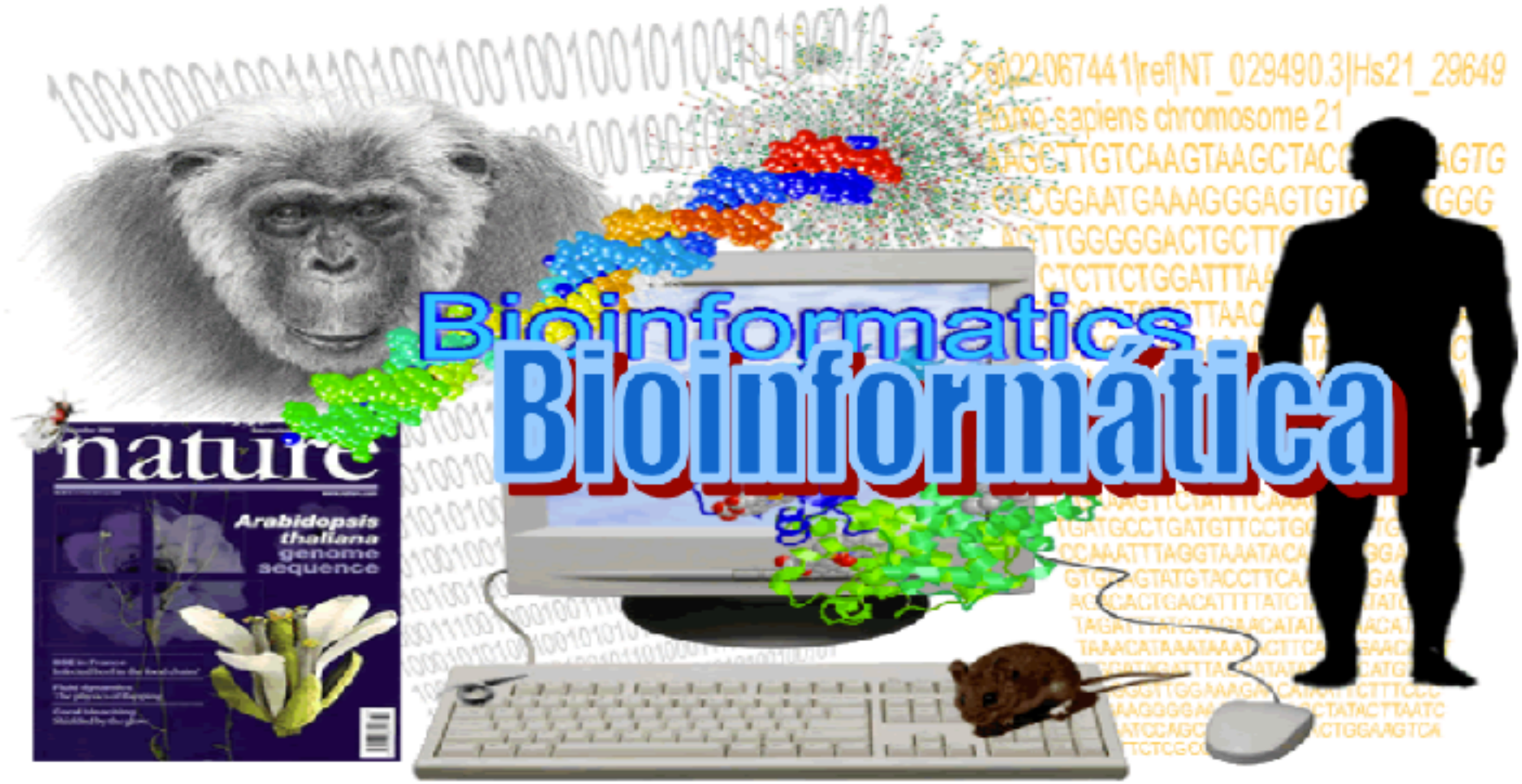
Bioinformatics
Bioinformática
nature
Arabidopsis thaliana genome sequence

Bioinformática aplicada à Nanociências

Introdução

Sistemas de Informação
CPO187 – Bioinformática aplicada à Nanociências

Prof. Sylvio A. G. Vieira – sylvio@unifra.br

Comparar sequências!

- Por que comparar sequências?
- Achar similaridades????
- Dadas 2 sequências, quão parecidas elas são?
- DNA e proteína
- Buscas em banco de dados
  - Achar quais sequências do banco são parecidas com minha sequência-consulta
  - Consulta (query) é tipicamente uma sequência nova
  - Busca no Google?

Comparar sequências!

- Porquê?????
  - Construir famílias
  - Saber quais organismos tem membros da família
  - Determinar uma "assinatura" para a família
  - Construir filogenias
  - Entender a história evolutiva de genes e organismos
    - (Bactérias, fungos, …)

Comparar sequências!

- Em geral buscamos sequências que sejam "aparentadas"

- Sequências aparentadas são similares
  - aparentadas = homólogas
    - Descendem de um mesmo ancestral

      Descendentes sofreram mutações ao longo do tempo

Comparar sequências!

Genes Homólogos

O gene homólogo a outro quer dizer que eles possuem um ancestral em comum, refletido na sua sequência de DNA comum. Se comparar as sequências de DNA, você vai ver que existem regiões entre o humano e o rato que possuem regiões idênticas, que provavelmente se mantiveram conservadas do ancestral comum (ou convergiram para a mesma sequência ao longo do tempo).

Genes Ortólogos

Um gene ortólogo de um outro significa que eles são de espécies diferentes, (1) que possuem um ancestral comum e (2) a proteína codificada por este código genético manteve suas propriedades e funções.

Exemplo: Ao pegar um gene X homólogo humano e colocá-lo em uma levedura com o gene Y homólogo inativado.

Se por acaso, o gene humano restabelecer o comportamento normal da levedura, você saberá que aquele gene manteve sua função e pode ser considerado um ortólogo àquele gene homólogo de levedura.

Alinhamento de DNA

GTGGTGGCCTACGAAGGT

GTAGTGCCTTCGAAGGGT

Como avaliar o Alinhamento

Sistema de pontuação

Match: +1

Mismatch: –1

Pontuação do Alinhamento

G T G G T G G C C T A C G A A G G T

G T A G T G C C T T C G A A G G G T

+1+1-1+1+1+1-1+1-1+1-1-1-1+1-1+1+1+1 = 4

Como melhorar o Alinhamento

Com a introdução de espaços!!!

Como melhorar o Alinhamento

G T G G T G G C C T A C G A A G G T
G T A G T G C C T T C G A A G G G T
+1+1-1+1+1+1-1+1-1+1-1-1-1+1-1+1+1+1 = 4

G T G G T G G C C T A C G A A - G G T
G T A G T G - C C T T C G A A G G G T

Como avaliar o Alinhamento

Sistema de pontuação com espaços

Match: +1

Mismatch: –1

Espaço: -2

Pontuação do Alinhamento

```
GTGGTGGCCTACGAA-GGT
GTAGTG-CCTTCGAAGGT
```

+1+1-1+1+1+1-2+1+1+1-1+1+1+1+1-2+1+1+1 = 9

Porque um sistema de pontuação?

Matches tem que ser recompensados (> 0)

Mismatches e espaços tem que ser penalizados (< 0)

Mismatches representam substituições

- Mutações (ocorrem com frequência)
- Podem não trazer letalidade

Espaços representam inserções ou remoções

- Mais prováveis de causarem letalidade
- Alteram quadro de leitura
- Ocorrem com muito menos frequência

Alinhamentos ótimos

São os alinhamentos de pontuação máxima similaridade entre duas sequências

É o valor da pontuação do alinhamento ótimo

No exemplo anterior

Similaridade = 9

9 é melhor que 4!!!

Pontuações para trocas de bases

Os nucleotídeos tem diferentes características:

Adenina e Guanina são Purinas

Citosina e Timina são Pirimidinas

Se base correta = + 1
Se base trocada mesma família = 0
Se base trocada família diferente = -1

Pontuações para trocas de bases

Algoritmo para achar alinhamentos ótimos de DNA

Desenvolvido com a técnica de Programação dinâmica

Técnica desenvolvida na década de 1950 por Richard Bellman

Programação dinâmica

Se usa para problemas que tem uma estrutura de subproblemas

Num alinhamento com sequências H e C um subproblema é qualquer alinhamento entre h′ e c′ tal que

h′ = um prefixo de H e c′ = um prefixo de C

```
H:    IIWGEDTLMEYLENPKKYIPGTKMIFVGIKKKEERADLIAYLKKATNE
C:    VVWTKETLFEYLLNPKKYIPGTKMVFAGLKKADERADLIKYIEVESAKSL
        *   ** *** ********** * * **  ****** *
```

```
H:    IIWGEDTLMEYLENPKKYIPGT
C:    VVWTKETLFEYLLNPKKYIPGT
        *   ** *** *********
```

Um prefixo

Programação dinâmica

Como fazer?

Achar soluções de subproblemas e armazená-las em uma tabela (matriz)

Para achar a solução ótima:

Procurar as soluções na direção dos subproblemas menores para os maiores

Programação dinâmica

Este processo precisa começar com o "menor subproblema possível".

Qual seria?

Quando pelo menos uma das sequências é vazia

Inicialização: Alinhar h com cadeia vazia e alinhar c com cadeia vazia

	h	G	A	T	C
c					
G					
T					
C					

Programação dinâmica

Alinhar caracter H com caracter C

- 3 possibilidades – X com Y
- Aplicar pontuação respectiva, dependendo das bases encontradas

	h	G	A	T	C
c	0				
G		1	0	-1	-1
T		-1	-1	1	0
C		-1	-1	0	1

Se base correta = + 1
Se base trocada mesma família = 0
Se base trocada família diferente = -1

Pontuação por PD

G T G G T G G C C T A C G A A G G T

G T A G T G C C T T C G A A G G G T

+1+1+0+1+1+1-1+1+0+1-1-1+0+1+0+1+1+1 = 8

Atividade

- Buscar duas sequências

- Implementar um algoritmo que faça a comparação e a pontuação
- Testar e entregar pelo moodle.