellaris mitochondrial DNA, complete genome	18484	18484	100%	0.0	100%	AP009202.1
Balistapus undulatus mitochondrial DNA, complete genome	12338	12338	99%	0.0	89%	AP009203.1
Rhinecanthus aculeatus mitochondrial DNA, complete genome	12059	12059	100%	0.0	88%	AP009210.1
Pseudobalistes flavimarginatus mitochondrial DNA, complete genome	11897	11897	100%	0.0	88%	AP009209.1
Odonus niger mitochondrial DNA, complete genome	11865	11865	99%	0.0	88%	AP009208.1
Balistes vetula mitochondrial DNA, complete genome	11814	11814	100%	0.0	88%	AP009204.1
Pseudobalistes fuscus mitochondrion, complete genome	11786	11786	100%	0.0	88%	KU985150.1

As we see in the results, the sequence present in the *genome.txt* has an identity of 100% and a 100% query cover with the *Abalistes stellaris mitochondrial* genome. This means the genome which we are been working is already present in the Genbank Nucleotide database and is not a novel one. We can also conclude that are a few more genomes present in this database that contain our sequence (Query cover=100%) but it also contains more information (Identity<100%).

c)Plotting the results obtained in the previous question (for the *Abalistes stellaris mitochondrial* genome) we obtain the following graph:



All the identified areas (as red and green in the image) contain the previous CpG-Islands identified by the multiple tools. This means, as expected, most genes can be mapped to ORFs and upstream CpG islands.

Question 2:

a)

Formally an HMM can be defined as tuple $(\mathcal{X}, \mathcal{Z}, \mathbf{P}, \mathbf{O})$ where:

- $\mathcal{X}$ represents the set of possible states.
- $\mathcal{Z}$ represents the set of possible observations/emissions.
- $\mathbf{P}$ is the transition probability matrix.
- $\mathbf{O}$ is the observation/emission probability matrix.

The graphical representation is presented in figure 2.1:

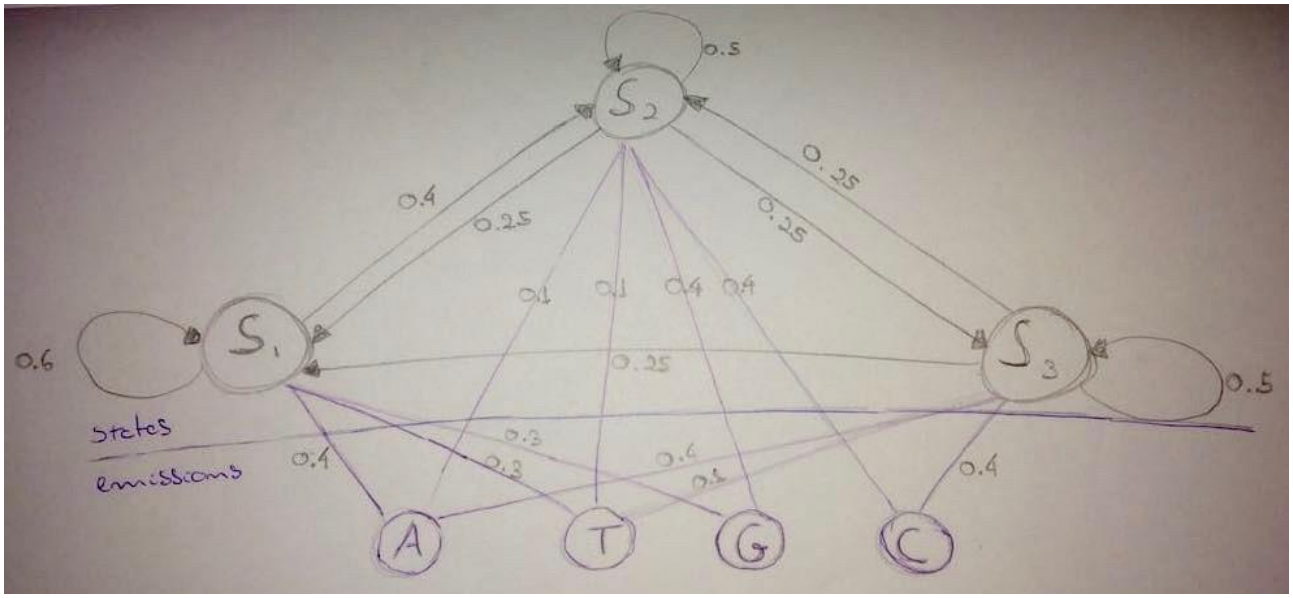


Figure 2.1 – Graphical representation of the HMM.

All HMM problems also have an initial distribution and a final distribution, but in this problem the initial and final distribution are equally distributed between the set of states. For this reason, we omitted both of them in the graphical representation.

b)

In this exercise we used Python 3.6 programming language and the numpy package 1.11.3. The source code can be found here: https://github.com/RicardoRei/Bioinformatics

In the file hmm.py we defined the HMM class that receives the tuple $(\mathcal{X}, \mathcal{Z}, \mathbf{P}, \mathbf{O})$ and optionally an initial distribution. This class has 2 main methods:

- viterbi – that receives a sequence of emissions and returns an index list of the most probable states defined in $\mathcal{X}$.
- forward – that receives a sequence of emissions and returns a vector with the probability of each state after observing that sequence of emissions.

To solve this exercise, we created an HMM class instance with $(\mathcal{X}, \mathcal{Z}, \mathbf{P}, \mathbf{O})$ model from the previous one and then called the Viterbi method with the argument seq. = CATGCGGGTTATAAC.

Viterbi output (the path that maximizes P(X|π):

S2 - S1 - S1 - S2 - S2 - S2 - S2 - S2 - S1 - S1 - S1 - S1 - S1 - S1 - S2

Given observation sequence $\boldsymbol{z}_{0:T}$

1. Multiply initial distribution by $\mathbf{O}(z_0 \mid :)$
2. At each time step:
   a. Multiply current distribution by $\mathbf{P}$
   b. Multiply by $\mathbf{O}(z_t \mid :)$

Figure 2.2 – Forward pseudo code used to implement Forward the algorithm (from ADI slides 16/17 prof. Francisco Melo).

Require: Observation sequence $z_{0:T}$

1. Initialize $\boldsymbol{m}_0 \leftarrow \operatorname{diag}(\mathbf{O}_{:,z_0})\boldsymbol{\mu}_0^\top$
2. **for** $\tau = 1, \ldots, T$ **do**
3. $\quad \boldsymbol{m}_t \leftarrow \operatorname{diag}(\mathbf{O}_{:,z_t}) \max\{\mathbf{P}^\top \operatorname{diag}(\boldsymbol{m}_{t-1})\}$
4. $\quad \boldsymbol{i}_t = \operatorname{argmax}\{\mathbf{P}^\top \operatorname{diag}(\boldsymbol{m}_{t-1})\}$
5. **end for**
6. $x_T^* = \operatorname{argmax}_{x \in \mathcal{X}} m_T(x)$
7. **for** $t = T - 1, \ldots, 0$ **do**
8. $\quad x_t^* = i_{t+1}(x_{t+1}^*)$
9. **end for**
10. **return** $\boldsymbol{x}_{0:T}^*$

Figure 2.3 – Viterbi pseudo code used to implement Viterbi algorithm (from ADI slides 16/17 prof. Francisco Melo).

c)

To compute the probability of a sequence being generated by a model we can use the Forward Algorithm.

The Forward algorithm is used to calculate a 'belief state' (the probability of a state at a certain time) given a set of observations and some model. In order to do this, we first calculate the probabilities over the states computed for the previous observation and use them for the current observations, extending it for the next steps using the transition probability table. This approach basically caches all the intermediate state probabilities so they are computed only once.

In the end we obtain the probability of each state given the set of observations and the probability of a certain sequence (set of observations) is given by the sum of the probability of each state divided by the number of states.

d)

To solve this exercise, we used the HMM instance created in exercise b) and called the method forward with the argument seq. = CATGCGGGTTATAAC.

Yet the result of the forward only gives the probability of each state after observing that sequence of emissions. In order to obtain P(X) we had to sum all entries of the vector resulting from running the forward method and then multiply the sum by 1/3.

According to our code P(X) = 3.12881999021e-10.