Universidade de São Paulo
Instituto de Química

Comparação de sequências

aula 1

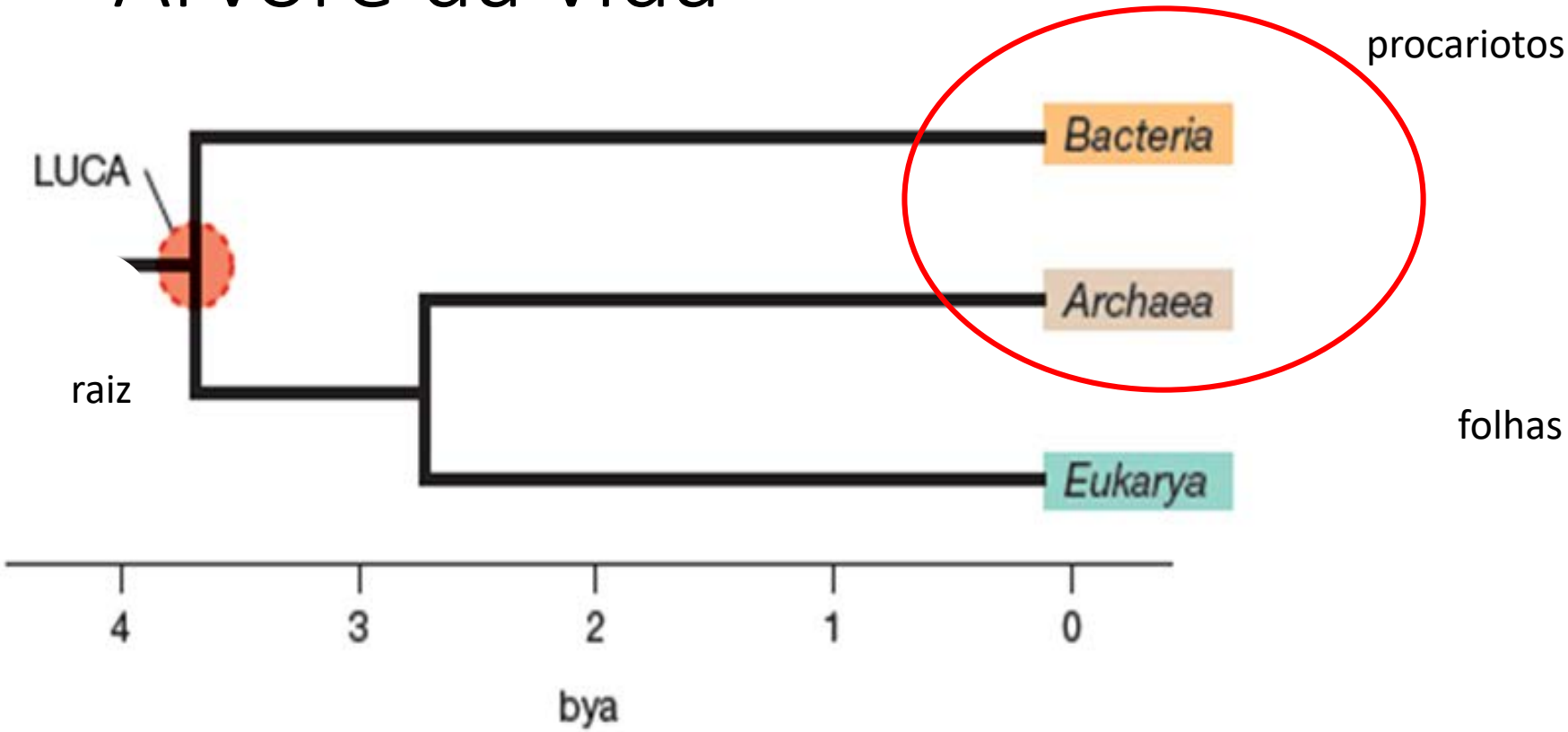
João Carlos Setubal

2022

Motivação

- Um tópico importante em biologia é evolução
- Evidências diversas sugerem que todas as formas de vida na terra são descendentes de um mesmo ancestral
- Ou seja, todas as formas de vida são aparentadas entre si
- Uma pergunta natural é: como se agrupam essas diversas formas de vida?
- é possível construir uma “árvore da vida”?
- Resposta: sim

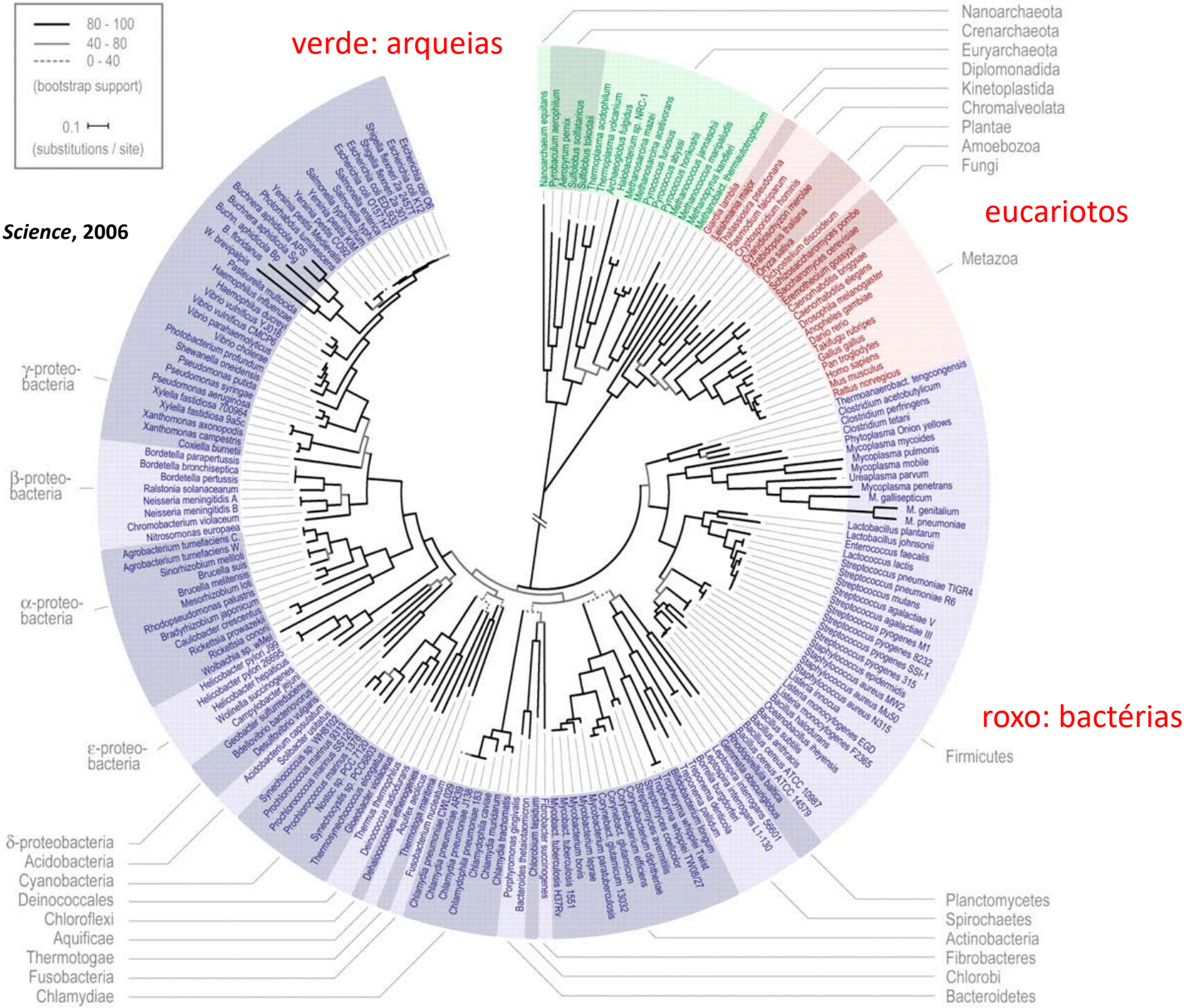
Árvore da vida



+ virus (inclui os bacteriófagos)

Gostaríamos de obter uma árvore com mais “resolução”

- uma que incluísse espécies nas folhas, ao invés dessas categorias amplas como “bactérias” e “eucariotos”
- isto tem sido feito por diversos grupos; um resultado de 2006 está no próximo slide



Ciccarelli et al, *Science*, 2006

Como foi possível criar tal árvore?

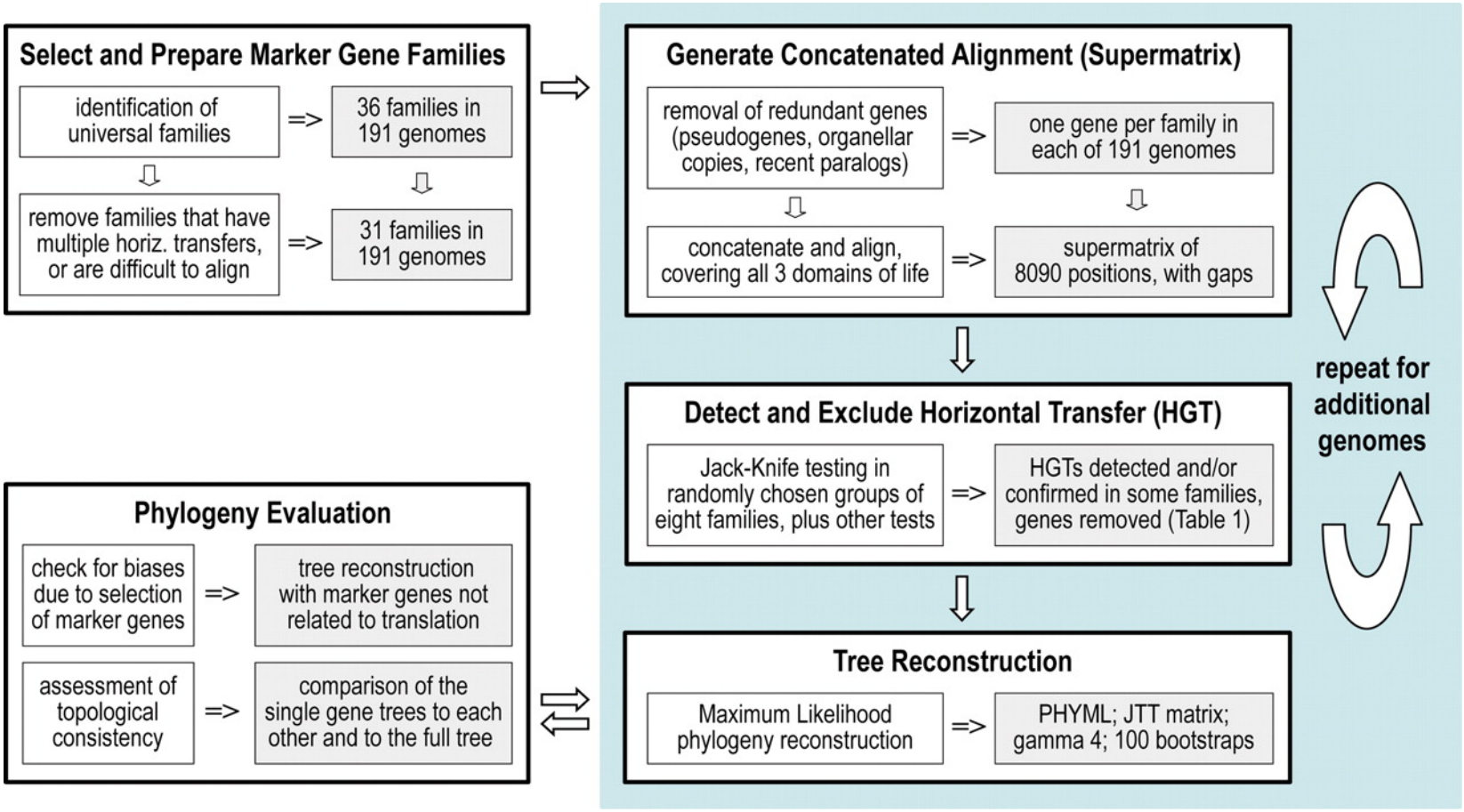
- Determinar genes que são compartilhados por todos os seres vivos
- O que é “compartilhar”?
  - Gene x_1 em organismo *A* tem função α
  - Gene x_2 em organismo *B* é **homólogo** e tem a **mesma função** α
  - Então x_1 e x_2 são “o mesmo gene *x*” e portanto ele é compartilhado por *A* e *B*
- Quais genes são esses?

Orthologous Group	Av. Length	Annotation	Genes in Prok.	Genes in Euk.	Total Genes
COG0012	380	Predicted GTPase, probable translation factor	171	30	201
COG0016	423	Phenylalanine-tRNA synthethase alpha subunit	168	42	210
COG0018†	548	Arginyl-tRNA synthetase	175	45	220
COG0048	137	Ribosomal protein S12	168	48	216
COG0049	182	Ribosomal protein S7	169	41	210
COG0052	240	Ribosomal protein S2	168	79	247
COG0060*	956	Isoleucyl-tRNA synthetase	172	42	214
COG0080	154	Ribosomal protein L11	170	61	231
COG0081	230	Ribosomal protein L1	168	61	229
COG0085†	1138	DNA-directed RNA polymerase, beta subunit	178	60	238
COG0087	288	Ribosomal protein L3	168	54	222
COG0091	157	Ribosomal protein L22	168	75	243
COG0092	240	Ribosomal protein S3	168	30	198
COG0093	130	Ribosomal protein L14	168	41	209
COG0094	182	Ribosomal protein L5	169	36	205
COG0096	131	Ribosomal protein S8	168	55	223
COG0097	177	Ribosomal protein L6P/L9E	168	65	233
COG0098	220	Ribosomal protein S5	168	110	278
COG0099‡	133	Ribosomal protein S13	168	49	217
COG0100	145	Ribosomal protein S11	169	51	220
COG0102	167	Ribosomal protein L13	168	54	222
COG0103	172	Ribosomal protein S9	168	52	220
COG0124*	472	Histidyl-tRNA synthetase	178	31	209
COG0143*†	646	Methionyl-tRNA synthetase	180	35	215
COG0172	442	Seryl-tRNA synthetase	177	37	214
COG0184	154	Ribosomal protein S15P/S13E	168	41	209
COG0186	122	Ribosomal protein S17	170	46	216
COG0197	175	Ribosomal protein L16/L10E	168	54	222
COG0200	166	Ribosomal protein L15	168	70	238
COG0201	445	Preprotein translocase subunit SecY	178	37	215
COG0202	323	DNA-directed RNA polymerase, alpha subunit	171	45	216
COG0256	178	Ribosomal protein L18	168	50	218
COG0495	854	Leucyl-tRNA synthetase	172	43	215
COG0522	199	Ribosomal protein S4 and related proteins	174	46	220
COG0525*‡	880	Valyl-tRNA synthetase	169	37	206
COG0533	375	Metal-dependent proteases with chaperone activity	168	35	203

Questões

- Onde podemos achar as sequências dos genes?
- Como determinar compartilhamento?
- Como preparar esses dados para construir uma árvore?
- Como construir uma árvore?
- Como saber se ela esta correta?

Fig. 1. Overview of the procedure.



F D Ciccarelli et al. Science 2006;311:1283-1287

Published by AAAS

Respostas

- Onde podemos achar as sequências dos genes?
  - Bancos de dados públicos (NCBI)
- Como determinar compartilhamento?
  - Através de *comparação de sequências*
- Como preparar esses dados para construir uma árvore?
  - Alinhamento múltiplo concatenado
- Como construir uma árvore?
  - Métodos de reconstrução filogenética
- Como saber se ela esta correta?
  - Inferência é um termo melhor do que construção
  - Argumentos probabilísticos
  - É preciso tomar cuidado com o fenômeno de Transferência Horizontal (Lateral) de Genes

Por que comparar sequências?

- Achar similaridades
  - Dadas 2 sequências, quão parecidas elas são?
  - É possível comparar em termos de DNA (nucleotídeos) e em termos de proteína (aminoácidos)
- É uma operação básica das buscas em bancos de sequências
  - Achar quais sequências do banco são parecidas com minha sequência-consulta (query)
  - A sequência-consulta é tipicamente uma sequência nova

Motivação (cont.)

- Construir famílias de proteínas
  - Saber quais organismos tem membros da família
  - Determinar uma “assinatura” para a família

- Construir filogenias
  - Entender a história evolutiva de genes e organismos
  - Ou seja, a comparação de sequências é uma operação fundamental para inferir a árvore da vida

Premissas

- Em geral buscamos sequências “aparentadas”
- Sequências “aparentadas” são similares
- “aparentadas” = **homólogas**
  - Descendem de um mesmo ancestral
- Descendentes sofreram diferentes mutações ao longo do tempo

Artigo mais recente onde vários dos conceitos vistos são usados

Agosto/2018

nature ecology & evolution

ARTICLES

https://doi.org/10.1038/s41559-018-0644-x

Integrated genomic and fossil evidence illuminates life's early evolution and eukaryote origin

Holly C. Betts[1], Mark N. Puttick[1,2], James W. Clark[1], Tom A. Williams[1,3], Philip C. J. Donoghue[1] and Davide Pisani[1,3]*

Establishing a unified timescale for the early evolution of Earth and life is challenging and mired in controversy because of the paucity of fossil evidence, the difficulty of interpreting it and dispute over the deepest branching relationships in the tree of life. Surprisingly, it remains perhaps the only episode in the history of life where literal interpretations of the fossil record hold sway, revised with every new discovery and reinterpretation. We derive a timescale of life, combining a reappraisal of the fossil material with new molecular clock analyses. We find the last universal common ancestor of cellular life to have predated the end of late heavy bombardment (>3.9 billion years ago (Ga)). The crown clades of the two primary divisions of life, Eubacteria and Archaebacteria, emerged much later (<3.4 Ga), relegating the oldest fossil evidence for life to their stem lineages. The Great Oxidation Event significantly predates the origin of modern Cyanobacteria, indicating that oxygenic photosynthesis evolved within the cyanobacterial stem lineage. Modern eukaryotes do not constitute a primary lineage of life and emerged late in Earth's history (<1.84 Ga), falsifying the hypothesis that the Great Oxidation Event facilitated their radiation. The symbiotic origin of mitochondria at 2.053–1.21 Ga reflects a late origin of the total-group Alphaproteobacteria to which the free living ancestor of mitochondria belonged.

Métodos desse artigo

- The dataset consists of 102 species and 29 universally distributed, protein-coding genes
- **Exercício**: compare essa lista com a lista dos genes universais usados por Cicarelli et al., 2006 (mostrados em slide anterior)

- Proteomes were downloaded from GenBank and putative orthologues were identified using BLAST. The top hits were compiled and aligned into gene-specific files in MUSCLE

Lista dos 29 genes utilizados por Betts et al.

Rps14bp (1)
Rps23bp (6)
Fun12p (14)
Rpl11ap (15)
Rsp3p (20)
Rps16ap (22)
Rpl1ap (24)
Rpl2bp (29)
Rpl23bp (30)
Rpl12ap (31)
Eft1p (33)
Kae1p (34)
Rps0bp (35)
Rps2p (36)
Rps5p (37)
Srp54p (40)
Tef1p (4)
Rli1p (5)
Dps1p (10)
Rpa190p (11)
Sec61p (12)
Cct5p (16)
Rfc2p (17)
Vma2p (23)
Map2p (25)
Rpl16ap (28)
Gln4p (32)
Rpa135p (39)
Srp101p (41)

Esses nomes correspondem aos nomes em *Saccharomyces cerevisiae*; o número entre parênteses é uma particularidade deste estudo, e é irrelevante para este exercício

Feita a motivação, vamos agora entender em detalhes...

- ...como se comparam sequências

Alinhamento de DNA

GTGGTGGCCTACGAAGGT
GTAGTGCCTTCGAAGGGT

Informalmente, um alinhamento significa colocar uma sequência “em cima da outra” para estabelecer uma correspondência 1:1 entre cada base da sequência “de cima” e cada base da sequência “de baixo”

Como avaliar um alinhamento?

- Sistema de pontuação
  - Match: +1
  - Mismatch: –1

Pontuação do alinhamento

GTGGTGGCCTACGAAGGT

GTAGTGCCTTCGAAGGGT

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

Esta é a nota desse alinhamento

É possível melhorar o alinhamento?

- Sim
- Pela introdução de espaços

Alinhamento com espaços

```
GTGGTGGCCTACGAA-GGT
GTAGTG-CCTTCGAAGGGT
```

Sistema de pontuação com espaços

- Match: +1
- Mismatch: –1
- Espaço: –2
- (O termo *buraco* deve ser usado para uma sequência de espaços)
  - Em inglês: *gaps*

Pontuação do alinhamento

```
GTGGTGGCCTACGAA-GGT
GTAGTG-CCTTCGAAGGGT
```

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

Justificativa para o 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 morte à célula ou organismo (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
- O conceito de similaridade entre duas sequências
  - É o valor da pontuação do alinhamento ótimo
- No exemplo anterior
  - Similaridade = 9

Comparação de sequências de aminoácidos

Pontuação de alinhamento de proteínas

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

% de identidade é uma medida simples mas válida de similaridade de sequências de proteínas

Aminoácidos se dividem em famílias

- Hidrofóbicos
  - Ala, Val, Phe, Pro, Met, Ile, Leu
- Com carga
  - Asp, Glu, Lys, Arg
- Polares
  - Ser, Thr, Tyr, His, Cys, Asn, Gln
  - Trp
- Gly

Twenty-One Amino Acids

⊕ Positive ⊖ Negative

• Side chain charge at physiological pH 7.4

A. Amino Acids with Electrically Charged Side Chains

Positive

Arginine (Arg) R

Histidine (His) H

Lysine (Lys) K

Negative

Aspartic Acid (Asp) D

Glutamic Acid (Glu) E

B. Amino Acids with Polar Uncharged Side Chains

Serine (Ser) S

Threonine (Thr) T

Asparagine (Asn) N

Glutamine (Gln) Q

C. Special Cases

Cysteine (Cys) C

Selenocysteine (Sec) U

Glycine (Gly) G

Proline (Pro) P

D. Amino Acids with Hydrophobic Side Chain

Alanine (Ala) A

Valine (Val) V

Isoleucine (Ile) I

Leucine (Leu) L

Methionine (Met) M

Phenylalanine (Phe) F

Tyrosine (Tyr) Y

Tryptophan (Trp) W

pKa Data: ©RC Handbook of Chemistry, v. 2010

Dan Gojocari, Department of Medical Biophysics, University of Toronto 2011

Mutações e proteínas

- Substituições que não alteram a estrutura da proteína tendem a ser preservadas (ou conservadas) durante a evolução
- A troca de um aminoácido de uma família por outro da mesma família em geral cai nessa categoria
- (Indels podem ter consequências mais drásticas)
- Então: como avaliar mismatches?

Matriz de substituição de amino ácidos BLOSUM62

```
#  Matrix made by matblas from blosum62.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 62
#  Entropy =   0.6979, Expected =  -0.5209
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1  0 -4
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3 -1  0 -1 -4
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3  3  0 -1 -4
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4 -1 -3 -3 -1  0 -1 -4 -3 -3  4  1 -1 -4
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1  1  0  0 -3  5  2 -2  0 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2  0  3 -1 -4
E -1  0  0  2 -4  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4 -2 -3 -3 -2  0 -2 -2 -3 -3 -1 -2 -1 -4
H -2  0  1 -1 -3  0  0 -2  8 -3 -3 -1 -2 -1 -2 -1 -2 -2  2 -3  0  0 -1 -4
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2 -3  1  0 -3 -2 -1 -3 -1  3 -3 -3 -1 -4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1 -4 -3 -1 -4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5 -1 -3 -1  0 -1 -3 -2 -2  0  1 -1 -4
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5  0 -2 -1 -1 -1 -1  1 -3 -1 -1 -4
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1 -3 -3 -1 -4
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2 -2 -1 -2 -4
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2  0  0  0 -4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5 -2 -2  0 -1 -1  0 -4
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11  2 -3 -4 -3 -2 -4
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1 -3 -2 -1 -4
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4 -3 -2 -1 -4
B -2 -1  3  4 -3  0  1 -1  0 -3 -4  0 -3 -3 -2  0 -1 -4 -3 -3  4  1 -1 -4
Z -1  0  0  1 -3  3  4 -2  0 -3 -3  1 -1 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2  0  0 -2 -1 -1 -1 -1 -1 -4
* -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4  1
```

Fonte: NCBI

Pontuação leva em conta a matriz

- Match: blosum62(i,i) sempre positivo (diagonal)
- Mismatch: blosum62(i,j) positivo, nulo, negativo
- Espaço: –2
- Pergunta: por que os valores da diagonal, embora positivos, são diferentes entre si?

```
H: GDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGYSYTAANKNKGIIWG
   GD EKGKK++  +C QCH V+     KTGP LHG+ GR +G   G+ Y+AANKNKG++W
C: GDYEKGKKVYKQRCLQCHVVDSTAT-KTGPTLHGVIGRTSGTVSGFDYSAANKNKGVVWT
```

Como obter alinhamentos ótimos?

- Precisamos de um algoritmo
- Algoritmos são diferentes de programas
  - Algoritmo é um método abstrato
  - Programa é uma encarnação física (numa linguagem de programação) de um algoritmo
  - Um programa pode ser executado num computador
  - Dizemos que um certo algoritmo foi implementado em tal ou qual linguagem
- <u>Nunca</u> escrevam algoritimo!

Algoritmo para achar alinhamentos ótimos de DNA

- Desenvolvido com a técnica de Programação dinâmica
- Técnica desenvolvida na década de 1950 por Richard Bellman
- Não é programação e não é dinâmica!
  - É um nome histórico que se tornou um jargão técnico

Programação Dinâmica

- PD se usa para problemas que tem uma estrutura de subproblemas
- Num alinhamento com sequências *s* e *t* um subproblema é qualquer alinhamento entre *s′* e *t′* tal que

  s′ = um prefixo de *s* e *t′* = um prefixo de *t*

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

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

Um prefixo

Ideia básica da PD

- Achar soluções de subproblemas e armazená-las numa tabela (matriz)
- Para achar a solução ótima:
  - Ir achando as soluções na direção dos subproblemas menores para os maiores
  - Último elemento da tabela a ser preenchido contém a solução do problema “completo”
- Vamos demonstrar PD para conseguir o alinhamento ótimo de GTC com GATC

Aqui está a sequência “de baixo”

A tabela ou matriz

	j	0	1	2	3	4
i		t	G	A	T	C
0	s					
1	G					
2	T					
3	C					

Aqui está a sequência “de cima”

Preenchimento da tabela

- Este processo precisa começar com o “menor subproblema possível”. Qual seria?
  - Quando pelo menos uma das sequências é vazia

- Inicialização: Alinhar *s* com cadeia vazia e alinhar *t* com cadeia vazia

Inicialização da zero-ésima linha e da zero-ésima coluna

	j	0	1	2	3	4
i		t	G	A	T	C
0	s	0	-2	-4	-6	-8
1	G	-2				
2	T	-4				
3	C	-6				

Continuação

- Alinhar caracter *X* com caracter *Y*
- usamos a palavra ‘caracter‘ como um genérico para ‘nucleotídeo’ ou ‘aminoácido’
- 3 possibilidades
  - *X* com *Y*
    - Aplicar pontuação respectiva, dependendo se for DNA ou proteína
  - *X* com espaço
    - Cobrar -2
  - *Y* com espaço
    - Cobrar -2

Preenchimento de (1,1)

- Significa determinar qual é o melhor alinhamento dos prefixos de *s* e *t* com apenas um caracter cada um
- Alternativas

```
G-        -G        G
-G        G-        G
```

Todos eles usam valores determinados na inicialização

	j	0	1	2	3	4
i		t	G	A	T	C
0	s	0	-2	-4	-6	-8
1	G	-2	1			
2	T	-4				
3	C	-6				

	j	0	1	2	3	4
i		t	G	A	T	C
0	s	0	-2	-4	-6	-8
1	G	-2	1	-1	-3	-5
2	T	-4	-1	0	0	-2
3	C	-6	-3	-2	-1	1

Exercícios

- Inventar duas sequências de DNA curtas e “rodar” (na mão) o algoritmo de PD
- Para se auto-corrigir:
- Baixe a demonstração (‘demo’) do seguinte site
  http://www.codeproject.com/Articles/304772/DNA-Sequence-Alignment-using-Dynamic-Programming-A

- Exige registro, mas me pareceu seguro
- Tem também código fonte, para quem estiver interessado

Complexidade computacional de PD

- Queremos saber quanto tempo gasta o algoritmo
- Mas algoritmos são abstratos, não estão associados a computadores físicos; então o que significa "tempo"?
- Contamos número de operações elementares
  - Operações aritméticas
  - Uso de posições da memória
  - Etc
- Porém queremos apenas algo aproximado
- Apenas uma ordem de grandeza

Complexidade computacional

- Queremos expressar o número de operações como uma função matemática do tamanho das entradas (a variável n)
- Por exemplo
  - n (linear)
  - n^2 (quadrático)
  - $n \log n$
- Vamos desprezar constantes (n e $30n$ dão na mesma)
- Só nos interessa o termo de maior grau (desprezar os demais)
  - $7n^2 + 1400n + 3000\log n$ vira apenas n^2
- Vamos usar a notação $O(f(n))$ para denotar essas aproximações

Complexidade computacional de PD

- A matriz tem tamanho n+1 por m+1
- Todos os elementos da matriz precisam ser preenchidos
- Supondo tempo constante para o preenchimento
  - $n+1 \times m+1 = nm + n + m + 1$
  - $O(nm)$
  - Se $n \approx m$, então resulta $O(n^2)$
    - Quadrático
- Memória: é quadrático também (pois é o tamanho da matriz)

Algoritmo de PD expresso em pseudo-código

Algorithm *Similarity*

input: sequences s and t

output: similarity between s and t

$m \leftarrow |s|$

$n \leftarrow |t|$

for $i \leftarrow 0$ **to** m **do**

$\quad a[i, 0] \leftarrow i \times g$

for $j \leftarrow 0$ **to** n **do**

$\quad a[0, j] \leftarrow j \times g$

for $i \leftarrow 1$ **to** m **do**

$\quad$ **for** $j \leftarrow 1$ **to** n **do**

$$a[i, j] \leftarrow \max(a[i-1, j] + g,\ a[i-1, j-1] + p(i, j),\ a[i, j-1] + g)$$

return $a[m, n]$

g é o valor da penalização de um espaço

O valor de *p(i,j)* vai depender de *i = j ou i ≠ j*

Penalização de espaços pode ser mais sofisticada

GTGGTGGCCTACGAAGGT

GTGGTCGC---CGAAGGT

GTGGTGGCC-ACGAAGGT

GT-GTCGCCTACGA-GGT

- No sistema de pontuação apresentado, *k* espaços consecutivos (um buraco ou *gap*) custam o mesmo que *k* espaços separados
- Seria melhor distinguir os dois casos
- *k* espaços consecutivos geralmente aparecem por causa de um único evento evolutivo
- *k* espaços separados geralmente aparecem por causa de *k* eventos distintos

Penalização de espaços feita por uma função matemática

- k = número de espaços
- $p(k) = a + bk$
- $p(k)$ é subtraído do score
- a = custo para abrir um buraco
- b = custo para continuar um buraco
- Por exemplo: $p(k) = 2 + k$
- 5 espaços consecutivos custam 7
- 5 espaços separados custam 15
- Compare com a função implícita do sistema simples: $p(k) = 2k$

Algoritmo de PD com penalização de buracos por função afim

- O algoritmo é mais complexo
  - São necessárias 3 tabelas ao invés de 1
- Mas a complexidade permanece a mesma (quadrática)
- Esta versão é conhecida como Algoritmo de Smith-Waterman

Smith-Waterman

$a[i, j]$ = maximum score of an alignment between $s[1..i]$ and $t[1..j]$ that ends in $s[i]$ matched with $t[j]$.
$b[i, j]$ = maximum score of an alignment between $s[1..i]$ and $t[1..j]$ that ends in a space matched with $t[j]$.
$c[i, j]$ = maximum score of an alignment between $s[1..i]$ and $t[1..j]$ that ends in $s[i]$ matched with a space.

The entries (i, j) of these arrays depend on previous entries according to the following formulas, valid for $1 \leq i \leq m$ and $1 \leq j \leq n$:

$$a[i, j] = p(i, j) + \max \begin{cases} a[i-1, j-1] \\ b[i-1, j-1] \\ c[i-1, j-1] \end{cases}$$

$$b[i, j] = \max \begin{cases} -(h+g) + a[i, j-1] \\ -g + b[i, j-1] \\ -(h+g) + c[i, j-1] \end{cases}$$

$$c[i, j] = \max \begin{cases} -(h+g) + a[i-1, j] \\ -(h+g) + b[i-1, j] \\ -g + c[i-1, j]. \end{cases}$$

As before, $p(i, j)$ indicates the score of a matching between $s[i]$ and $t[j]$.

- Se a função de penalização de buracos for genérica [ex. $p(k) = a + b \log k$)], então a complexidade passa para $O(n^3)$
  - Esta versão é conhecida como Algoritmo de Needleman-Wunsch

Como recuperar os alinhamentos a partir da matriz preenchida?

- “traceback”
- começando na célula inferior direita, é possível saber qual foi a célula de origem do seu valor (esquerda, superior, ou diagonal)
- Essa informação me permite reconstituir o alinhamento “do fim para o começo”
- é necessário tempo adicional para essa reconstituição, mas esse tempo é linear no tamanho das sequências