BLAST

- Biopython

BLAST - revisão

- BLAST (sigla em inglês que significa: Basic Local Alignment Search Tool), é um algoritmo para comparar informações de sequências biológicas primárias, tais como sequências de aminoácidos de diferentes proteínas ou nucleotídeos de sequências de DNA.

- Uma pesquisa BLAST permite comparar uma sequência fornecida com uma base de dados de sequências e identificar as sequências que se assemelham à sequência consultada e que estejam acima de um certo grau de semelhança, ou com outras sequências.

Biopython

BLAST com Biopython

- O BioPython tem interfaces com o BLAST que permitem:
  - Preparar queries ao BLAST, definindo os parâmetros relevantes
  - Executar queries
  - Tratar os resultados de queries, executas usando BioPython ou diretamente pelo utilizador
- Estas funcionalidades são extremamente úteis para automatizar estes procedimentos e tratar quantidades grandes de dados

BLAST com Biopython

- Módulo Bio.Blast.NCBIWWW: função **qblast()**, com três parâmetros não opcionais:
  - Programa a usar (string: “blastn”, “blastp”, “blastx”, “tblast”, “tblastx”)
  - Base de dados a procurar (string: “nt”, “nr”, ...) – lista completa no site do Blast
  - Sequência query: pode ser a própria sequência, a sequência em formato Fasta, ou um identificador (e.g. GI)

blastn = nucelotide blast
blastp = protein blast
blastx = procura contra sequências existentes na base de dados do NCBI
tblast = translated blast
tblastx = translated blast x

- Existem ainda vários parâmetros opcionais que permitem definir: o tipo de output (XML por omissão), o e-value, a matriz de substituição a usar, gap penalties, etc. - help(NCBIWWW.qblast) para saber mais !

BLAST remoto: exemplo

Correr BLASTN com sequência em formato FASTA (base de dados nt)

```
>>> from Bio.Blast import NCBIWWW
>>> from Bio import SeqIO
>>> record = SeqIO.read(open("m_cold.fasta"), format="fasta")
>>> result_handle = NCBIWWW.qblast("blastn", "nt", record.format("fasta"))
```

Guardar resultado

```
>>> save_file = open("my_blast.xml", "w")
>>> save_file.write(result_handle.read())
>>> save_file.close()
>>> result_handle.close()
```

Parsing dos resultados do BLAST

- Os parsers do BioPython funcionam garantidamente apenas com o formato de output **XML**; existe um parser para formato texto mas não funciona em todos os casos dadas as alterações aos formatos de saída do BLAST
- Os parsers do BLAST (xml) funcionam com
  - BLAST corrido na web
  - programas locais
  - BioPython com qblast

Parsing dos resultados do BLAST

- O primeiro passo será obter o handle com os resultados, dependendo da forma como se corre:

Usar qblast (neste caso para query por GI com blastn)

```
>>> from Bio.Blast import NCBIWWW
>>> result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")
```

Ler resultados de ficheiro

```
>>> result_handle = open("my_blast.xml")
```

Parsing dos resultados do BLAST

- Podemos então processar os resultados, usando as funções **read**() – para uma query- ou **parse**() – para multiplas queries.

```
>>> from Bio.Blast import NCBIXML
>>> blast_records = NCBIXML.read(result_handle)
```

```
>>> from Bio.Blast import NCBIXML
>>> blast_records = NCBIXML.parse(result_handle)
>>> for blast_record in blast_records:
... # faz alguma coisa com o record …
```

Resultados do BLAST: classe *BlastRecord*

- Um objeto **BlastRecord** contém toda a informação que se pode querer retirar do resultado de um Blast, incluindo: todos os parâmetros usados e os resultados completos
- Os resultados de cada alinhamento estão organizados pelos seus HSPs (High-scoring Segment Pairs)
- Os exemplos dão apenas alguns casos de uso possível, devendo consultar-se a documentação para mais detalhes

http://biopython.org/DIST/docs/api/Bio.Blast-module.html

Resultados do BLAST: classe *BlastRecord*

```
>>> E_VALUE_THRESH = 0.05
>>>for blast_record in blast_records:
          for alignment in blast_record.alignments:
...          for hsp in alignment.hsps:
...               if hsp.expect < E_VALUE_THRESH:
...                    print ('****Alignment****')
…                      print ('sequence:', alignment.title)
...                    print ('length:', alignment.length)
...                    print ('e value:', hsp.expect)
...                    print (hsp.query[0:75] + '...' )
...                    print( hsp.match[0:75] + '...' )
...                    print( hsp.sbjct[0:75] + '...' )
```

Conteúdo do *BlastRecord*

- Nível do “record”
  - *matrix* (matriz de substituição); *gap_penalties*; *database; alignments*
- Nível “alignment”
  - *accession*; *hit_id*; *hit_def*; *alignment.length*; *hsps*
  - *http://biopython.org/DIST/docs/api/Bio.Blast.Record.Alignment-class.html*
- Nível “hsp” (High Scoring Pairs)
  - *expect* (e-value), *score*, *align_length*; *query*; *match*; *sbjct*
  - http://biopython.org/DIST/docs/api/Bio.Blast.Record.HSP-class.html