Análise detalhada dos *scripts* em BioPython para os genes MC4R e SEC16B

Bioinformática - Opção A

Mestrado em Engenharia Biomédica

Grupo 17: PG50255 Beatriz Silva Matos
PG50291 Catarina Rodrigues Pereira
PG50443 Isabel Maria Ferreira da Silva
PG50626 Mariana Fernandes Gonçalves

Data de entrega: 2 fevereiro, 2023

Índice

1 Análise da sequência e das *features* presentes no NCBI

Com o intuito de perceber melhor a influência destes genes na patologia da diabetes, foi necessário proceder a uma análise da sua sequência e *features*. Esta análise foi conseguida com o auxílio de scripts em BioPython.

Previamente a esta análise, foi realizado o *download* dos ficheiros, em formato *genbank*, através do NCBI, nos quais compilavam a informação acerca da sequência dos mesmos - *"sequence-MC4R.gb"* e *"sequence-SEC16B.gb"*. Estes ficheiros podem ser acedidos no repositório associado a este trabalho.

As sequências foram lidas recorrendo à interface *Sequence Input/Output* (SeqIO) do BioPython de acordo com o código abaixo:

```
#Gene MC4R
    geneM= SeqIO.read("sequence-MC4R.gb","genbank")
    geneM
#Gene SEC16B
    geneS= SeqIO.read("sequence-SEC16B.gb","genbank")
    geneS
```

1.1 Gene MC4R

Relativamente ao gene MC4R, através da descrição dada pelo código abaixo, verifica-se que este gene é o gene do recetor de melanocortina 4 do organismo *Homo sapiens* e está localizado no cromossoma 18. Para além disto, através deste, também foi possível obter a sequência completa formada por 1714 nucleótidos.

```
    print ("Descricao:", geneM.description)
    print (geneM.seq)
    print ("Numero de nucleotidos:", len(geneM.seq))
```

De seguida, foi também analisada a informação presente nas anotações do ficheiro (`geneM.annotations`), apresentadas na Figura 1, as quais revelaram informações como o tipo de molécula - molécula de DNA-, a topologia - linear-, a taxonomia e ainda um comentário.

```
{'molecule_type': 'DNA',
 'topology': 'linear',
 'data_file_division': 'PRI',
 'date': '19-FEB-2021',
 'accessions': ['NG_016441', 'REGION:', '4994..6707'],
 'sequence_version': 1,
 'keywords': ['RefSeq', 'RefSeqGene'],
 'source': 'Homo sapiens (human)',
 'organism': 'Homo sapiens',
 'taxonomy': ['Eukaryota',
  'Metazoa',
  'Chordata',
  'Craniata',
  'Vertebrata',
  'Euteleostomi',
  'Mammalia',
  'Eutheria',
  'Euarchontoglires',
  'Primates',
  'Haplorrhini',
  'Catarrhini',
  'Hominidae',
  'Homo'],
 'comment': 'REVIEWED REFSEQ: This record has been curated by NCBI staff in\ncollaboration with LRG
Consortium. The reference sequence was\nderived from AC091576.11.\nThis sequence is a reference
standard in the RefSeqGene project.\nSummary: The protein encoded by this gene is a membrane-
bound\nreceptor and member of the melanocortin receptor family. The\nencoded protein interacts with
adrenocorticotropic and MSH hormones\nand is mediated by G proteins. This is an intronless gene.
Defects\nin this gene are a cause of autosomal dominant obesity. [provided\nby RefSeq, Jan 2010].'}
```


Figura 1: Anotações do ficheiro *"sequence-MC4R.gb"*

O comentário contém informação de interesse acerca da proteína codificada por este gene, como a sua interação com hormonas e outras proteínas. Estas interações serão discutidas mais à frente neste trabalho.

Para além disto, o comentário também revela que a sequência deste gene não é interrompida por intrões e que defeitos no gene MC4R são uma das causas da obesidade, daí a sua relação com a diabetes tipo 2.

Um dos objetivos desta parte era a análise das features. Esta foi feita com a ajuda das funções apresentadas na Figura 2.

A função `typelocal` é uma função inicial que sumariza, de certa forma, o conteúdo das *features*, dando-nos vários tipos existentes e a sua localização. Esta foi utilizada com o intuito de ver quais as *feautures* com maior interesse para analisar. Daqui retirou-se que os tipos de maior interesse seriam, então, a CDS, isto é, sequência codificante de proteína, e misc_features, ou seja, *features* que não podem ser descritas por mais nenhuma *feature key* mas que têm interesse biológico.

Desta forma, desenvolveu-se as funções `qualifiersCDS` e `misc` para que fosse retornado o local e o significado biológico das proteínas codificadas e locais de interesse no gene, respetivamente.

```python
def typelocal(features): #Dá-nos todos os tipos e localizações das features
    print ("Tipo e Local das  Features")
    for f in features:
        print(f.type, f.location)

def qualifiersCDS(features): #Significado Biológico das proteínas codificadas pelo gene
    print ("Qualifiers de CDS")
    for f in range (len(features)):
        if features[f].type=="CDS":
            print("Localização:",features[f].location)
            print (features[f].qualifiers)
            print()

def misc(features): #Retorna a localização e descrição de locais de interesse da sequência do gene
    print("Features de Interesse")
    for f in range (len(features)):
        if features[f].type=="misc_feature":
            print("Localização:",features[f].location)
            print(features[f].qualifiers)
            print()
```

Figura 2: Código relativo a funções utilizadas para a análise das *features* dos genes

Relativamente à CDS do gene MC4R, verificou-se que a sequência entre os nucleotidos 426 e 1425 codifica a proteína com o mesmo nome do gene, isto é, o recetor de melanocortina 4, e que esta é a única proteína a ser codificada por este gene. É esta proteína vai utilizada no BLAST e feita a análise das suas propriedades

Relativamente às misc_features, observa-se que a sequência do genoma apresenta 3 locais de N-glicosilação e 7 domínios transmembranares de interesse, todos localizados dentro do intervalo codificante da proteína MC4R, confirmando a literatura anterior. Estes locais vão sendo analisados com mais detalhe posteriormente no tópico da análise das propriedades da proteína.

1.2 Gene SEC16B

No que toca ao gene SEC16B, o processo de análise foi em tudo semelhante ao anterior, utilizando linhas de código semelhantes para obter a sequência nucleotídica completa, o seu comprimento e a sua descrição.

Desta forma, obteve-se que o gene SEC16B é composto por 55497 nucleotidos no total e pertence ao cromossoma 1 no organismo *Homo sapiens*. Relativamente às anotações do SEC16B, estas demonstraram que tinha uma topologia e taxonomia igual à do gene anterior, uma vez que são sequência do mesmo tipo e do mesmo organismo.

No que diz respeito à análise das *feautures*, através da função `typelocal`, viu-se que este não apresentava misc_features, no entanto, continha várias CDS.

Após correr a função `qualifiersCDS` nas *features* do gene SEC16B, observou-se que este codificava as isoformas 1,2 e 4 da proteína transportadora da proteína SEC16B, isto é, proteínas semelhantes umas às outras que atuam de forma semelhante na célula. Para além disto, focando na localização destas CDS, verifica-se que estas são compostas por várias junções, significando que a sequência codificante é interrompida por vários intrões, ao contrario do gene MC4R.

2 Análise de homologias por BLAST

2.1 Blast através de scripts de BioPython

De forma a otimizar a informação obtida através da *web*, foi realizado, novamente, o BLAST mas a partir de scripts de BioPython. Considerou-se, para esta otimização, que sequências homologas seriam aquelas com um e-value igual a 0.

Previamente, foram descarregados ficheiros *fasta* com informações relativas às proteínas codificadas pelos genes - *"prot1.fasta"* (MC4R) e *"prot2.fasta"* (SEC16B). De seguida, estes ficheiros foram lidos recorrendo ao package SeqIO do BioPython da seguinte forma:

```
#MC4R
    P1seq= SeqIO.read(open("prot1.fasta"), format="fasta")
```

```
#SEC16B
    P2seq= SeqIO.read(open("prot2.fasta"), format="fasta")
```

Após o carregamento do ficheiro de interesse, foram desenvolvidas algumas funções para que o processo do BLAST e de análise possa ser repetido para ambos.

As primeiras funções a serem desenvovidas foram `blast` e `blast_org` (Figura 3). Estas permitem a procura de sequências homólogas à proteína em estudo de todos os organimos (`blast`) ou de apenas um organismo específico (`blast_org`) através do NCBI, guardando a informação recolhida num ficheiro, de preferência XML.

```
1  def blast(Pseq,nome_ficheiro): #pesquisa por sequencias semelhantes no Blast e guarda o ficheiro em XML
2      result_handle = NCBIWWW.qblast("blastp", "swissprot", Pseq.format("fasta"))
3      save_file= open(nome_ficheiro, "w") #exemplo de nome: "apaf-blast-sp.xml"
4      save_file.write(result_handle.read())
5      save_file.close()
6      result_handle.close()
7      result_handle= open(nome_ficheiro)
8      record= NCBIXML.read(result_handle)
9      return record
10
11 def blast_org(Pseq,organismo,nome_ficheiro): #escrever organismo neste formato: "Saccharomyces cerevisiae[organism]"
12     result_handle = NCBIWWW.qblast("blastp", "swissprot", Pseq.format("fasta"), entrez_query = organismo)
13     save_file= open(nome_ficheiro, "w") #exemplo de nome: "apaf-blast-sp2.xml"
14     save_file.write(result_handle.read())
15     save_file.close()
16     result_handle.close()
17     result_handle= open(nome_ficheiro)
18     record= NCBIXML.read(result_handle)
19     return record
```

Figura 3: Função que permite fazer o BLAST de sequências homologas de todos os organismos, `blast` , e de organismos específicos, `blast_org`

Após realizado o BLAST para ambos as proteínas, chamando a função `blast`, atribui-se uma variável ao record retornado pela função, ou seja:

```
#MC4R
    record1 = blast(P1seq,"apaf-blast-spM.xml")
```

```
#SEC16B
    record2 = blast(P2seq,"apaf-blast-spS.xml")
```

Adicionalmente foram desenvolvidas outras funções para verificar os alinhamentos das sequências. A primeira, `align` (Figura 4), é uma função que resume os resultados do BLAST tendo apenas em conta o número de acesso das sequências, o seu e-value e o comprimento do alinhamento. Esta função foi necessária para verificar quantas possíveis sequências homologas existiam, uma vez que o critério era um e-value igual a zero. Assim, verificou-se que, ao chamar a função `align(record1)` e `align(record2)`, retornaram 6 sequências homologas à proteína MC4R e 5 homologas à proteina transportadora de SEC16B, sendo a primeira sequência de e-value igual a 0 a própria proteína.

Com esta informação, decidiu-se obter apenas informações acerca dos primeiros 6 alinhamentos cujo e-value era 0. Para isso, usou-se a função `short_align` (Figura 4). Esta função, permitiu a construção de uma lista com apenas as sequências que se considerou homológas, dando algumas informações sobre as mesmas como o seu id, descrição, nº de acesso, quantidade de HSP (High-scoring Segnment Pairs) - que em ambos os casos, apenas existia um -, o comprimento do HSP e ainda percentagem de resíduos da query que coincidiam com a seqência homóloga (Figuras 5 para MC4R e 6 para SEC16B).

```
21 def align(record): #Retorna todos os e-values e o comprimento do alignments obtidos pelo Blast
22     res = []
23     for alignment in record.alignments:
24         evalue = alignment.hsps[0].expect
25         accession = alignment.accession
26         leng = alignment.hsps[0].align_length
27         res.append(accession + " - " + str(evalue) + " length:" + str(leng))
28     print("E-values and length of alignments:")
29     for s in res:
30         print(s)
31 
32 def short_align(record): #Primeiros alinhamentos com E-value igual a 0
33     aligns=[]
34     for i in range(7): #Só queremos ver os primeiros 6 valores de todos os alignments
35         alignment = record.alignments[i]
36         evalue=alignment.hsps[0].expect
37         if evalue == 0:
38             aligns.append(alignment) #lista com os aligments de e-value igual 0
39     for salign in aligns:
40         print("Hit: " + salign.hit_id + " - " + salign.hit_def )
41         print ("Accession: ", salign.accession)
42         print ("Nº de HSP (high-scoring segment pairs): ", len(salign.hsps))
43         hsp=salign.hsps[0]
44         print ("Comprimento do HSP: ", hsp.align_length)
45         identities= (hsp.identities)*100/(hsp.align_length) #Percentagem de resíduos da query que são iguais aos do hit
46         print ("Identities: ", identities, "%")
47         print ()
```

Figura 4: Funções `align` e `short_align`

```
Hit: sp|P32245.2| - RecName: Full=Melanocortin receptor 4; Short=MC4-R [Homo sapiens]
Accession:  P32245
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  332
Identities:  100.0 %

Hit: sp|Q8HXX3.1| - RecName: Full=Melanocortin receptor 4; Short=MC4-R [Macaca fascicularis]
Accession:  Q8HXX3
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  332
Identities:  98.49397590361446 %

Hit: sp|O97504.1| - RecName: Full=Melanocortin receptor 4; Short=MC4-R [Sus scrofa]
Accession:  O97504
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  332
Identities:  96.3855421686747 %
```

Figura 5: Output incompleto da função `short_align(record1)` para as sequências homologas da proteína MC4R

```
Hit: sp|Q96JE7.2| - RecName: Full=Protein transport protein Sec16B; AltName: Full=Leucine zipper transcription regulator 2; AltName:
Full=Regucalcin gene promoter region-related protein p117; Short=RGPR-p117; AltName: Full=SEC16 homolog B [Homo sapiens]
Accession:  Q96JE7
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  1060
Identities:  100.0 %

Hit: sp|Q75NY9.1| - RecName: Full=Protein transport protein Sec16B; AltName: Full=Regucalcin gene promoter region-related protein p117;
Short=RGPR-p117; AltName: Full=SEC16 homolog B [Bos taurus]
Accession:  Q75NY9
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  1064
Identities:  76.2218045112782 %

Hit: sp|Q91XT4.3| - RecName: Full=Protein transport protein Sec16B; AltName: Full=Leucine zipper transcription regulator 2; AltName:
Full=Regucalcin gene promoter region-related protein p117; Short=RGPR-p117; AltName: Full=SEC16 homolog B [Mus musculus]
Accession:  Q91XT4
Nº de HSP (high-scoring segment pairs):  1
Comprimento do HSP:  1068
Identities:  72.75280898876404 %
```

Figura 6: Output incompleto da função `short_align(record2)` para as sequências homologas da proteína SEC16B

Desta forma, verificou-se que as sequências homologas da proteína em estudo, em ambos os casos, eram sequências de proteínas com o mesmo nome mas pertencentes a organismos diferentes. As Tabelas 1 e 2 resumem os outputs da função `short_align` para a proteína MC4R e SEC16B, respetivamente.

É possível, então, observar que estes resultados estão de acordo com os apresentados anteriormente pelo BLAST através da web.

Tabela 1: Número de acesso, organismo, comprimento do HSP e percentagem de identidades das sequências homologas à proteína MC4R do Homo sapiens, obtidas por scripts em BioPython

Nº de Acesso	Organismo	Comprimento HSP	Identidades (%)
Q8HXX3	*Macaca fascicularis*	332	98,49
O97504	*Sus scrofa*	332	96,39
Q0Z8I9	*Vulpes vulpes*	332	96,08
P56450	*Mus musculus*	332	93,98
P70596	*Rattus norvegicus*	332	93,98
Q9GLJ8	*Bos taurus*	332	93,37

Tabela 2: Número de acesso, organismo, comprimento do HSP e percentagem de identidades das sequências homologas à proteína SEC16B do *Homo sapiens*, obtidas por scripts em BioPython

Nº de Acesso	Organismo	Comprimento HSP	Identidades (%)
Q75NY9	*Bos taurus*	1064	76,22
Q91XT4	*Mus musculus*	1068	72,75
Q75N33	*Rattus norvegicus*	1069	72,50
Q6BCB4	*Oryctolagus cuniculus*	1065	69,20
Q6AW68	*Gallus gallus*	1004	45,32

Adicionalmente, desenvolveu-se a função `alignmentx` (Figura 7) - função complementar - para que se podesse retirar mais informações acerca de uma sequência homóloga com um certo index. Esta fornece mais informações que a função `short_align`, uma vez que explora todos os HSPs e permite verificar o match entre a query e o hit.

```
50  def alignmentx(record,index): #Fornece mais informações acerca de um alinhamento com index específico
51      alignment = record.alignments[index] #o index 0 vai ser a própria proteína
52      print ("Accession: " + alignment.accession) #Identifica o accession (registo da Uniprot da sequência obtida)
53      print ("Hit id: " + alignment.hit_id)
54      print ("Definition: " + alignment.hit_def) #descrição
55      print ("Number of HSPs: ", len(alignment.hsps)) #nº de HSP
56      print ()
57      for hsp in alignment.hsps: #para cada HSP
58          print ("HSP",alignment.hsps.index(hsp))
59          print ("E-value: ", hsp.expect)
60          print ("Length: ", hsp.align_length) #comprimento do alinhamento
61          print ("Identities: ", hsp.identities)
62          print ("Query start: ", hsp.query_start) #início do HSP na query
63          print ("Sbjct start: ", hsp.sbjct_start) #início do HSP na sequência
64          print (hsp.query[0:90])
65          print (hsp.match[0:90]) #verifica quais os AA que estão na mesma posição na seq da query e da seq da proteína original
66          print (hsp.sbjct[0:90])
67          print ()
```

Figura 7: Função `alignmentx`

Nas Figuras 8 e 9 estão apresentados os outputs da função `alignmentx` para o alinhamento com index 1 (exemplo de teste) tanto da proteína MC4R como da SEC16B, respetivamente.

É possível observar que no caso do gene SEC16B, há certas diferenças entre a query e o hit logo nos primeiros aminoácidos.

```
Accession: Q8HXX3
Hit id: sp|Q8HXX3.1|
Definition: RecName: Full=Melanocortin receptor 4; Short=MC4-R [Macaca fascicularis]
Number of HSPs:  1

HSP 0
E-value:  0.0
Length:  332
Identities:  327
Query start:  1
Sbjct start:  1
MVNSTHRGMHTSLHLWNRSSYRLHSNASESLGKGYSDGGCYEQLFVSPEVFVTLGVISLLENILVIVAIAKNKNLHSPMYFFICSLAVAD
MVNSTHRGMH SLHLWNRSS+RLHSNASESLGKGYSDGGCYEQLFVSPEVFVTLGVISLLENILVIVAIAKNKNLHSPMYFFICSLAVAD
MVNSTHRGMHASLHLWNRSSHRLHSNASESLGKGYSDGGCYEQLFVSPEVFVTLGVISLLENILVIVAIAKNKNLHSPMYFFICSLAVAD
```

Figura 8: Output da função `alignmentx(record1,1)` para a proteína MC4R

```
Accession: Q75NY9
Hit id: sp|Q75NY9.1|
Definition: RecName: Full=Protein transport protein Sec16B; AltName: Full=Regucalcin gene promoter region-related protein p117;
Short=RGPR-p117; AltName: Full=SEC16 homolog B [Bos taurus]
Number of HSPs:  1

HSP 0
E-value:  0.0
Length:  1064
Identities:  811
Query start:  1
Sbjct start:  1
MELWAPQRLPQTRGKATAPSKDPDRGFRRDGHHRPVPHSWHNGERFHQWQDNRGSPQPQQEPRADHQQQPHYASRPGDWHQPVSGVDYYE
ME W PQ LPQ  G+  APSKDPDRG  +D +++P+PHSWHNGER HQ QD   SPQPQQ+PR D  Q+PHYA+R G+W  PVSGVDYYE
MEPWIPQWLPQPSGRPPAPSKDPDRGLWKDRYYQPIPHSWHNGERVHQRQDVGRSPQPQQDPRED-LQEPHYAARSGEWRPPVSGVDYYE
```

Figura 9: Output da função `alignmentx(record2,1)` para a proteína SEC16B