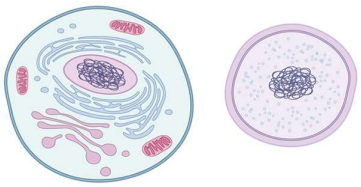
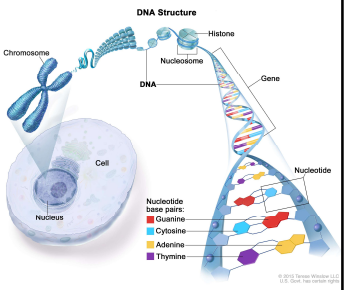
Sequências

- Dogma central da Biologia Molecular
- Sequências: formato Fasta
- BioPython
- Descoberta de genes

Célula

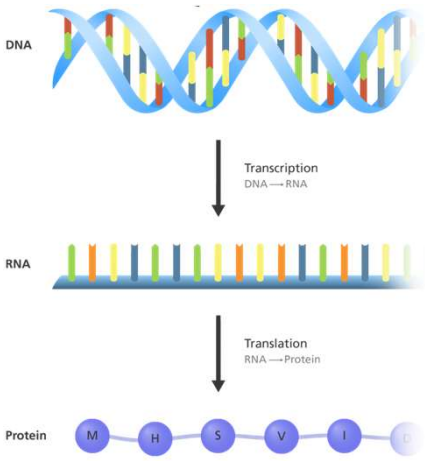
- **Célula**: unidade básica de vida. Compostas essencialmente por: proteinas, ácidos nucleicos, carbohidratos e lipidos.
- **Procariontes**:
  - Apresentam seu material genético delimitado por uma membrana
- **Eucariontes**:
  - Material genético fica dentro do núcleo, ficando assim separado do citoplasma.
- **DNA**: cadeias compostas por sequências de ácidos nucleios (Adenina (A), Guanina (G), Citosina (C), Timina (T)) que contém a informação genética responsável pelo desenvolvimento e funcionamento da célula.





Dogma central da Biologia Molecular

- A síntese de proteínas acontece em dois passos:
  - **Transcição**: sintese de RNA
    - é o processo de formação do RNA a partir da cadeia de DNA.
  - **Tradução**: sintese de proteína
    - processo onde a informação presente no RNAm (RNA mensageiro) é utilizada pelos ribossomas para sintetizar uma cadeia de aminoácidos.
    - a cada conjunto de 3 acidos nucleicos corresponde 1 aminoácido.



Genoma e sequenciação

- **Genoma**: é toda a informação presente no DNA da célula.

- **Sequenciação do DNA**:
  - Processo que têm como finalidade determinar a ordem das bases nitrogenadas adenina (A), guanina (G), citosina (C) e timina (T) da molécula de DNA.

Bases de Dados de Sequências (1º aula)

- Contêm informação sobre a sequências de nucleotidos (Genes) ou de aminoácidos (Proteínas).
- Classificação:
  - Primárias: dados de sequenciação da responsabilidade dos seus autores; dados não são tratados nem curados.
  - Secundárias: BDs com dados curados por especialistas; implicam trabalho de validação dos dados.

Formato de Sequências

- Por razões históricas, as BDs de sequências permitem a visualização (ou exportação) dos seus registos em *flat files* (texto) com uma dada estrutura
- Vários formatos distintos são usados pelas BDs e pelas ferramentas existentes
- Formatos usados pelo NCBI, EBI e DDBJ são muito semelhantes

Formato FASTA

- Linha de definições/ comentários iniciada com >
- Esta linha pode incluir vários identificadores de BDs e identificação da sequência
- Seguido de várias linhas com a sequência

.fna

```
>gi|1322283|gb|U54469.1|DMU54469 Drosophila melanogaster (…)
CGGTTGCTTGGGTTTTATAACATCAGTCAGTGACAGGCATTTCCAGAGTT (…)
GCTGCCTTTGGCCACCAAAATCCCAAACTTAATTAAAGAATTAAATAATT (…)
TAACCTACGCAGCTTGAGTGCGTAACCGATATCTAGTATACATTTCGATA (…)
```

.faa

```
>gi|1322285|gb|AAC03525.1| eukaryotic initiation factor 4E-I
[Drosophila melanogaster]
MQSDFHRMKNFANPKSMFKTSAPSTEQGRPEPPTSAAAPAEAKDVKPKEDPQET
GEPAGNTATTTAPAGDDAVRTEHLYKHPLMNVWTLWYLENDRSKSWEDMQNEI
TSFDTVEDFWSLYNHIKPPSEIKLGSDYSLFKKNIRPMWEDAAN (…)
```


Biopython

- Conjunto de ferramentas gratuitas escritas na linguagem de programação Python que facilitam tarefas relacionadas com biologia computacional.
- Instalação :
  - pip install biopython
  - docker pull biopython/biopython
- Upgrade ou desinstalação:
  - pip install biopython –upgrade
  - pip uninstall biopython
  - docker rmi biopython/biopython

biopython

https://biopython.org/

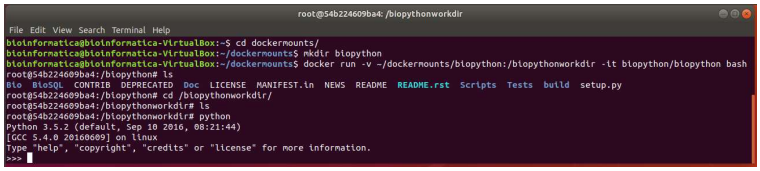
Biopython - Principais funcionalidades

- Parsing de ficheiros em formatos importantes para a Bioinformática:
  - Formatos: FASTA, GenBank, PubMed, SwissProt, Unigene, SCOP, etc.
  - Resultado de ferramentas: Blast, ClustalW, etc.
  - Possibilidade em muitos casos de iterar sobre registos, nos casos em que os ficheiros representam múltiplas entidades
- Interface com ferramentas e bases de dados bioinformáticas
  - Através da Web / web services: Blast, Entrez, PubMed, SwissProt, Prosite, etc
  - Através de instalações locais: Blast, ClustalW, etc.

Biopython - Principais funcionalidades

- Implementação de classes para manipulação de sequências e sua anotação (features), bem como operações de processamento sobre sequências (tradução, transcrição, motifs, etc.)
- Implementação de algoritmos de alinhamento de sequências e tratamento de matrizes de scoring
- Ferramentas básicas de mineração de dados
- Interface com o BioSQL, um esquema de manipulação de bases de dados de sequências

Biopython - Seq class



- Um dos principais objetos no BioPython é o objeto **Seq**. Este permite guardar sequências e trabalhar sobre elas.
- Objeto contem:
  - String com a sequência

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import generic_dna, generic_protein
>>> my_seq = Seq("AGTACACTGGT")
>>> my_seq
        Seq('AGTACACTGGT')
>>> my_dna = Seq("AGTACACTGGT")
>>> my_dna
        Seq('AGTACACTGGT')
```


Seq class - métodos

- **.find(*pattern*)**: procura o padrão na sequência, retornando o index onde a pattern se inicia
- **.count(*pattern*)**: conta quantas vezes o padrão existe na sequência
- **.complement()**: retorna a sequência complementar
- **.reverse_complement()**: retorna o complement reverse da sequência
- **.transcribe()**: converte a seq DNA em RNA (normalmente assume-se que a sequência DNA é a cadeia codificante), substituição de T por U.
- **.translate()**: converte sequências de DNA ou RNA em sequência de aminoácidos

Seq class – manipulação

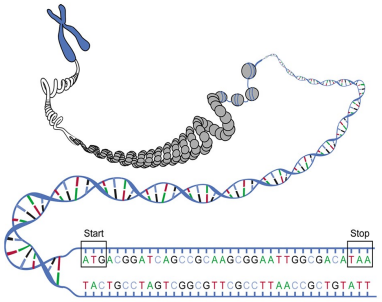
```
>>> from Bio.Seq import Seq
>>> my_seq = Seq("GATCG")
>>> len(my_seq)
5
>>> my_seq[2]
'T'
>>> my_seq.count("G")
2
>>> my_seq[-2]
'C'
>>> my_seq.lower()
Seq('gatcg')
>>> "GATC" in my_seq
True
>>> my_seq.find("ATC")
1
```

Slicing de sequências

```
>>> my_seq = Seq("GATCGATGGGCCTATATAGGATCG")

>>> my_seq[4:12]
Seq('GATGGGCC')

>>> my_seq[0::3]
Seq('GCTGTAGT')

>>> my_seq[1::3]
Seq('AGGCATGC')
```


ORFS

- **Open Reading Frame**: sequência com início no codão de início (ATG) da tradução até ao codão de terminação (TAG, TAA, TGA) que potencialmente codifica uma proteína.
- Cada cadeia de DNA tem 3 configurações possíveis para sintetizar proteína:



TGATGATATTTAGCGAGTAAC

Encontrar ORFs a partir de sequência DNA

- Sequência:

  CGCTACGTCTTACGCTGGAGCTCTCATGGATCGGTTCGGTAGGGCTCGATCACATCGCTAGCCAT

- Configurações de leitura:

Frame 1: CGC TAC GTC TTA CGC TGG AGC TCT CAT GGA TCG GTT CGG TAG GGC TCG ATC ACA TCG CTA GCC AT

Frame 2: C GCT ACG TCT TAC GCT GGA GCT CTC ATG GAT CGG TTC GGT AGG GCT CGA TCA CAT CGC TAG CCA T

Frame 3: CG CTA CGT CTT ACG CTG GAG CTC TCA TGG ATC GGT TCG GTA GGG CTC GAT CAC ATC GCT AGC CAT

Frame -1: ATG GCT AGC GAT GTG ATC GAG CCC TAC CGA ACC GAT CCA TGA GAG CTC CAG CGT AAG ACG TAG CG

Frame -2: A TGG CTA GCG ATG TGA TCG AGC CCT ACC GAA CCG ATC CAT GAG AGC TCC AGC GTA AGA CGT AGC G

Frame -3: AT GGC TAG CGA TGT GAT CGA GCC CTA CCG AAC CGA TCC ATG AGA GCT CCA GCG TAA GAC GTA GCG

Encontrar ORFs a partir de sequência DNA

- Identificar codões de iniciação e stop

FRAME +1: CGC TAC GTC TTA CGC TGG AGC TCT CAT GGA TCG GTT CGG **TAG** GGC TCG ATC ACA TCG CTA GCC AT

FRAME +2: C GCT ACG TCT TAC GCT GGA GCT CTC **ATG** GAT CGG TTC GGT AGG GCT CGA TCA CAT CGC **TAG** CCA T

FRAME +3: CG CTA CGT CTT ACG CTG GAG CTC TCA TGG ATC GGT TCG GTA GGG CTC GAT CAC ATC GCT AGC CAT

FRAME -1: **ATG** GCT AGC GAT GTG ATC GAG CCC TAC CGA ACC GAT CCA **TGA** GAG CTC CAG CGT AAG ACG **TAG** CG

FRAME -2: A TGG CTA GCG **ATG** **TGA** TCG AGC CCT ACC GAA CCG ATC CAT GAG AGC TCC AGC GTA AGA CGT AGC G

FRAME -3: AT GGC **TAG** CGA TGT GAT CGA GCC CTA CCG AAC CGA TCC **ATG** AGA GCT CCA GCG **TAA** GAC GTA GCG

- ORFs:

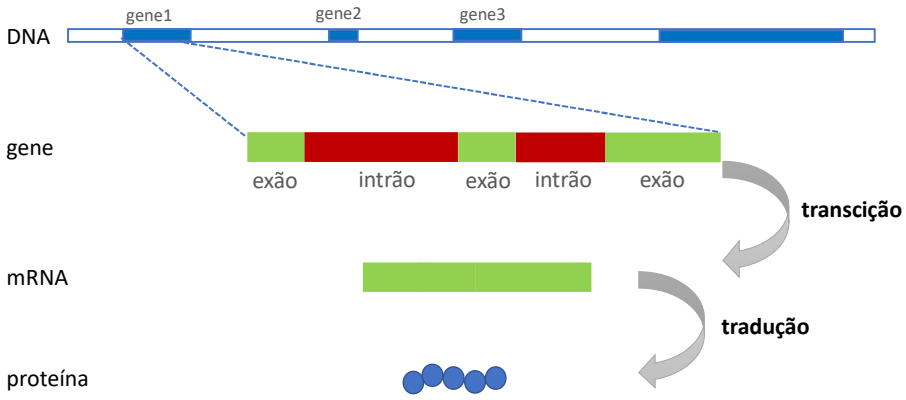
FRAME +2: ATG GAT CGG TTC GGT AGG GCT CGA TCA CAT CGC TAG

FRAME -1: ATG GCT AGC GAT GTG ATC GAG CCC TAC CGA ACC GAT CCA TGA

FRAME -3: ATG AGA GCT CCA GCG TAA

Descoberta de Genes

- Objetivo: encontrar genes a partir de uma sequência de DNA não caracterizada, incluindo as coordenadas dos intrões /exões.

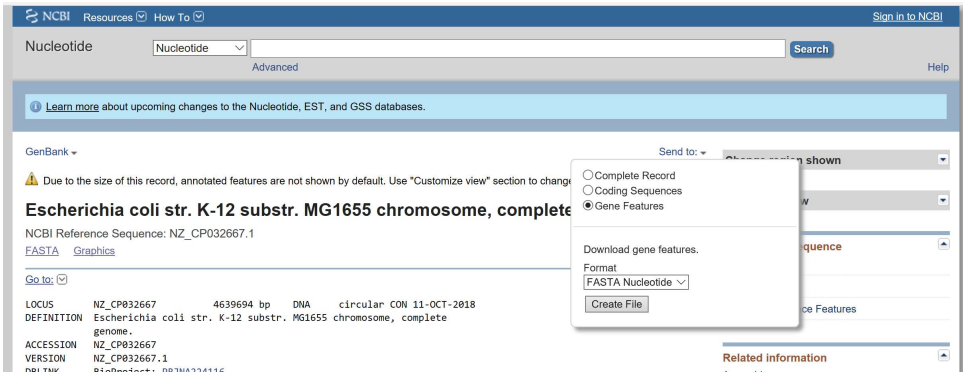


Glimmer3

- Ferramenta que permite a procura de genes em DNA microbiano, especialmente em genomas de bactérias, archea e virus.
- Usa *interpolated Markov models* (IMMs) para identificar zonas codificantes e distingui-las de DNA não codificante.
- http://ccb.jhu.edu/software/glimmer/index.shtml
- https://ccb.jhu.edu/software/glimmer/glim302notes.pdf

Glimmer – exemplo

- 1 – Obter todas as sequências de genes presentes no organism de interesse.



Glimmer – exemplo

- 2 – Abrir glimmer docker com o seguinte comando:

```
docker run \
-v /usr/lib/x86_64-linux-gnu/libstdc++.so.6:/usr/lib/libstdc++.so.6:ro \
-v ~/dockermounts/glimmer:/glimmerwork \
-it quay.io/biocontainers/glimmer:3.02--3 bash
```

O container necessita de uma biblioteca do Linux

- 3 – Criação dos modelos interpolated context model (ICM) usando o comando built-icm
  - **`build-icm icm_file.icm < sequences_file.txt`**

Glimmer – exemplo

- 4 - Descoberta de genes presentes numa sequência de DNA.
  - `glimmer3 my_seq.fna icm_file.icm my_seq.ref`

- Manual da ferramenta:
  - Verificar o significado dos outputs dos ficheiros **.detail** e **.predict**

    https://ccb.jhu.edu/software/glimmer/glim302notes.pdf