Prosite Motif Scanner
Trabalho prático AASB

Alfredo Gomes
Universidade do Minho
MIEI
Grupo02
Número de Aluno: A71655
Email: a71655@alunos.uminho.pt

João Vieira
Universidade do Minho
MIEI
Grupo02
Número de Aluno: A71489
Email: a71489@alunos.uminho.pt

***Resumo*—Foi realizado um projecto com o intuito de realizar uma procura de motifs numa dada sequência, de maneira a relatar, por exemplo, o respetivo score e a respectiva posição na sequência dada.**

I. Introdução

O presente documento consiste numa proposta de resolução do módulo **Prosite Motif Scanner** do trabalho prático da unidade curricular Algoritmos para Análise de Sequências Biológicas inserida no Mestrado Integrado em Engenharia Informática da Universidade do Minho.

Este projeto tem como objetivo a utilização das bases de dados **Prosite Scan** e **NCBI CDD** de maneira a encontrar motifs e domínios e reportar a informação sobre os mesmos e a sua localização na sequência de query.

De seguida, será apresentada uma proposta de resolução para o problema apresentado.

II. Desenvolvimento

A. Análise inicial do módulo/Motivação

*1) **Levantamento de requisitos**:* A tarefa proposta consiste em, dada uma sequência a pesquisar, procurar na base de dados PrositeScan ou CDD (Conserved Domains database) os motifs associados a essa sequência.

Para realização do levantamento de requisitos, começou-se por fazer uma breve análise à definição de motifs, através do estudo da matéria dada nas aulas e com uma breve pesquisa na Internet. Posteriormente, foi pensado como iria ser organizada a informação em termos de estrutura de dados. Também foi feito um balanço dos recursos programáticos que iriam ser precisos para a realização do módulo.

Todo este processo será descrito posteriormente.

*2) **Motifs**:* O primeiro passo na realização deste trabalho consistiu numa breve análise à definição de motifs.

O termo **motif**, conforme apresentado nas aulas de AASB, designa tipicamente um padrão existente numa sequência biológica e que é partilhado por várias sequências dada a função biológica associada.

Assim, os motifs podem ser encontrados em sequências de **DNA**, por exemplo, em locais de ligação de proteínas regulatórios ao DNA para controlo de transcrição, em sequências de **RNA**, na representação de padrões de microRNAs, em sequências de **proteínas**, na representação de domínios conservados de proteínas com funções biológicas determinadas.

*3) **Definição dos formatos de input e output**:* O **input** será simplesmente a sequência a pesquisar (identificador ou fasta).

Por outro lado, o **output** será uma estrutura de dados com informação dos motifs Prosite ou CDD que estejam presentes na sequência de query.

Esta estrutura terá a seguinte informação, relativamente a cada motif:

- A sequência respetiva;
- Acession e identificador da sequência respetiva;
- Acession e identificador do PROSITE Motif encontrado;
- A localização do motif na sequência de input;
- O seu score

4) Recursos programáticos: REST API do **scanProsite** e **BioPython**, fundamentais para realizar a procura de motifs, na base de dados Prosite, existentes numa certa sequência. Também foi utilizado o módulo **re** que fornece operações de match de expressões regulares e os modulos **NCBIWWW** para aceder à base de dados CDD e retirar informação e **NCBIXML** para fazer parse dessa informação (que vem em formato XML).

*5) **Serviços e bases de dados a aceder**:* Para concretização deste trabalho, houve a necessidade de aceder a bases de dados de maneira a realizar uma procura dos motifs associados à sequência de input.

Assim, usaram-se duas bases de dados:

- **Prosite**: Uma base de dados de proteínas que contém caracterização funcional e anotação. O ScanProsite permite fazer um scan de sequências (proteínas) na coleção PROSITE de motifs.
- **NCBI CDD** (Conserved Domain Database): uma base de dados onde estão guardadas as unidades funcionais de várias proteínas. as suas coleções de dominios são curadas pelo NCBI.

6) Versão inicial da API do módulo: Para além da classe definida pela iniciação da estutura (todos os campos a zero ou string vazias) e das funções set para atribuir valores à estrutura, foi também pensado em realizar 3 funções:

- parseMotif: função que recebe os motifs, faz parse dos mesmos e coloca na estrutura os valores do motif encontrado
- scanSeq: função que recebe uma sequência, procura os motifs da mesma, e usa a função anterior para devolver a estrutura preenchida.
- scanCDD: função que faz o mesmo que a função anterior, mas procura os motifs numa base de dados diferente.

B. Implementação

1) Estrutura de dados:

- **score** - valor numérico que representa o grau de similaridade entre a sequência e o motif.
- **pos** - posição inicial e final, relativas à sequência original, onde se situa este motif.
- **seq** - excerto da sequência ao qual o motif pertence (note que este campo só é preenchido se o input for em formato fasta e não o identificador).
- **idSeq** - id da sequência.
- **acSeq** - *Accession number* da sequência. Este valor serve para identificar uma sequência e é sempre constante sendo o campo indicado para citar numa publicação, permitindo que outros possam aceder à mesma informação.
- **idMot** - id do motif.
- **acMot** - *Accession number* do motif. Este valor serve para identificar um motif e é sempre constante sendo o campo indicado para citar numa publicação, permitindo que outros possam aceder à mesma informação.

```
class motif:
def __init__(self):
    self.score = 0
    self.pos = (0,0)
    self.seq = ("")
    self.idSeq=("")
    self.acSeq=("")
    self.idMot=("")
    self.acMot=("")

def setScore(self, score):
    self.score= score

def setPos(self, start, stop):
    self.pos = (start, stop)

def setSeq(self, seq):
    self.seq= seq

def setId(self, idSeq):
    self.idSeq= idSeq

def setAccesion(self, acSeq):
    self.acSeq= acSeq

def setIdMot(self, idMot):
    self.idMot=idMot

def setAcMot(self, acMot):
    self.acMot= acMot
```

2) Funções/Funcionalidades:

- **parseMotif** - função auxiliar que recebe os motifs da função scanSeq, procede a fazer parse do input de modo a retirar tudo aquilo necessário para popular a classe Motifs e devolve um array com elementos desta classe.

  O motif lido do ScanProsite e que será input desta função terá o seguinte formato:

```
mot= "[{'sequence_ac': 'USERSEQ1',
'start': 11, 'stop': 18, 'signature_ac':
'PS01266', 'level_tag': '(0)'},
{'sequence_ac': 'USERSEQ1',
'start': 133, 'stop':
144, 'signature_ac': 'PS00513',
'level_tag': '(0)'}]"

def parseMotif(mot, seq):

#se houver mais do que um motif,
faz um split em "}, {" de maneira
a dividir os motifs um por um

motifs = re.split("}, {", mot)

#faz o parser de cada item do motif.
myreg=re.compile(r"((\[(\{))?\'(\w+)\':
((\')?(\()?(\w+)(\()?(\')?)(\}\])?)")
j=0      res=[]

#percorre todos os motifs

for i in motifs:

    #Separa os "atributos" do motif
    pela virgula
    atributos= re.split(",", motifs[j])
    v=ini=fim=0

    #inicializa uma nova
    instancia de Motif
    exp=motif()

    #percorre todos os atributos

    for m in atributos:
        #para cada atributo, faz o parse
        "myreg" e preenche na estrutura
```

```
    o valor de acordo com cada campo
    for data in re.findall(myreg,
    atributos[v]):
     if(data[3]=='score'):
        motif.setScore(exp,data[7])
     if(data[3]=='start'):
        ini=data[7]
     if(data[3]=='stop'):
        fim= data[7]
     if(data[3]=='sequence_id'):
      motif.setId(exp, data[7])
     if(data[3]=='sequence_ac'):
      motif.setAccesion(exp, data[7])
     if(data[3]=='signature_id'):
      motif.setIdMot(exp, data[7])
     if(data[3]=='signature_ac'):
      motif.setAcMot(exp, data[7])

    #no fim de percorrer todos os
    atributos, se ini e fim forem
    diferentes de zero, preenche o
    "campo" pos da estrutura

    if(ini!=0 and fim!=0): motif.
        setPos(exp,ini,fim)
    i=int(ini)-1
    sequencia=""

    #Se a sequência tiver algum
    caracter que não seja maisculo
    ou se tiver números, ou seja,
    #se a sequência for em formato
    fasta e não o identificador ou
    acession, irá preencher o campo
    #seq com a sequência
    que corresponde ao motif

    if (seq.isalpha()
    and (not(any(char.isdigit()
    for char in seq)))):
     while(i<int(fim)):
      sequencia= sequencia + seq[i]
      i+=1
     motif.setSeq(exp, sequencia)

    v+=1
#imprime todos os campos da estrutura

print("Acession Sequência:", exp.acSeq)
print("Identificador Sequência:"
, exp.idSeq)
print("Acession Prosite Motif:"
, exp.acMot)
print("Identificador Prosite Motif:"
, exp.idMot)
print("Posição:", exp.pos)
    print("Score:" ,exp.score)
    if (seq.isalpha() and (not(
    any(char.isdigit() for
    char in seq)))):
        print("Sequência:", exp.seq)
    res.append(exp)
    print("\n")
    j+=1

return res
```

- **scanSeq** - esta função recebe uma sequência (identificador ou fasta) e usa o método scan do modulo ScanProsite para, em conjunto com o o método read do mesmo modulo, obter os motifs da sequência inserida. Este resultado é enviado à função parseMotif e devolve o resultado desta.

```
def scanSeq(seq):
    keywords={'CC':'/SKIP-FLAG=FALSE;'}
    response = ScanProsite.scan(seq,
    'http://www.expasy.org', 'xml',
    **keywords )

    #Lê todos os motifs encontrados
    obj = ScanProsite.read(response)

    #Tendo já os motifs, irá passá-los
    como argumento,
    assim como a sequência de query
    res= (parseMotif(str(obj), seq))

    return res
```

- **scanCDD** - esta função recebe uma sequência (identificador ou fasta) e usa o método qblast do modulo NCBIWWW e o metodo parse do modulo NCBIXML para procurar informação relacionada aos motifs dessa sequência. Infelizmente não foi possível concluir esta função pois, apesar de estar num estado funcional, apenas devolve dados da proteína introduzida e não dos seus motifs.

```
def scanCDD(seq):
    res= []
    response = NCBIWWW.qblast("blastp",
    "cdd",seq,
    'https://blast.ncbi.
    nlm.nih.gov/Blast.cgi',
    format_type='XML')
    obj= NCBIXML.parse(response)

    for o in obj:
        alignments = sorted(o.alignments,
        key=lambda a: a.hsps[0].expect)
        [0:10]
        for a in alignments:
```

```
                res.append(a)
        return res
```

3) Exemplos de input e output: Após a implementação explicada anteriormente, é apresentado, de seguida, alguns exemplos de input e output do script desenvolvido.

Assim sendo, **scanSeq()**, como função principal deste módulo, pode ser chamada com 3 inputs:

Input: **Sequência**

```
seq="MGKNVVVLGTQWGDEGKGKIVDLLTQDAQVVVRY
QGGHNAGHTLKINGVKTVLRLIPSGMLRPNVTCYIANGV
VLSPQALLSEIKELEGNGINVRERLRISLACPLILPYH
IALDKARETHMGKSAIGTTGRGIGPAYEDKVARRALRVG
DLFHRDRFANKLTELLDYHNFVLTQYFKQPAVDLESLLG
ESLQWAEELRPMVCDVSACLHEHRKQGENILFEGAQGVY
LDIDHGTYPYVTSSNTCVGSVINGAGFGPRYIDYVLGIT
KAYTTRVGGGPFPTELLDDVGKRIAERGQEFGAVTGRPR
RCGWFDAVLLKRSIELNSISGLCVTKLDVLDGLEVLRIA
VAYKDRDGNILSRPPLAADDFNDLLPVYEELPGWQESTA
DVTVMSDLPANARAYLKRIEEILGIPIDMLSTGPERDST
ITLRGPFL"
```

Output:

```
Acession Sequência: USERSEQ1
Identificador Sequência:
Acession Prosite Motif: PS01266
Identificador Prosite Motif:
Posição: ('11', '18')
Score: 0
Sequência: QWGDEGKG


Acession Sequência: USERSEQ1
Identificador Sequência:
Acession Prosite Motif: PS00513
Identificador Prosite Motif:
Posição: ('133', '144')
Score: 0
Sequência: GIGPAYEDKVAR
```

Input: **Identificador**

```
seq= "ENTK_HUMAN"
```

Output:

```
Acession Sequência: P98073
Identificador Sequência: ENTK_HUMAN
Acession Prosite Motif: PS50024
Identificador Prosite Motif: SEA
Posição: ('54', '169')
Score: 32


Acession Sequência: P98073
Identificador Sequência: ENTK_HUMAN
Acession Prosite Motif: PS50068
Identificador Prosite Motif: LDLRA_2
Posição: ('183', '222')
Score: 10


Acession Sequência: P98073
Identificador Sequência: ENTK_HUMAN
Acession Prosite Motif: PS01209
Identificador Prosite Motif: LDLRA_1
Posição: ('197', '221')
Score: 0

.................


Acession Sequência: P98073
Identificador Sequência: ENTK_HUMAN
Acession Prosite Motif: PS00135
Identificador Prosite Motif: TRYPSIN_SER
Posição: ('965', '976')
Score: 0
```

Input: **Acession**

```
seq="Q04962"
```

Output:

```
Acession Sequência: Q04962
Identificador Sequência: FA12_CAVPO
Acession Prosite Motif: PS51092
Identificador Prosite Motif: FN2_2
Posição: ('41', '89')
Score: 19


Acession Sequência: Q04962
Identificador Sequência: FA12_CAVPO
Acession Prosite Motif: PS00023
Identificador Prosite Motif: FN2_1
Posição: ('46', '87')
Score: 0

..............................

Acession Sequência: Q04962
Identificador Sequência: FA12_CAVPO
Acession Prosite Motif: PS00134
Identificador Prosite Motif: TRYPSIN_HIS
Posição: ('394', '399')
Score: 0


Acession Sequência: Q04962
Identificador Sequência: FA12_CAVPO
Acession Prosite Motif: PS00135
```

```
Identificador Prosite Motif: TRYPSIN_SER
Posição: ('545', '556')
Score: 0
```

Para além do output que é impresso na consola, a função retorna, em todo o caso, a estrutura preenchida.

C. Ferramentas utilizadas

Para além das ferramentas de bases de dados explicadas anteriormente, para a resolução deste trabalho foram também utilizadas:

- **Slack**: usado para facilitar a comunicação e a colaboração entre os elementos do grupo e destes com os *project managers* e com outros elementos de outros grupos. Esta ferramenta foi o principal meio de comunicação usado, e serviu para a visualização do enunciado e explicação mais aprofundada dos diferentes módulos.
- **Github**: plataforma que permite a hospedagem de código-fonte com controlo de versões. É extremamente útil pois permite uma fácil partilha de código entre os elementos do grupo, e, com a facilidade do controlo de versões, permite uma melhor organização no desenvolvimento do projecto.
- **Trello**: aplicação que permite uma melhor gestão de todas as fases de desenvolvimento do projecto. Assim sendo, usando uma filosofia de "post-it"pode se dividir o trabalho, por exemplo, em tarefas em atraso (*Backlog*), tarefas prontas para começar a respetiva realização (*Groomed*), tarefas em desenvolvimento (*Doing*) e tarefas concluidas (*Done*).

Estas ferramentas facilitaram a implementação de uma metodologia agile. Esta metodologia tem por base a realização de vários *sprints* de maneira a ter uma nova versão do produto num curto espaço de tempo. É encorajada a colaboração, quer dentro da mesma equipa quer entre equipas.

III. Conclusion

Neste projeto foi proposta a criação de um modulo com funções para, dada uma sequência (identificador ou fasta), devolver uma estrutura de dados com informação dos motifs que estejam presentes na sequência.

Primeiramente foi feito o levantamento de requisitos. Nesta fase foram tomadas decisões quanto às funções necessárias e as variáveis necessárias para guardar toda a informação relevante.

Após concluído o trabalho prático, é possível dizer que, com a exceção da parte adicional de acesso ao CDD, o mesmo foi realizado com sucesso.

De salientar que todas as ferramentas propostas foram utilizadas com sucesso, nomeadamente, o slack, trello e github.

Numa vertente mais pedagógica, o presente trabalho prático permitiu o desenvolvimento de capacidades na interação com bases de dados ProSite e NCBI.