Sandra Manuela Gonçalves Oliveira

Ferramentas informáticas para a selecção e o alinhamento de genes de DNA mitocondrial

Aplicação ao estudo de diversidade da ordem Primatas

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Engenharia Matemática

Departamento de Matemática Pura e Matemática Aplicada
Faculdade de Ciências da Universidade do Porto
2007

Agradecimentos

À Doutora Luísa Pereira, Mestre e amiga um muito obrigada especial, por toda a sua sabedoria, ensinamento, dedicação, empenho, ajuda e força, principalmente nos momentos mais difíceis.

Ao Professor Doutor Gueorgui Smirnov, pelo seu contributo, paciência, disponibilidade, que tornou possível a realização deste trabalho.

Ao Professor Doutor Manuel Sobrinho Simões, presidente do IPATIMUP, por me proporcionar uma investigação num excelente ambiente, com óptimas condições de trabalho e um espírito de equipa inigualável.

A todo o grupo de Diversidade Genética e Bioinformática, Carla, Fernando, Joana, Marta e Verónica pelo companheirismo, espírito de equipa, boa disposição, entreajuda e apoio incondicional. Um obrigado muito sincero e especial ao Fernando pela preciosa ajuda, paciência, dedicação, empenho, sabedoria e pela colaboração aqui presente.

Aos meus conhecidos, colegas e aos meus amigos, em especial ao Bruno pelo indefectível apoio de incentivos e compreensão nos bons e maus momentos deste percurso.

E a vós que estais no principio, no meio e no fim desta realidade compartilhada para quem muitas vezes fui

Filha, irmã e namorada presente/ausente.

Resumo

O trabalho desenvolvido nesta tese teve um carácter prático, pretendendo responder a algumas das necessidades básicas resultantes da análise de sequências de DNA mitocondrial com aplicações filogenéticas. De facto, nos últimos anos tem-se vindo a acumular uma enorme quantidade de dados genéticos, depositados em bases de dados informatizadas, algumas das quais estão a ser melhoradas no sentido de haver uma revisão biológica dos dados, de modo a representarem as espécies. As potencialidades informativas que podem ser retiradas destas bases de dados estão ainda dependentes do desenvolvimento e do melhoramento de programas que automatizem a selecção e organização dos ficheiros depositados.

Assim, desenvolveram-se duas ferramentas informáticas, operativas na plataforma Windows, para fácil utilização por biólogos e que utilizam como *input* os ficheiros no formato da base de dados genética com mais sucesso, o GenBank. Uma das ferramentas informáticas permite a selecção de qualquer um dos 38 genes que constituem as moléculas de DNA mitocondrial dos mamíferos. Esta ferramenta permite uma rápida e automatizada preparação dos ficheiros GenBank para a análise da heterogeneidade genética das diversas regiões que compõem o DNA mitocondrial.

A outra ferramenta informática testa um novo algoritmo de alinhamento de sequências, isto é, de procura de homologias entre as bases que constituem duas sequências. O algoritmo implementado consiste na procura sequencial de blocos de bases iguais nas duas sequências, a partir do maior tamanho até ao menor (igual a 1). O número de bases não emparelháveis, bem como a localização e tamanho dos blocos, são indicados. Deste modo, não só o alinhamento das duas sequências é disponibilizado, como uma medida de homologia entre as duas sequências. O programa tem outras vantagens comparativamente aos programas existentes para alinhamento de sequências, permitindo (1) alinhar a sequência na totalidade ou só um dos 38 genes; (2) no caso da análise da sequência na sua totalidade, é implementada a circularidade da molécula, não ficando nenhum fragmento por alinhar devido a desfasamentos de numeração entre as duas sequências.

A operacionalidade das ferramentas informáticas desenvolvidas foram testadas em sequências de DNA mitocondrial de Primatas.

Abstract

The work developed in this thesis has an applied purpose, dealing with basic needs of statistical analyses of mitochondrial DNA sequences for phylogenetic applications. Recently, a huge quantity of genetic data is being deposited on online databases, some aiming good quality standards in order to function as representative of species genetic diversity. The information that can be retrieved from these databases still depends on the development and improvement of software tools allowing automatic selection and organisation of deposited files.

We developed two software tools in Windows platform, to encourage its use by biologists, that use as input files the format GenBank, the most used genetics database. One of the tools allows the selection of any of the 38 genes which constitute the mitochondrial DNA molecule of mammals. This tool allows a fast preparation of GenBank files for the analyses of genetic heterogeneity between the mitochondrial DNA regions.

The other software tool uses a new algorithm for sequence alignment, that is, the searching of homology between the bases constituting two sequences. The algorithm searches sequentially equal blocks of bases in the two sequences, from the biggest one to the lowest (equal to one base). The number of no-matching bases and the location and size of the blocks are given by the program. In this way, one obtains the alignment and a measure of homology between the two sequences. The program has other advantages, allowing (1) to aligning the sequence in its totality or only one of the 38 genes; (2) in the case of aligning the total sequence, to implement the circularity of the molecule, avoiding not-aligned fragments due to numbering differences between the two sequences.

Analyses performed with mitochondrial DNA from Primates were used as tests for both tools.

Índice

Agradecimentos....3
Resumo....4
Abstract....5
Abreviaturas....8

1 - Introdução....10
1.1 O DNA....10
1.2 O DNA mitocondrial....12
1.3 Diversidade genética....14
1.4 As bases de dados genéticas....16
1.5 Alinhamento de sequências....19
1.6 Objectivos....21

2 - Material e Métodos....22
2.1 Implementação do programa de selecção de genes do mtDNA....22
2.2 Base de dados de sequências de mtDNA de Primatas e medidas de diversidade genética....24
2.3 Algoritmo do programa de alinhamento de genes do mtDNA....26
2.4 Implementação do programa de alinhamento de genes do mtDNA....27

3 - Resultados....30
3.1 Aplicação do programa de selecção de genes mtDNA aos Primatas....30
3.2 Comparação de parâmetros de diversidade genética entre os genes do mtDNA dos Primatas....33
3.3 Aplicação do programa de alinhamento de genes mtDNA aos Primatas....36

4 - Conclusões....41

Referências bibliográficas....44

Anexo 1 – Ficheiro GenBank do Primata *Pan troglodytes*....47

Anexo 2 – *Output* do programa de selecção do gene tRNA-Gln dos Primatas.....55
Anexo 3 – Código fonte do programa de selecção de genes do mtDNA.....57
Anexo 4 - *Output* (reorganizado em quatro colunas) do programa de alinhamento dos genes do mtDNA para as sequências totais de *Homo sapiens* e *Pan troglodytes*.....65
Anexo 5 - Código fonte do programa de alinhamento de genes do mtDNA.....74

Abreviaturas:

A – adenina;

Ala – alanina;

Arg – arginina;

Asn – asparagina;

Asp – ácido aspártico;

bp- *base pairs*; pares de bases;

C – citosina;

Cys – cisteína;

DNA – ácido desoxirribonucleico;

G – guanina;

Gln – glutamina;

Glu – ácido glutâmico;

Gly – glicina;

His – histidina;

Ile – isoleucina;

Leu – leucina;

Leu (CUN) – Leucina codificada pelo tripleto CUN;

Leu (UUR) – Leucina codificada pelo tripleto UUR;

Lys – lisina;

Met – metionina;

mtDNA – DNA mitocondrial;

NCBI – *National Center for Biotechnology Information*;

NHI – *National Institutes of Health*;

Phe – fenilalanina;

Pro – prolina;

RNA – ácido ribonucleico;

ROS – *reactive oxigen species*; espécies reactivas de oxigénio;

rRNA – ácido ribonucleico ribossómico;

Ser – serina;

Ser (AGY) – Serina codificada pelo tripleto AGY;

Ser (UCN) – Serina codificada pelo tripleto UCN;

T – timina;

Thr – treonina;

tRNA – ácido ribonucleico de transferência;

Trp – triptofano;

Tyr – tirosina;

URL – *Uniform Resource Locator*;

Val – valina;

1 - Introdução

1.1 - O DNA

A molécula da vida é o ácido desoxirribonucleico (na sua sigla internacional DNA). Esta molécula é constituída por elementos designados nucleótidos. Todos os nucleótidos possuem um açúcar (pentose) do tipo desoxirribose, um grupo fosfato e uma base azotada. No DNA, as bases azotadas são quatro, designando-se adenina (abreviadamente A), timina (T), guanina (G) e citosina (C). As bases possuem dois tipos de estrutura ligeiramente diferente: uma com um anel azotado simples, designando-se por pirimidinas, como a T e a C; e a outra com um anel azotado duplo, chamando-se purinas, como a A e a G.

Este alfabeto de apenas quatro letras possui a informação para as características dos diferentes seres vivos e o conjunto de todas as letras de um ser vivo designa-se genoma. O genoma humano é constituído por 23 pares de cromossomas, que se localizam no núcleo das células. Muito do texto não tem ainda significado biológico, não se sabendo qual a sua função e sendo extensamente constituído por pequenas sequências de texto altamente repetidas; as partes que codificam informação genética designam-se por genes.

Com a descoberta da estrutura em dupla hélice, por Watson e Crick, em 1953, ficou claro não só como a informação contida nas quatro bases era duplicada, aquando a divisão celular, mas também como era transcrita para a transmissão da ordem nela contida. De facto, as quatro bases estão sempre emparelhadas em dois arranjos: a uma timina de uma cadeia corresponde uma adenina na cadeia complementar, e vice-versa; a uma guanina numa cadeia corresponde uma citosina na outra, e vice-versa. Assim sendo, basta ocorrer a abertura da dupla hélice para que a construção de duas novas cadeias se faça de forma organizada por complementaridade das cadeias já existentes.

A informação biológica contida nos genes tem que ser traduzida para outra linguagem, descodificada em proteína, a qual é interpretada pela célula. As proteínas são constituídas por unidades designadas aminoácidos e existem 20 aminoácidos diferentes presentes nos mamíferos. A passagem da informação das bases para aminoácidos faz-se rigidamente segundo um código genético (Figura 1.1). Como só existem quatro bases diferentes para significar 20 aminoácidos, as bases têm que estar organizadas em arranjos de três, designados por codão ou tripleto. Estes arranjos dão 64 tripletos, mais do que seria necessário para codificar os 20 aminoácidos. O que se verificou é que alguns aminoácidos são codificados por vários codões, podendo ir de um a quatro. Esta propriedade do código genético é geralmente

designada por redundância. Alguns dos codões não codificam um aminoácido, mas são interpretados como sinais de pontuação, incluindo o STOP. O primeiro codão de todas as proteínas no genoma nuclear nos mamíferos é o AUG que significa INICIAÇÃO, mas quando se encontra no meio da informação significa o aminoácido metionina.

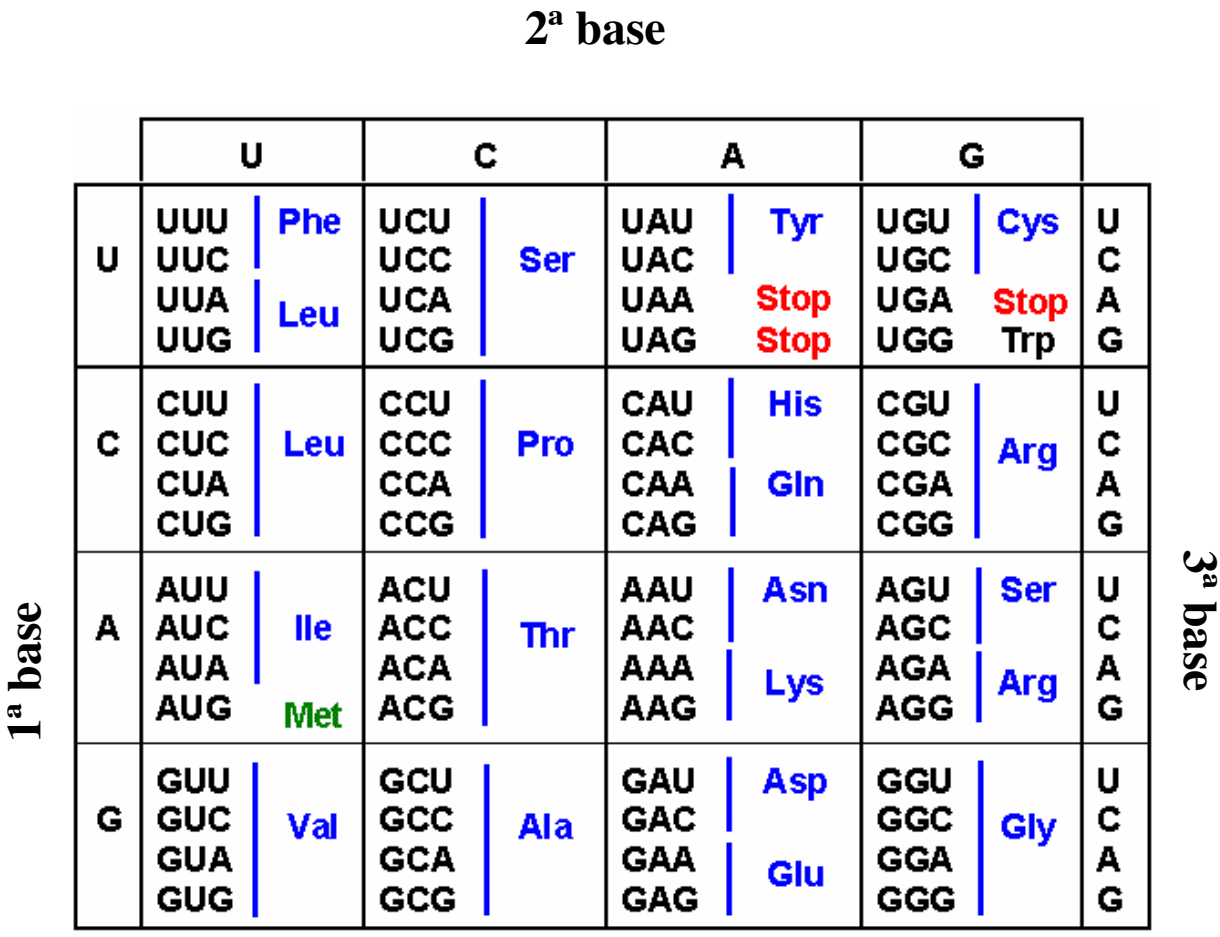


Figura 1.1 - Código genético nuclear dos mamíferos. As siglas de três letras designam os diferentes aminoácidos: Phe – fenilalanina; Leu – leucina; Ile – isoleucina; Met – metionina; Val – valina; Ser – serina; Pro – prolina; Thr – treonina; Ala – alanina; Tyr – tirosina; His – histidina; Gln – glutamina; Asn – asparagina; Lys – lisina; Asp – ácido aspártico; Glu – ácido glutâmico; Cys – cisteína; Trp – triptofano; Arg – arginina; Gly – glicina.

1.2 - O DNA mitocondrial

Para além do genoma nuclear, as células dos mamíferos possuem um organelo celular, a mitocôndria, que contém várias cópias do seu próprio DNA (Figura 1.2).

As mitocôndrias são organelos essenciais à vida porque é nelas que se produz, através da respiração celular, a maior parte da energia que as células necessitam para desempenharem as suas funções. A respiração celular é um processo eficiente de produção de energia biológica, a partir dos açúcares (glicose) ingeridos e na presença de oxigénio. As mitocôndrias são, por isso, locais da célula onde decorrem inúmeras reacções químicas, com a produção de compostos potencialmente tóxicos e altamente reactivos e oxidativos (ROS, *reactive oxygen species*), que têm que ser eficazmente eliminados.

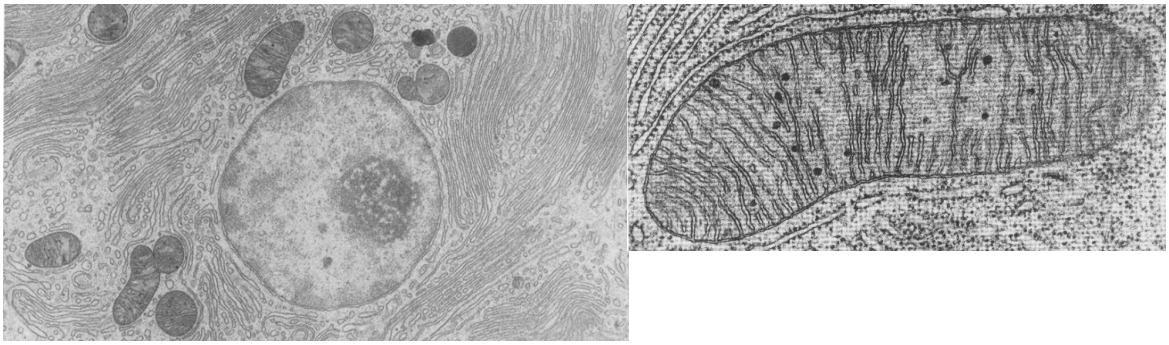


Figura 1.2 - Imagens de microscopia electrónica. Do lado esquerdo, porção de uma célula de pâncreas, evidenciando o núcleo central (com o DNA disperso) e as mitocôndrias em forma de bastão, no espaço que rodeia o núcleo, o citoplasma. Do lado direito, uma ampliação de uma mitocôndria, mostrando estar rodeada por uma dupla membrana, a interior da qual forma invaginações em forma de dedo ou crista; os grânulos mais escuros no interior são as moléculas de DNA mitocondrial.

O DNA mitocondrial (abreviadamente mtDNA) consiste numa molécula também em cadeia dupla, composta pelos mesmos nucleotidos e mesmas quatro bases, mas é circular e não linear. Para além disso, o mtDNA é muito compactado, sendo maioritariamente codificante, logo, não possuindo quase elementos repetitivos sem significado biológico, em quase todos os organismos. Por exemplo, o mtDNA humano (Figura 1.3) tem cerca de 16.569 pares de bases (Anderson et al., 1981; revisto por Andrews et al., 1999), 93% das quais representam 37 genes (2 RNAs ribossómicos ou rRNAs, 22 RNAs de transferência ou tRNAs

e 13 proteínas da cadeia respiratória mitocondrial), sendo por isso designada região codificante, e apenas 1122 bases constituem a região controlo ou D-loop, sem genes, mas com os locais para iniciação da duplicação e transcrição da molécula.

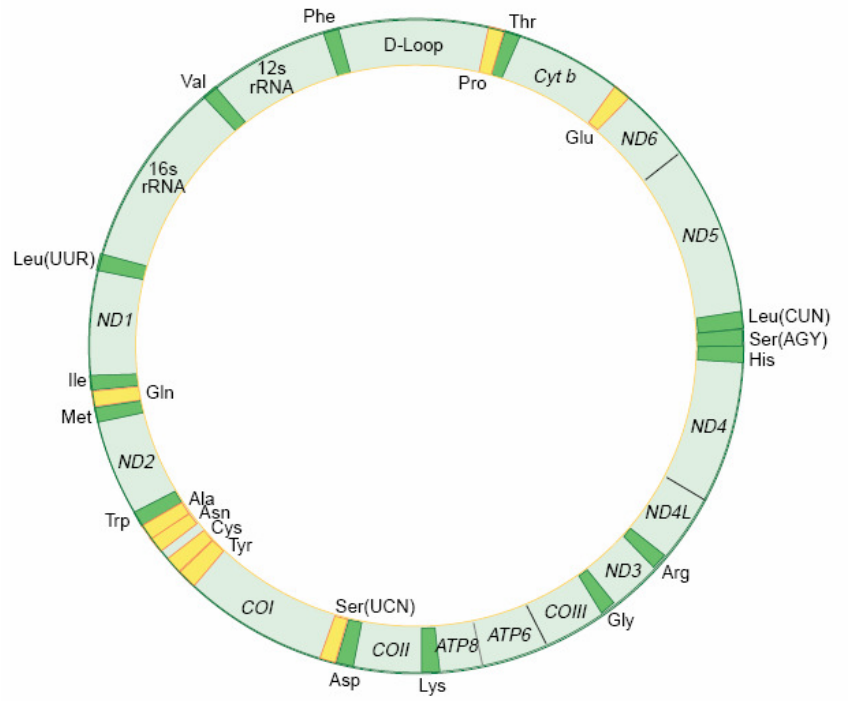


Figura 1.3 – Representação gráfica da molécula de mtDNA humano, evidenciando as localizações dos diversos genes na região codificante e o D-Loop ou região controlo.

Curiosamente, o mtDNA é transmitido apenas pela mãe (Giles et al., 1980), num processo diferente da transmissão partilhada pelos progenitores do DNA nuclear. Pensa-se que as mitocôndrias do espermatozóide que fecundou um oócito são destruídas após a fertilização, não contribuindo para os genes do embrião. Deste modo, para o mtDNA não vai ocorrer um fenómeno importantíssimo denominado recombinação, o qual é responsável pela troca de material genético entre os cromossomas nucleares masculino e feminino. Pode-se, assim, definir linhagens mitocondriais maternas, isto é, todos os indivíduos aparentados pelo lado materno vão possuir a mesma molécula de mtDNA, excepto quando ocorre uma mutação.

1.3 - Diversidade genética

As mutações são alterações aleatórias do material genético, em que a sequência de bases que o constituem é modificada. Muitas vezes reserva-se o termo mutação para a designação de uma alteração que ocorre numa zona codificante, num gene, em que há alteração da proteína codificada, conferindo vantagem ou desvantagem ao indivíduo. Quando a alteração ocorre numa zona não codificante, utiliza-se a designação polimorfismo, desde que atinja pelo menos uma frequência de 1% na população em estudo; neste caso, salvas raras excepções, será em termos práticos neutro, não afectando o indivíduo.

As mutações podem ter diversas origens e afectar uma porção maior ou menor do genoma. No caso das mutações pontuais, em que há modificação de apenas uma base, podem ser classificadas como delecções (perda), inserções (ganho) e substituições de uma base por outra. As substituições podem ser ainda classificadas em transições e em transversões. Uma transição é a substituição de uma base purina por outra purina, ou de uma pirimidina por outra pirimidina. Uma transversão é a substituição de uma base purina por uma pirimidina, ou vice-versa. As transições são mutações mais frequentes que as transversões (Belle et al., 2005).

A mutação é directamente proporcional ao número de transmissões, passagens de material genético entre indivíduos, dependendo do tempo, e directamente proporcional ao tamanho do genoma. Pode-se assim, estimar as taxas de mutação para os diversos genomas, nas diferentes espécies. Para a espécie humana, *Homo sapiens*, a taxa de mutação pontual no genoma nuclear é de cerca de 10^{-9} por base por ano, enquanto que no mtDNA é cerca de 10 vezes mais elevada na região codificante (10^{-8} por base por ano) e 100 vezes no D-loop (10^{-7} por base e por ano) (Ingman et al., 2000). A taxa de mutação mitocondrial é mais elevada por vários motivos: (1) o facto de o mtDNA não estar envolvido por proteínas – as histonas – tal como o DNA nuclear está; (2) não ter mecanismos de reparação de mutações tão eficientes como o DNA nuclear; (3) durante o processo de produção de energia são produzidos os radicais livres, agentes oxidantes que podem danificar as moléculas de mtDNA.

Deste modo, sequenciando o mtDNA de diferentes espécies pode-se construir uma árvore filogenética, que ilustre quão conservadas ou divergentes são as sequências de mtDNA, logo, quão conservadas ou divergentes são as diversas espécies. Esta informação do DNA, e também das proteínas por si codificadas, que regista a história evolutiva, levou Zuckerlandl e Pauling, em 1965, a sugerir a hipótese do relógio molecular da evolução

(revisto em Ayala, 1986). O relógio molecular dataria os eventos evolutivos, tornando possível reconstruir a história filogenética.

A teoria de neutralidade da evolução molecular, desenvolvida por Kimura em finais de 1960 e inícios de 1970, postulava que as taxas de evolução molecular seriam estocasticamente constantes porque a vasta maioria das diferenças moleculares entre as espécies eram selectivamente neutras, sendo governadas pela deriva genética (efeito do acaso na sobrevivência de diversidade genética). Assim sendo, o relógio molecular seria um relógio estocástico. Uma variedade de testes mostrou, contudo, que tal não é a realidade, sendo as taxas evolutivas demasiado elevadas para serem consistentes com a teoria de neutralidade. Mas este facto não invalida o relógio molecular (Ayala, 1986), sendo a evolução suficientemente regular para ser aplicada em muitas situações como um relógio, tendo apenas que se ter em atenção as incertezas quanto às propriedades do relógio molecular (por exemplo, acerca das circunstâncias que podem levar a grandes oscilações nas taxas de substituição de tempos a tempos ou de linhagem para linhagem).

Quando estes conceitos teóricos foram desenvolvidos estavam descritas poucas sequências de DNA ou de proteínas de diferentes organismos que permitissem testar a sua validade. Uma das primeiras moléculas a ser usada para essa validação foi a proteína mitocondrial citocromo c, a qual mostrou um comportamento aceitavelmente regular para funcionar como relógio molecular (Ayala, 1986).

A maior taxa de mutação do mtDNA e a ausência de recombinação neste genoma fazem com que continue a ser o eleito para a realização de estudos filogenéticos. A maior taxa de mutação permite uma maior capacidade de distinção entre os indivíduos e a ausência de recombinação torna mais fácil a reconstrução da história evolutiva da molécula. Actualmente, o desenvolvimento tecnológico veio proporcionar a publicação de elevado número de sequências completas de mtDNA de inúmeras espécies ao longo da escala evolutiva. É assim possível utilizar toda a informação do mtDNA para estudos filogenéticos.

Recentemente, e especialmente para a espécie humana, demonstrou-se que existia uma grande heterogeneidade na taxa de mutação em algumas regiões particulares da molécula, como na designada região hipervariável II da região controlo ou D-loop (Meyer et al., 1999), tendo-se iniciado uma discussão de que talvez a taxa de mutação do mtDNA fosse demasiado rápida para ser usada fidedignamente em estudos filogenéticos (ver Macaulay et al., 1997 *versus* Howell et al., 1996). Mais uma vez, parece que estas irregularidades não invalidam a utilização do mtDNA como relógio molecular, mas é claro que estudos comparativos das diferentes regiões da molécula são necessários.

Muito provavelmente, as diferentes taxas de mutação nas diferentes regiões da molécula levarão a vários relógios moleculares, o que permitirá a obtenção de várias estimativas, com a possibilidade de comparação.

Estes estudos de heterogeneidade de variabilidade genética entre as diferentes porções da molécula de mtDNA estão numa fase incipiente. Dependem do desenvolvimento de ferramentas que permitam alguma flexibilidade e automatização da selecção de diferentes regiões da molécula. Uma ferramenta deste tipo, que permite a análise particular de genes do mtDNA, foi criada por Vasconcelos et al. (2005), denominada MamMiBase (http://www.mammibase.lncc.br). Esta ferramenta permite comparar os 13 genes mitocondriais codificantes de proteínas entre alguns mamíferos. Outra base de dados, disponibiliza os alinhamentos dos 22 tRNAs em várias espécies – é denominada Mamit-tRNA e está depositada no URL http://mamit-trna.u-strasbg.fr (Helm et al., 2000). Os alinhamentos desta base de dados têm em atenção não só os dados filogenéticos de várias espécies como também incluem dados de estrutura tridimensional dos tRNAs, mas não permite ao operador fazer qualquer tipo de análise, sendo apenas um reportório de dados.

1.4 - As bases de dados genéticas

Na sua forma mais básica, a Genética é informação digital: milhões e milhões das quatro bases que codificam a informação para a Vida.

Recentemente, os desenvolvimentos tecnológicos, nomeadamente da sequenciação automática, método que permite identificar a sequência de bases numa dada região do genoma, estão a levar a uma acumulação enorme de informação para os mais diversos genomas. A gestão desta informação trouxe muitos desafios bioinformáticos e inúmeras questões éticas, como a disponibilização pública ou o patenteamento de dados genéticos.

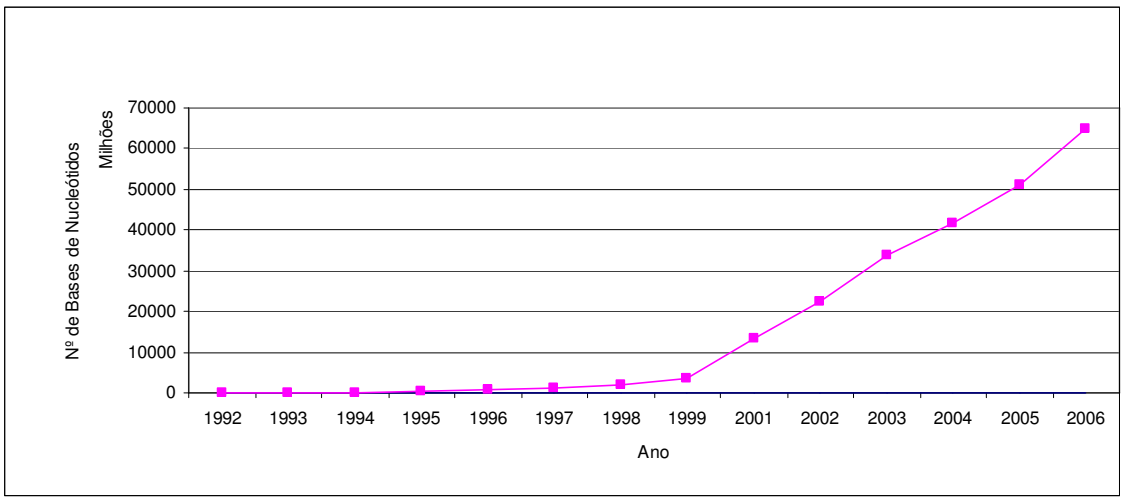
A base de dados genética com maior sucesso é o GenBank, localizada no URL http://www.ncbi.nlm.nih.gov. O GenBank é mantido pelo NCBI, *National Center for Biotechnology Information*, um ramo pertencente ao NIH, *National Institutes of Health*. Foi criado em 1982 em resposta a uma necessidade científica crítica de gerar um repositório acessível, centralizado e actualizado de sequências genéticas. Desde essa altura, inclui todas as sequências de nucleótidos e de proteínas conhecidas a partir de 1967. A figura 1.4 ilustra o aumento significativo de dados depositados no GenBank ao longo do tempo.

Cada sequência depositada no GenBank inclui uma descrição, contendo o nome, a taxonomia, a identificação do código das regiões e outros locais biologicamente significantes, as unidades de transcrição, os locais de mutação, as modificações e as repetições. Inclui ainda as referências bibliográficas e um código identificador único para cada sequência publicada, denominado *Accession Number*. Um exemplo de ficheiro GenBank está reproduzido no anexo 1.

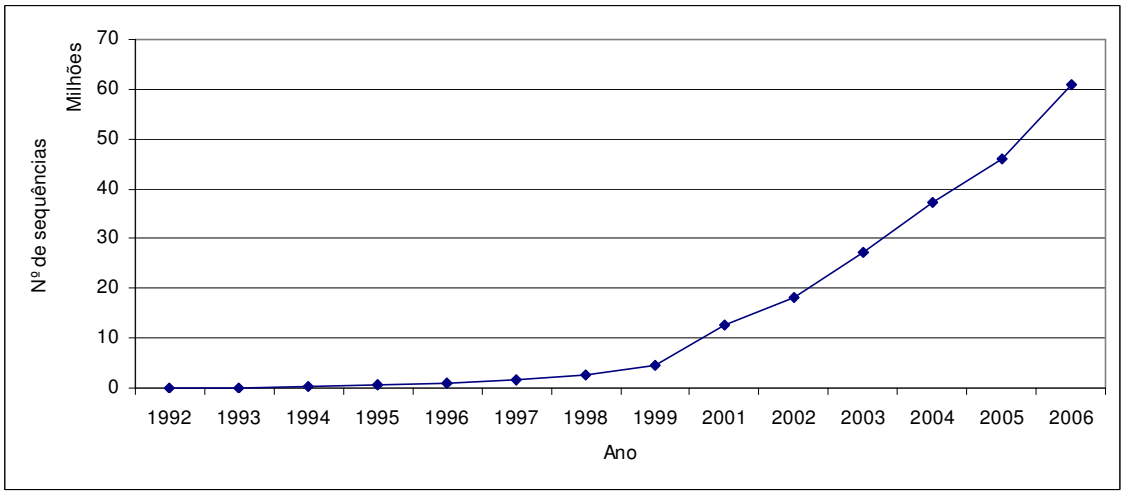
Com base nas sequências publicadas no GenBank, o NCBI criou uma outra base de dados designada por *NCBI Reference Sequence*, ou abreviadamente RefSeq, (http://www.ncbi.nlm.nih.gov/RefSeq/). A RefSeq é uma base de dados revista, contendo sequências não redundantes de genomas, transcritos (RNA) e proteínas (Pruitt et al. 2005). Para esta base de dados é seleccionada uma sequência de um certo genoma, transcrito ou proteína de um certo indivíduo que vai ficar representativo da espécie. Deste modo, a RefSeq serve de base para estudos médicos, funcionais e filogenéticos e pretende providenciar uma referência estável para identificação e caracterização de genes, análise de mutações, estudos de expressão, descoberta de polimorfismos e análises comparativas (Pruitt et al. 2005).

As principais características da RefSeq são: 1) não redundância; 2) sequências nucleotídicas e proteicas explicitamente relacionadas; 3) actualizações que evidenciem o conhecimento recente dos dados de sequência e biológicos; 4) validação de dados e consistência do formato; 5) *Accession Numbers* distintivos (todos incluem um carácter '_'); 6) revisão contínua por membros do NCBI e colaboradores, sendo todas as revisões indicadas no ficheiro.

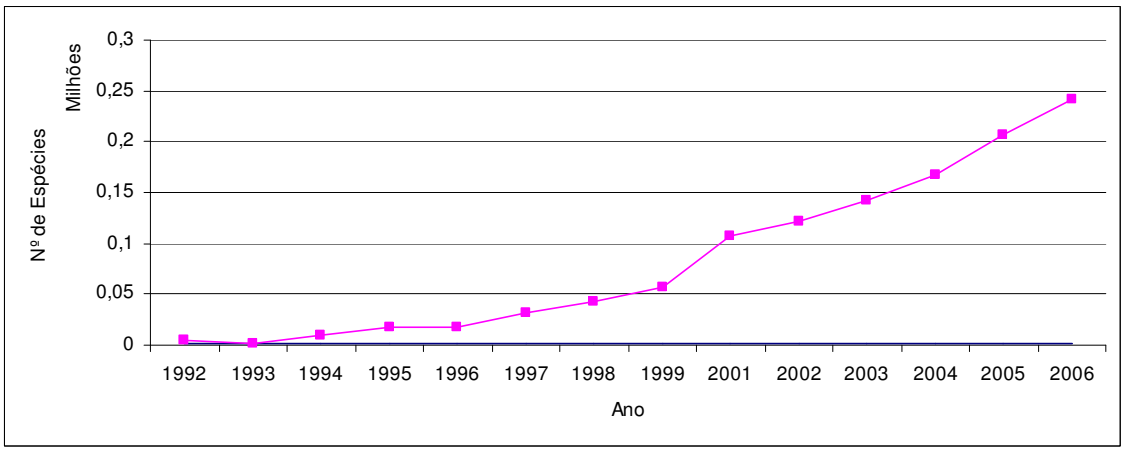
Existe uma secção da RefSeq que inclui apenas as sequências de mtDNA de mamíferos (http://www.ncbi.nlm.nih.gov/genomes/ORGANELLES/40674.html). Até ao dia 12 de Julho de 2007 existiam 213 sequências da classe Mammalia, das quais 23 são da ordem Primatas.



A



B



C

Figura 1.4 – Dados depositados no GenBank ao longo dos anos: (A) número de bases de nucleótidos; (B) número de sequências; (C) número de espécies. Dados compilados das referências Benson et al. (1993, 1994, 1996-2000 e 2002-2007).

1.5 - Alinhamento de sequências

Para a reconstrução da filogenia das espécies e outros cálculos comparativos, o primeiro passo, após a obtenção das sequências de bases de uma dada molécula de DNA de várias espécies, é o alinhamento dessas sequências. O alinhamento da sequência consiste no cálculo do melhor *match* para todas as bases de duas sequências que se estejam a comparar, isto é, procura a homologia, a identidade, entre as sequências.

Todos os alinhamentos começam pela comparação de duas sequências entre si. O algoritmo mais usado é o de programação dinâmica de alinhamento global de Needleman-Wunsch (Needleman e Wunsch, 1970). Existem vários alinhamentos possíveis e o que vai ser escolhido será o que tiver melhor *score*. O *score* de um alinhamento vai ser determinado de acordo com o número de substituições ou de delecções/inserções (*gaps*). De acordo com o algoritmo, é construída uma matriz de N x M (tamanho das sequências) e as distâncias entre cada par de base possível da sequência são determinadas do seguinte modo: 1 se houver *match*; 0 se houver *mismatch*; e um valor de penalização subtraído por cada *gap* inserida. Depois de calculadas todas as distâncias entre cada par de bases das duas sequências, vão sendo iterativamente seleccionadas as distâncias que permitem obter o melhor *score*.

A ideia de penalização para a inserção de um *gap* reflecte o facto biológico de uma delecção/inserção de bases em não múltiplos de três numa região codificante levar a *frameshift* de toda a proteína a jusante desse local. Se essa delecção/inserção ocorrer perto do início da proteína, esta será extremamente diferente, podendo haver mesmo um sinal para terminação levando a uma proteína muito mais curta; ambos os casos seriam, provavelmente, incompatíveis com a vida. Assim sendo, é muito mais tolerável uma substituição na região codificante, que leva a mudança de um só aminoácido na proteína, ou nem leva a nenhuma alteração devido à redundância do código genético.

O algoritmo de Needleman-Wunsch é implementado no programa de alinhamento mais usado pela comunidade científica, o CLUSTAL W (Thompson et al., 1994), acessível no URL http://www.ebi.ac.uk/Tools/clustalw/. Este programa impõe mais restrições às *gaps*, penalizando não só a abertura de uma *gap* como a sua extensão. Por ser biologicamente mais provável que aconteça uma delecção simultânea de várias bases do que várias delecções de uma só base numa pequena região da molécula é atribuída uma maior penalização aos alinhamentos com várias *gaps* próximas do que aos alinhamentos com várias *gaps* contínuas nessa região.

O CLUSTAL W permite ainda o alinhamento múltiplo, efectuado em três etapas: (1) cálculo da matriz de distância, indicando a divergência entre cada par de sequências alinhadas separadamente; (2) construção de uma árvore a partir da matriz de distância; (3) alinhamento progressivo das sequências de acordo com a ordem dos ramos da árvore.

Resumidamente, a medida de distância para cada par de alinhamento é determina como a divisão entre o número de bases onde ocorreu substituição pelo número total de bases excluindo os *gaps*. A matriz de distância é usada para construir uma árvore sem raiz pelo método Neighbour-Joining (Satou e Nei, 1987). A raiz é depois posicionada, pelo método "mid-point", numa posição onde as médias da extensão dos ramos em qualquer lado da raiz sejam iguais. Esta árvore é então usada para determinar uma nova medida que permite "pesar" a combinação dos alinhamentos desde as extremidades da árvore até à raiz. Deste modo, a relação biológica entre as espécies contribui para o resultado final do alinhamento múltiplo.

Geralmente, os programas de alinhamento mostram as sequências lado a lado, evidenciando as identidades e diferenças. Estas sequências alinhadas podem ser posteriormente importadas para outros programas que estimam vários parâmetros de diversidade genética, como o DnaSP (Rozas et al., 2003) e BioEdit (Hall, 1999).

Uma das limitações do CLUSTAL W para aplicação ao mtDNA é não considerar circularidade da molécula. Por exemplo, se a base 1 do mtDNA na espécie X corresponder ao tRNA-Pro, enquanto na espécie Y o tRNA-Pro começa na base 200, os 199 nucleótidos iniciais na espécie Y não serão alinhados, o mesmo se passando para os 199 nucleótidos finais da espécie X. Esta situação poderia ser resolvida por um passo inicial de colocar ambas as sequências a começar na mesma região genómica, o que actualmente tem que ser feito manualmente.

Muitos outros melhoramentos poderiam ser efectuados aos programas de alinhamento, tendo em conta alguns parâmetros biológicos como a heterogeneidade do tipo de mutação. É conhecido que as transversões são muito mais raras que as transições (Belle et al., 2005), podendo tal facto ser implementado nas penalizações para escolha entre alinhamentos alternativos.

1.6 - Objectivos:

Desenvolver uma ferramenta informática, de utilização simples, que permita a selecção de porções específicas da molécula de mtDNA. Pretende-se que esta ferramenta tenha em conta os seguintes critérios: (1) *input* de ficheiros do GenBank; (2) importe simultâneo de vários ficheiros; (3) dados de *output* compatíveis com os programas mais usados actualmente para análise de diversidade genética (em formato FASTA); (4) compatível com a plataforma Windows.

Testar a funcionalidade da ferramenta para selecção de genes de mtDNA em sequências de Primatas publicadas na base de dados RefSeq. Testar os ficheiros *output* como ficheiros de *input* nos programas BioEdit e DnaSP. Estimar algumas medidas de diversidade genética para as diferentes regiões do mtDNA em Primatas.

Desenvolver uma ferramenta informática, de utilização simples, que permita o alinhamento da molécula de mtDNA, quer na sua totalidade ou apenas para um gene específico, testando um novo algoritmo de procura de blocos de bases iguais nas duas sequências, a partir do maior tamanho até ao menor.

2 - Material e Métodos

2.1 - Implementação do programa de selecção de genes do mtDNA

O programa selecciona as posições de todos os genes/regiões mitocondriais presentes em mamíferos: região controlo, tRNA-Phe, 12s ribosomal RNA, tRNA-Val, 16s ribosomal RNA, tRNA-Leu (UUR), ND1, tRNA-Ile, tRNA-Gln, tRNA-Met, ND2, tRNA-Trp, tRNA-Ala, tRNA-Asn, tRNA-Cys, tRNA-Tyr, COX1, tRNA-Ser (UCN), tRNA-Asp, COX2, tRNA-Lys, ATP8, ATP6, COX3, tRNA-Gly, ND3, tRNA-Arg, ND4L, ND4, tRNA-His, tRNA-Ser (AGY), tRNA-Leu (CUN), ND5, ND6, tRNA-Glu, CYTB, tRNA-Thr e tRNA-Pro. Foi desenvolvido em C++, usando o ambiente de desenvolvimento integrado (IDE) C++ Builder e as suas bibliotecas pré-definidas para a criação de interfaces gráficas.

O funcionamento básico do programa consiste em seleccionar um determinado gene/região do mtDNA de vários ficheiros (ficheiro de *input* – exemplo no Anexo 1) e guardar a informação pretendida num outro ficheiro (ficheiro de *output* – exemplo no Anexo 2).

Os ficheiros são seleccionados numa "janela Abrir" típica e o seu nome é guardado em memória. Na interface do utilizador apenas aparecerá o nome do ficheiro, mas em memória será guardado o seu caminho completo de forma a aceder ao seu conteúdo numa fase posterior, evitando manter em memória desnecessariamente a informação pretendida. Por isso, apenas quando o utilizador seleccionar o botão "Run", os ficheiros serão lidos e analisados.

O formato dos ficheiros de *input* aceite pelo programa é o GenBank (Anexo 1) que contém a informação da localização dos genes na sequência, podendo por isso ter extensão de GenBank (.gbk, .gen, .gb, .gnk) ou de ficheiro de texto (.txt). No entanto, a informação será guardada no ficheiro de *output* num formato conhecido como FASTA. Este é um formato standard de ficheiro contendo uma primeira linha iniciada pelo carácter '>' e que tem o nome da sequência, seguindo-se várias linhas, geralmente com 70 caracteres, que representam as bases da sequência. O ficheiro pode conter várias sequências, sendo que todas estão separadas pela linha que inicia pelo carácter '>'. O ficheiro de *output* terá sempre extensão de ficheiro de texto (.txt) e a primeira linha terá o *Accession Number* e o gene seleccionado.

A procura da localização de um determinado gene num ficheiro de *input* passa pelo processo de *matching* de determinadas *strings*. Por exemplo, a localização da proteína ND1 está na linha imediatamente anterior à que contém o conteúdo

`/gene="ND1"`

onde terá a localização de início e fim com o seguinte formato:

`3308..4264`

A localização também pode ser precedida pela palavra “complement”.

Uma particularidade ocorre para os genes dos tRNA dos aminoácidos leucina e serina, existindo dois tRNAs para cada um destes aminoácidos, e logo duas linhas com informação `/product="tRNA-Leu"` e `/product="tRNA-Ser"`, respectivamente. Esta ambiguidade é resolvida com informação da linha seguinte que indica o codão que cada uma reconhece `/note="codons recognized: X"`, onde X pode ser UUR ou CUN para a leucina e UCN ou AGY para a serina. Os RNA ribossómicos também podem ter mais de uma forma de serem identificados ("12S ribosomal RNA", "s-rRNA", "12S rRNA", "16S ribosomal RNA", "l-rRNA" e "16S rRNA").

Outro caso especial ocorre para a região de controlo, que para além de ter várias formas de ser identificada, pode apresentar uma localização caracterizada por quatro números da seguinte forma: `join(1..579,16029..16571).` Esta identificação é incorporada numa só região, respeitando-se a circularidade da molécula.

O programa é compatível com as plataformas Windows. A linguagem usada na interface gráfica é o inglês para que o programa possa ser futuramente divulgado pela comunidade científica internacional.

O código fonte do programa é apresentado no Anexo 3.

2.2 - Base de dados de sequências de mtDNA de Primatas e medidas de diversidade genética

O programa foi testado em todas as sequências de Primatas depositadas na base de dados RefSeq, até ao dia 12 de Julho de 2007, (Tabela 2.1).

Na base de dados está ainda disponibilizada uma árvore filogenética que representa a proximidade biológica entre estas espécies de Primatas. Esta árvore está representada na Figura 2.1, tendo-se usado o programa Treeview (http://taxonomy.zoology.gla.ac.uk/rod/rod.html), para representação gráfica dos dados extraídos.

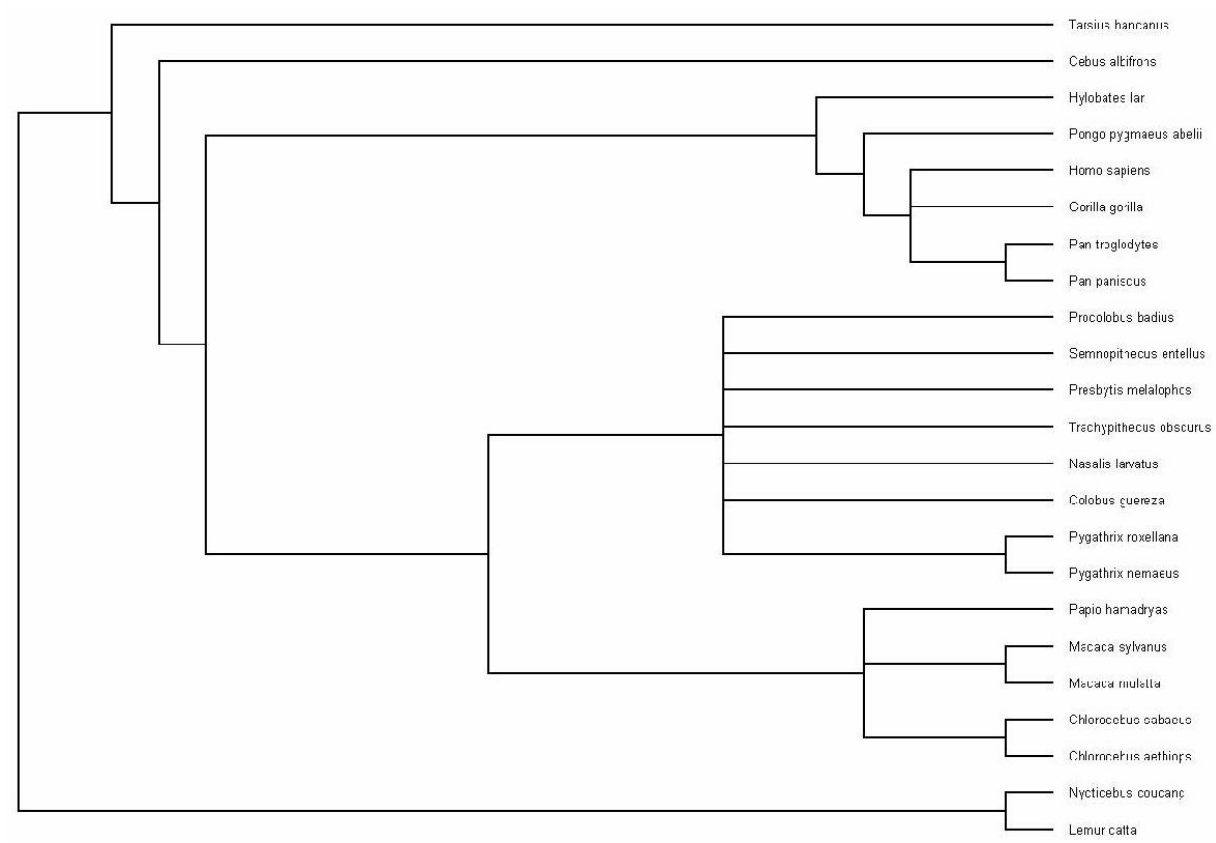


Figura 2.1 - Árvore filogenética (cladograma) para as espécies de Primatas estudadas.

Tabela 2.1 - Sequências completas de mtDNA da ordem Primatas depositadas na RefSeq.

Accession Number	Espécie	Classificação taxonómica (subordem; parvordem; família)	Tamanho do genoma (bp)
NC_001643	*Pan troglodytes*	Haplorrhini; Catarrhini; Hominidae	16554
NC_001644	*Pan paniscus*	Haplorrhini; Catarrhini; Hominidae	16563
NC_001645	*Gorilla gorilla*	Haplorrhini; Catarrhini; Hominidae	16364
NC_001646	*Pongo pygmaeus*	Haplorrhini; Catarrhini; Hominidae	16389
NC_001807	*Homo sapiens*	Haplorrhini; Catarrhini; Hominidae	16571
NC_001992	*Papio hamadryas*	Haplorrhini; Catarrhini; Cercopithecidae; Cercopithecinae	16521
NC_002082	*Hylobates lar*	Haplorrhini; Catarrhini; Hylobatidae	16472
NC_002763	*Cebus albifrons*	Haplorrhini; Platyrrhini; Cebidae; Cebinae	16554
NC_002764	*Macaca sylvanus*	Haplorrhini; Catarrhini; Cercopithecidae; Cercopithecinae	16586
NC_002765	*Nycticebus coucang*	Strepsirrhini; Chiromyiformes; Lorisidae	16764
NC_002811	*Tarsius bancanus*	Haplorrhini; Tarsiiformes; Tarsiidae	16927
NC_004025	*Lemur catta*	Strepsirrhini; Lemuriformes; Lemuridae	17036
NC_005943	*Macaca mulatta*	Haplorrhini; Catarrhini; Cercopithecidae; Cercopithecinae	16564
NC_006900	*Trachypithecus obscurus*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16560
NC_006901	*Colobus guereza*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16648
NC_007009	*Chlorocebus aethiops*	Haplorrhini; Catarrhini; Cercopithecidae; Cercopithecinae	16389
NC_008066	*Chlorocebus sabaeus*	Haplorrhini; Catarrhini; Cercopithecidae; Cercopithecinae	16550
NC_008215	*Semnopithecus entellus*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16532
NC_008216	*Nasalis larvatus*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16570
NC_008217	*Presbytis melalophos*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16543
NC_008218	*Pygathrix roxellana*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16549
NC_008219	*Procolobus badius*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	16595
NC_008220	*Pygathrix nemaeus*	Haplorrhini; Catarrhini; Cercopithecidae; Colobinae	15467

Como medidas de diversidade genética usaram-se (Nei, 1987):

- diversidade nucleotídica: probabilidade de dois nucleótidos homólogos escolhidos ao acaso serem diferentes:

$$\hat{\pi}_n = \frac{\sum_{i=1}^{k}\sum_{j<i} p_i p_j \hat{d}_{ij}}{L}$$

em que L é o número total de nucleótidos, $\hat{d}_{ij}$ é uma estimativa do número de mutações ocorridas desde a divergência das sequências i e j, k é o número de sequências e pi é a frequência da sequência i.

- diversidade haplotípica: probabilidade de duas sequências (haplótipos) escolhidas ao acaso serem diferentes:

$$\hat{H} = \frac{n}{n-1}(1-\sum_{i=1}^{k} p_i^2)$$

em que n é o número de sequências estudadas.

Estas medidas foram calculadas no programa DnaSP, bem como o cálculo do conteúdo G+C. O DnaSP implica alinhamento prévio das sequências noutro programa. O alinhamento dos genes/regiões do mtDNA dos Primatas, seleccionados através da ferramenta de selecção de genes do mtDNA, foi efectuado no programa BioEdit, usando o algoritmo nele implementado, que é o do CLUSTAL W.

2.3 - Algoritmo do programa de alinhamento de genes do mtDNA

```
/* Algoritmo Recursive_Find encontra todos os blocos de tamanho
* máximo m, entre duas strings (A e B) limitadas nos índices
* indicados para cada string: ai e af para A, bi e bf para B
*/
Algoritmo Recursive_Find(A, B, ai, bi, af, bf, m):
  Input: as strings, A e B; os índices limitadores das strings
         ai e af, bi e bf; tamanho bloco máximo, m;
  Output: 2 vectores bidimensionais (Ar e Br) para cada string
         contendo as posições iniciais e finais dos blocos
         comuns
```

```
  se m<1 ou af-ai<1 ou bf-bi<1 então termina;

  (Pa, Pb, mComum) ← Find(A, B, ai, bi, af, bf, m);

  se mComum<1 então termina;

  Ar[num_bloco][0] ← Pa;
  Ar[num_bloco][1] ← Pb;
  Br[num_bloco][0] ← Pa+mComum;
  Br[num_bloco++][1] ← Pb+mComum;

  Recursive_Find(ai,bi,Pa,Pb,mComum-1);
  Recursive_Find(Pa+mComum,Pb+mComum,af,bf,mComum);


/*
* Algoritmo Find encontra a maior substring, até um máximo de m,
* comum entre duas strings (A e B) começando e terminando nos
* índices indicados para cada string: ai e af para A, bi e bf
* para B
*/

Algoritmo Find(A, B, ai, bi, af, bf, m):
  Input: as strings A e B; os índices limitadores das strings ai
         e af, bi e bf; tamanho máximo do bloco, m
  Ouput: Pa e Pb: índice nas strings A e B onde inicia o maior
         bloco comum; mComum: tamanho bloco máximo entre as duas
         strings

  m ← min(m, af-ai, bf-bi)
  para mi ← m até 1:
    para i←ai até af-mi fazer:
      para j←bi até bf-mi fazer:
        para k←0 até mi fazer:
          se A[i+k]!=B[j+k] então
            flag ← 0;
            break;
        se flag=1 então
           Pa ← ai+i
           Pb ← bi+j
           mComum ← mi
           return (Pa,Pb,mComum)
```

2.4 - Implementação do programa de alinhamento de genes do mtDNA

O programa de alinhamento de mtDNA procura blocos de bases comuns a duas sequências, desde o maior tamanho até 1, indicando as suas localizações e o número de posições que permaneceram por alinhar. Foi desenvolvido em C++, usando o ambiente de

desenvolvimento integrado (IDE) C++ Builder e as suas bibliotecas pré-definidas para a criação de interfaces gráficas.

O programa permite não só fazer o alinhamento de sequências completas de mtDNA, mas também de genes/regiões de mtDNA seleccionados pelo utilizador. O método para a selecção de genes é o mesmo descrito para o programa de selecção de genes (ponto 1), sendo neste caso necessário um ficheiro de *input* com o formato GenBank (Anexo 1). Quando se pretende fazer o alinhamento de sequências completas, o programa lê ficheiros com o formato FASTA ou com o formato GenBank. Devem ser lidos dois ficheiros, pertencentes a 2 indivíduos da mesma ou de diferentes espécies. O programa impede que o segundo ficheiro ("File 2") seja lido se ainda não tiver sido lido um primeiro ficheiro ou se as opções de extracção ("Range") tiverem sido alteradas entretanto.

Existe uma diferença importante ente o alinhamento de sequências inteiras e o de determinado gene. Com sequências inteiras, e por se tratar de um genoma circular, depois de encontrado o primeiro maior bloco, as sequências são reorganizadas de forma a "começarem" por esse bloco, prevenindo que nenhum fragmento fique por alinhar devido a desfasamentos de numeração entre as duas sequências. Depois deste primeiro alinhamento as sequências são tratadas de acordo com o algoritmo descrito a seguir começando na posição ainda não alinhada. Para o alinhamento de genes, a molécula é considerada linear, não se efectuando aquele passo.

De forma a diminuir o tempo de execução do programa foram introduzidos máximos de tamanho para os blocos a procurar. No caso de sequências inteiras tem que se indicar o tamanho máximo do bloco a procurar para reorganizar o início da molécula em "First Aligment" (por default aparece 100 na caixa). Para a procura dos restantes blocos no caso da sequência completa e todos os blocos no caso de um gene, tem que se introduzir o tamanho máximo no "Maximal Length" (por default aparece 40 na caixa). Os blocos com tamanho superior aos indicados no "First Aligment" e "Maximal Length" são detectados de forma separada, mas na fase final são unidos e apresentados como um único bloco.

Existem dois modos de funcionamento: automático e manual. O modo automático localiza os blocos comum às sequências ou genes lidos (*S1* e *S2*) através da força bruta, da seguinte forma:

1. Encontra bloco de maior tamanho, m, até ao máximo de "Maximal Length", M ($m \leq M$) comum a *S1* e *S2* que inicia em A1 e A2, respectivamente.

2. Para cada Sx, $x \in \{1,2\}$:

2.1 Repete passo 1 para a sequência $Sx(0,Ax)$ com $M=m-1$;

2.2 Repete passo 1 para a sequência *Sx(Ax+m, tamanho(Sx))* com *M=m*;

O modo manual funciona da mesma maneira, mas a procura é limitada às posições indicadas pelos utilizadores pelos campos "From…to". Os números das posições devem estar correctas de acordo com a numeração dos genes indicados no ficheiro GenBank.

O *output* (Anexo 4) indica o número de bases não alinhadas (*a* para a sequência 1 e *b* para a sequência 2) e a lista com a localização dos blocos comuns às sequências em que o primeiro número indica a posição inicial do bloco no primeiro ficheiro, o segundo número a posição inicial do mesmo bloco no segundo ficheiro e o terceiro número indica o tamanho do bloco. Esta lista apresenta-se ordenada pela posição inicial do bloco no primeiro ficheiro. A informação do *output* aparece não só na interface gráfica, mas também é incluída num ficheiro chamado "alignment.txt". Toda a numeração corresponde à indicada nos ficheiros GenBank analisados.

O código fonte do programa é apresentado no Anexo 5.

3 - Resultados

3.1 - Aplicação do programa de selecção de genes mtDNA aos Primatas

Na figura 3.1 está representada a interface gráfica do programa de selecção de genes de mtDNA. As sequências importadas para análise (ficheiros GenBank em formato .txt) são incluídas na janela da direita. A janela da esquerda permite a selecção do gene/região de uma lista.

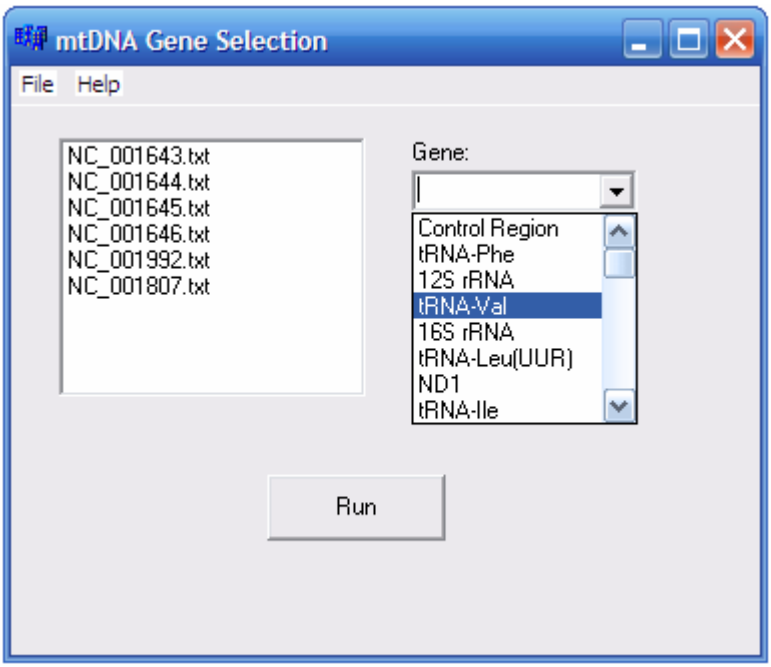


Figura 3.1 - Interface gráfica do programa de selecção de genes de mtDNA.

Este programa foi usado para extrair todas as 38 regiões do mtDNA dos 23 Primatas referidos no Material e Métodos.

Curiosamente, apesar de os ficheiros GenBank dos Primatas terem sido retirados da base RefSeq, uma base revista e com controlo de validação dos dados e de formatação, foram encontradas várias inconsistências.

Como exemplo de algumas inconsistências de formatação, pode referir-se que algumas das localizações dos genes tinham o sinal “>” a anteceder a numeração dos nucleótidos, o que impedia a sua selecção. Foi necessário incluir esta particularidade no código fonte do programa. É de esperar que outras inconsistências de formatação possam

existir noutros ficheiros GenBank, principalmente nos depositados na base geral e não na RefSeq. Nesse caso, o operador terá que alterar manualmente o ficheiro GenBank para o formato standard de modo a que o programa funcione.

Na espécie *Chlorocebus sabaeus*, a região controlo não estava indicada na totalidade, apenas se referindo a localização de duas sub-zonas (denominadas regiões hipervariáveis). Foi necessário incluir no ficheiro GenBank a linha

```
D-loop          join(1..582,16010..16550)
```

para que o programa extraísse toda a informação da região controlo.

Foram também detectadas várias inconsistências biológicas. A sequência de mtDNA reportada para a espécie *Pygathrix nemaeus* não possui região controlo. Optou-se por excluir esta espécie de todas as análises efectuadas.

Os *outputs* criados para os 38 genes dos 22 Primatas foram testados no programa BioEdit. Não se notaram incompatibilidades do formato destes ficheiros com o BioEdit, tendo-se efectuado em seguida o alinhamento dos genes, usando o algoritmo de alinhamento incorporado neste programa (o mesmo do CLUSTAL W).

Para testar a qualidade dos alinhamentos efectuados no BioEdit para os genes já isolados, foram comparados os alinhamentos dos 22 tRNAs contra os alinhamentos dos tRNAs descritos na base de dados Mamit-tRNA (Helm et al. 2000). A comparação evidenciou várias inconsistências de início e fim dos tRNAs dos Primatas depositados na base de dados RefSeq com as do Mamit-tRNA.

Também no alinhamento dos 13 genes codificantes de proteínas se detectaram inconsistências na sua terminação. Todas as alterações efectuadas aos ficheiros GenBank estão indicadas pormenorizadamente na Tabela 3.1.

Tabela 3.1 - Alterações efectuadas aos ficheiros GenBank.

Accession number	Gene	Alterado para	Indicado no ficheiro
NC_001643	tRNA-Ser (UCN)	6863-6931	6862-6933
NC_001644	tRNA-Ser (UCN)	6864-6932	6863-6934
NC_001645	tRNA-Ser (UCN)	6867-6935	6866-6937
	tRNA-Thr	15310-15375	15310-15377
NC_001646	tRNA-Ser (UCN)	6874-6942	6873-6944
	tRNA-Pro	15404-15472	15403-15472
	COX1	5332-6873	5332-6871
NC_001807	tRNA-Ser (UCN)	7447-7515	7446-7517
	COX3	9208-9991	9208-9988
	CYTB	15699-15784	15699-15778
	tRNA-Pro	15957-16024	15956-16024
NC_001992	tRNA-Ser (UCN)	6877-6945	6878-6945
	ATP6	7951-8631	7951-8630
	tRNA-Thr	15313-15376	15312-15376
	COX1	5336-6876	5336-6877
NC_002082	tRNA-Trp	4928-4996	4929-4996
NC_002763	tRNA-Val	1029-1096	1029-1086
	16s rRNA	1097-2651	1087-2651
NC_002764	tRNA-Tyr	5245-5310	5246-5327
NC_002765	ND4	10219-11596	10219-11602
	tRNA-His	11597-11665	11600-11665
NC_002811	tRNA-Gly	2423-9492	9422-9493
NC_004025	tRNA-Leu (UUR)	2666-2739	2665-2739
	tRNA-Arg	9827-9894	9827-9893
	ND6	13584-14102	13584-14099
NC_005943	tRNA-Cys	5712-5774	5712-5774
	tRNA-Tyr	5774-5845	5775-5845
	tRNA-Ser (UCN)	7391-7459	7390-7461
	tRNA-Pro	15947-16014	15946-16014
NC_006900	tRNA-Tyr	5251-5317	5253-5317
	tRNA-Glu	14119-14187	14118-14187
NC_006901	tRNA-Tyr	5257-5322	5258-5322
NC_007009	tRNA-Tyr	5249-5314	5250-5314
	tRNA-Leu (CUN)	11689-11760	11689-11754
NC_008066	tRNA-Tyr	5246-5311	5247-5311
	tRNA-Ser (UCN)	7440-7508	7439-7510
	tRNA-Pro	15942-16009	15941-16009
	COX1	5899-7439	5899-7437
	tRNA-Ile	4258-4326	4259-4326
	tRNA-Trp	5507-5572	5507-5569
	tRNA-Ala	5580-5649	5580-5648
	tRNA-Cys	5754-5815	5754-5815
NC_008215	tRNA-Tyr	5246-5311	5247-5311
	CYTB	14181-15321	14181-15315
NC_008216	CYTB	14193-15333	14193-15327
	tRNA-Tyr	5253-5318	5254-5318
NC_008217	tRNA-Tyr	5245-5310	5246-5326
	tRNA-Glu	14114-14181	14114-14182
	CYTB	14187-15327	14187-15321
	tRNA-Glu	14113-14182	14114-14182
NC_008218	tRNA-Tyr	5256-5321	5257-5342
	CYTB	14190-15330	14190-15324
NC_008219	tRNA-Tyr	5250-5315	5251-5315
	CYTB	14190-15330	14190-15324

3.2 - Comparação de parâmetros de diversidade genética entre os genes mtDNA dos Primatas

Existe uma elevada diversidade, quando medida em termos de diversidade nucleotídica, para os diversos genes do mtDNA nos Primatas (Figura 3.2), com alguma heterogeneidade.

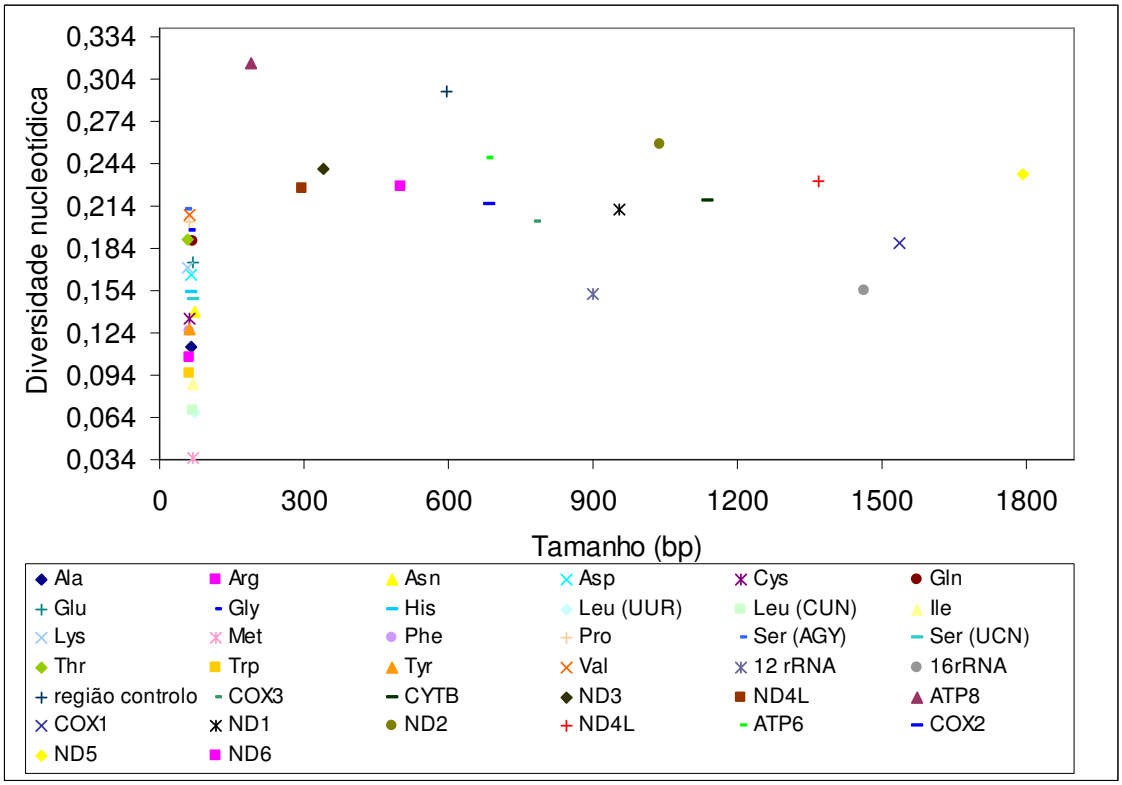


Figura 3.2 – Dispersão da diversidade nucleotídica em função do tamanho excluindo posições com *gaps* (em pares de bases) dos diversos genes do mtDNA.

Os tRNAs, que apresentam um comprimento semelhante, variam bastante quanto à diversidade nucleotídica, sendo o tRNA-Met o que apresenta menor diversidade; por oposição, o que apresenta maior diversidade é o tRNA-Ser(AGY).

O tRNA-Met, o mais conservado nos Primatas, é o tRNA que vai ser reconhecido para o início da tradução da proteína, desempenhando uma função essencial à vida. Esta elevada conservação do tRNA-Met é extensível a todos os mamíferos (Helm et al., 2000).

Na população humana, o tRNA que apresenta maior diversidade é o tRNA-Thr, atingindo 7 vezes mais mutações que os restantes tRNAs (Kivisild et al., 2006). Globalmente nos Primatas, o tRNA-Thr é um dos que apresenta uma elevada taxa de diversidade, estando em quinto lugar.

Ambos os genes para os RNAs ribossómicos (rRNA), de tamanho semelhante aos genes codificantes de proteínas, apresentam menor diversidade que estes. Os rRNAs apresentam uma estrutura tridimensional bastante complexa, com ansas e hélices, que poderá ser incompatível em termos funcionais com o aparecimento de diversidade genética em muitas das suas regiões.

O menor gene codificante de proteínas, o ATP8, é o que apresenta uma maior diversidade nucleotídica, mesmo considerando todas as regiões da molécula. O facto de ser muito diverso tinha sido já observado para entre espécies mais divergentes que os Primatas (Broughton e Reneau, 2006). Curiosamente, o ATP8 quase que não varia dentro da população humana (Mishmar et al., 2003). Por oposição, tem sido referido que o ATP6 é bastante conservado entre espécies muito distantes (Mishmar et al., 2003), mas tal não foi verificado para estas 22 espécies de Primatas, sendo o terceiro gene codificante de proteínas com maior diversidade nucleotídica. Mishmar e colaboradores (2003) já tinham notado uma elevada diversidade nucleotídica para o ATP6 na população humana e esta característica, face aos resultados aqui descritos, parece ser partilhada pelos restantes Primatas.

É sabido que o conteúdo em bases G e C, denominado G+C, é importante para a estabilização da mólecula de DNA e diminuição da propensão para mutação. De facto, o número de ligações de pontes de hidrogénio entre as bases complementares G-C é igual a três, enquanto que a ligar as bases A-T existem duas pontes de hidrogénio, sendo necessária mais energia para abrir as bases G-C do que as A-T. E dos genes codificantes das proteínas nos Primatas, o que apresenta menor conteúdo G+C é o ATP8 (Figura 3.3).

A segunda região com maior diversidade nucleotídica do mtDNA nos Primatas é a região controlo. De notar que dadas as menores contrições biológicas desta região, o nível de homologia é menor para as 22 espécies de Primatas, tornando o alinhamento mais complicado, com a inclusão de muitas *gaps*. As posições com *gaps* não foram contabilizadas para o tamanho da região, diminuindo este de aproximadamente 1200bp para 597bp. Esta é a região que apresenta maior diversidade nucleotídica dentro da população humana.

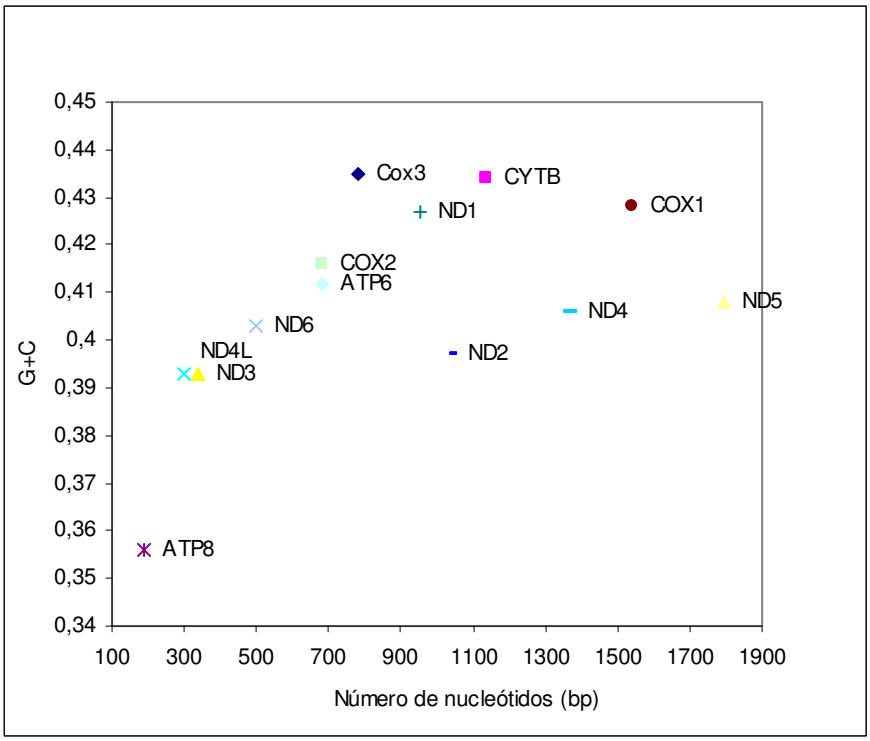


Figura 3.3 – Conteúdo G+C (em %) versus o número de nucleótidos (bp) nos 13 genes codificantes de proteínas.

As sequências dos 22 Primatas são todas diferentes nas várias regiões, atingindo uma diversidade haplotípica igual a 1 (Figura 3.4), com a excepção de para a maioria dos tRNAs (16 dos 22).

De facto, o tRNA-Met é o que apresenta menor capacidade distintiva, seguindo-se, por ordem crescente, os tRNA-Phe, tRNA-Ala, tRNA-Arg, tRNA-Leu(CUN), tRNA-Glu, tRNA-Ser(UCN), tRNA-Val, tRNA-Asn, tRNA-His, tRNA-Leu(UUR), tRNA-Ile, tRNA-Lys, tRNA-Ser(AGY), tRNA-Trp e tRNA-Tyr.

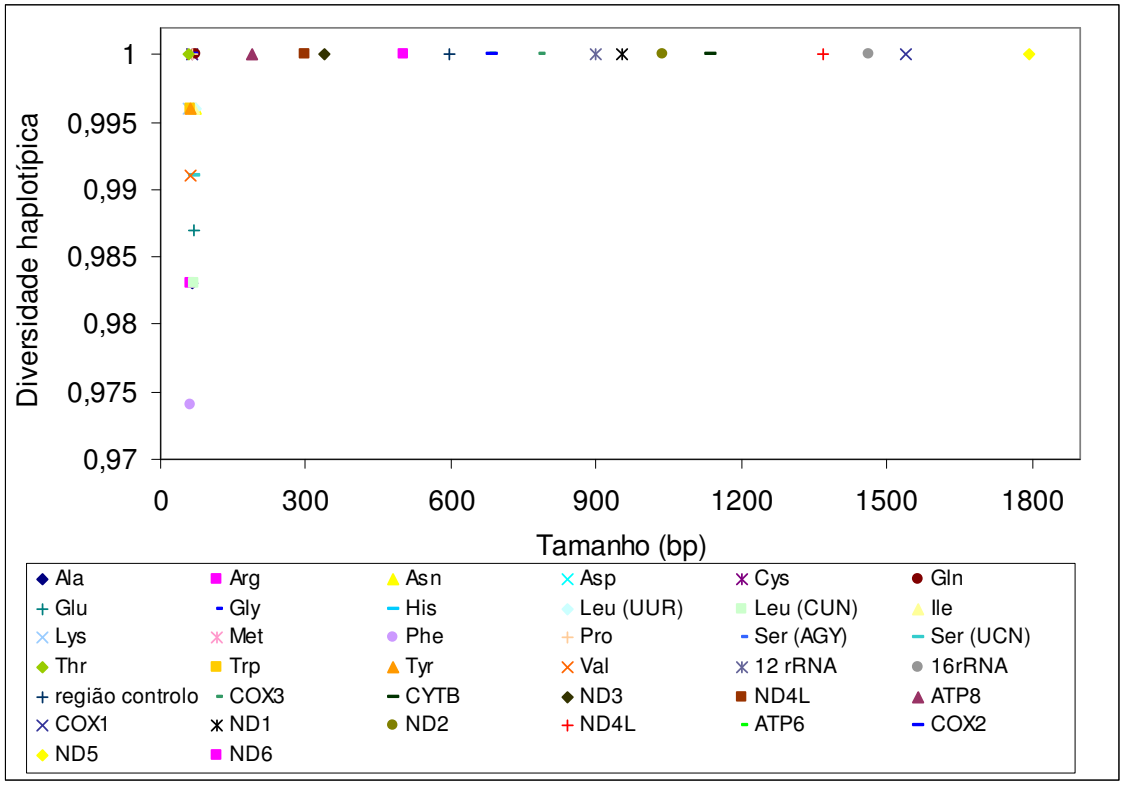


Figura 3.4 – Dispersão da diversidade haplotípica em função do tamanho excluindo posições com *gaps* (em pares de bases) dos diversos genes do mtDNA.

3.3 - Aplicação do programa de alinhamento de genes mtDNA aos Primatas

Para o teste do alinhamento com o algoritmo de procura de blocos comuns sequencialmente menores, efectuou-se o alinhamento de dois pares de sequências completas: um de espécies próximas, *Homo sapiens* e *Pan troglodytes*; outro de espécies afastadas, *Homo sapiens* e *Macaca mulatta*.

Na Figura 3.5 está representada a interface gráfica do programa. Nos botões da esquerda é possível: (1) escolher entre o "Range" do fragmento a analisar, podendo ser a sequência na sua totalidade ("Intire Sequence") ou um gene/região que pode ser seleccionado de uma lista; (2) carregar os dois ficheiros ("File 1" e "File2"); (3) escolher os máximos dos tamanhos dos blocos a procurar em "First Alignment" e "Maximal Length"; (4) escolher entre o modo "Automatic" e "Manual", tendo que neste se indicar a região a analisar. No Anexo 4 está depositado o ficheiro de resultados para o alinhamento de *Homo sapiens* e *Pan troglodytes*, que apareceria na janela central.

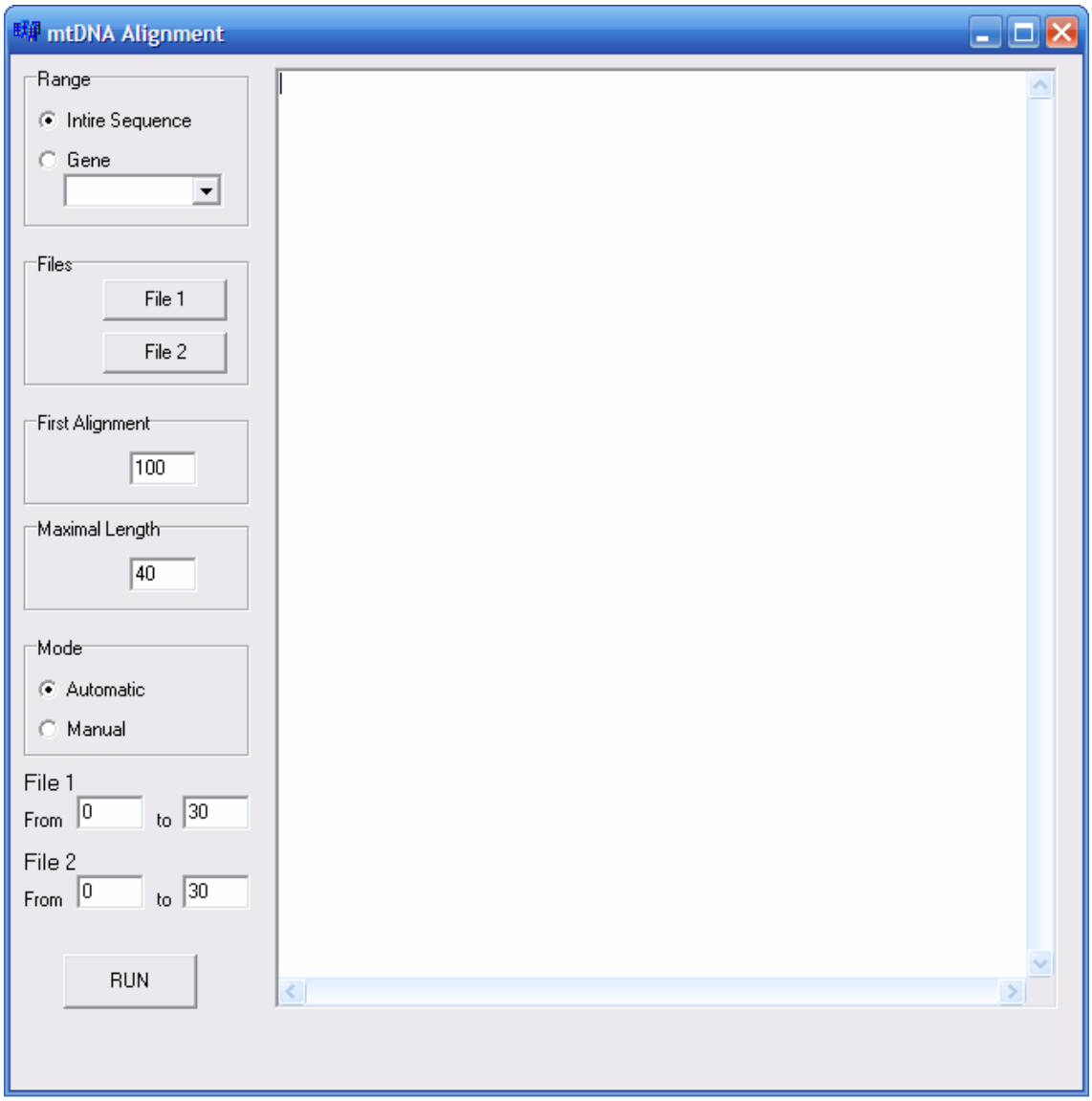


Figura 3.5 - Interface gráfica do programa de alinhamento de genes de mtDNA.

Para comparação, efectuou-se o alinhamento dos mesmos dois pares de sequências na ferramenta interactiva CLUSTAL W.

Quanto a tempos de computação da nova ferramenta, os testes efectuados estão resumidos na tabela 3.2. Estes foram realizados num computador Intel Core 2 a 2.13GHz e 2GB de RAM.

Como se pode comprovar, os tempos estão muito dependentes do nível de homologia entre as sequências a comparar. Na comparação *Homo-Pan*, o maior bloco tem um tamanho de 167bp, pelo que a escolha de 100bp para o "First Alignment" é eficiente e, na procura dos blocos sequenciais, "Maximal Length", a opção de 60bp é mais eficiente do que 40bp e 20bp,

porque ainda restam blocos com tamanho considerável (149bp, 119bp, 103bp, 88bp, 78bp, 67bp, 62bp).

Tabela 3.2 - Tempos de computação da ferramenta de alinhamento.

			Homo - Pan		*Homo - Macaca*	
		bp	Tempo parcial	Tempo total	Tempo parcial	Tempo total
Teste 1	First alignment	100	4’11’’	1h49’21’’	2h4’48’’	2h35’37’’
	Maximal length	20	1h45’10’’		30’49’’	
Teste 2	First alignment	100	4’11’’	49’54’’	2h4’48’’	2h37’10’’
	Maximal length	40	45’43’’		32’22’’	
Teste 3	First alignment	100	4’11’’	37’23’’	2h4’48’’	4h46’19’’
	Maximal length	60	33’12’’		2h41’31’’	

Para a comparação do *Homo-Macaca*, os mesmos testes foram muito menos eficientes, especialmente os 100bp como máximo para o “First Alignment”, o que aumentou 30 vezes a extensão deste passo. De facto, o maior bloco comum entre estas espécies tem 65bp, e o programa terá que procura o maior bloco começando com 100, seguindo para 99 e assim sucessivamente. Os tempos podem ser melhorados para alinhamentos entre estas duas espécies com a diminuição do comprimentos dos blocos a procurar.

Em termos comparativos, os alinhamentos *Homo-Pan* e *Homo-Macaca* na ferramenta online do CLUSTAL W demoram aproximadamente 2 minutos cada.

O programa de alinhamento calcula os valores dos números de bases não alinhadas nas duas sequências. Para a comparação *Homo-Pan*, esses valores foram, 1470 e 1453, respectivamente. Estes valores são ligeiramente inferiores aos números de bases não alinhadas pelo CLUSTAL W, de 1477 e 1460.

Os valores para a comparação *Homo-Macaca* foram de 3867 e 3860 para este alinhamento e 3549 e 3542 para o do CLUSTAL W, respectivamente.

Quanto à eficiência do método de alinhamento, quando se compara o alinhamento dos blocos de tamanho superior a 10bp com o alinhamento resultante do CLUSTAL W apenas existe um tipo de discrepância mínima. Esta resulta de uma decisão arbitrária do CLUSTAL W de incluir uma *gap* na última posição de uma dada base quando repetida sequencialmente, enquanto que o novo algoritmo poderia levar a inclusão de *gap* no início se a última das bases repetidas contribuir para aumentar o tamanho do bloco seguinte. Um exemplo está representado na Figura 3.6.

```
NC_001807        CCCATCAACAACCGCTATGTATTTCGTACATTAC
NC_001643        TTCATTA-CAACCGCTATGTATTTCGTACATTAC
                   ***  *  **************************
```

Figura 3.6 - Alinhamento efectuado pelo programa de alinhamento representado em blocos coloridos sobreposta ao alinhamento efectuado pelo CLUSTAL W. As estrelas indicam identidade da base nas duas sequências no alinhamento CLUSTAL W.

Grandes diferenças são observadas, contudo, para alguns alinhamentos de blocos com menos de 10bp, se estes se localizarem em regiões com considerável diversidade genética. Não estabelecendo restrições *a priori*, o algoritmo procura os maiores blocos, levando à necessidade de inclusão de muitas *gaps*.

Por exemplo, no alinhamento da Figura 12, seria necessário incluir 14 *gaps*, ficando ainda uma substituição do tipo transversão C-A. Se em vez de procurar e valorizar a organização dos blocos de 4 e 2 bases (Figura 3.7A), se valorizar os blocos de 3 e 3 bases (Figura 3.7B), já só será necessário incluir 2 *gaps*, ficando todas as substituições do tipo transição.

Esta situação é claramente mais crítica para a região controlo do que para a zona codificante da molécula e para a comparação de espécies mais afastadas.

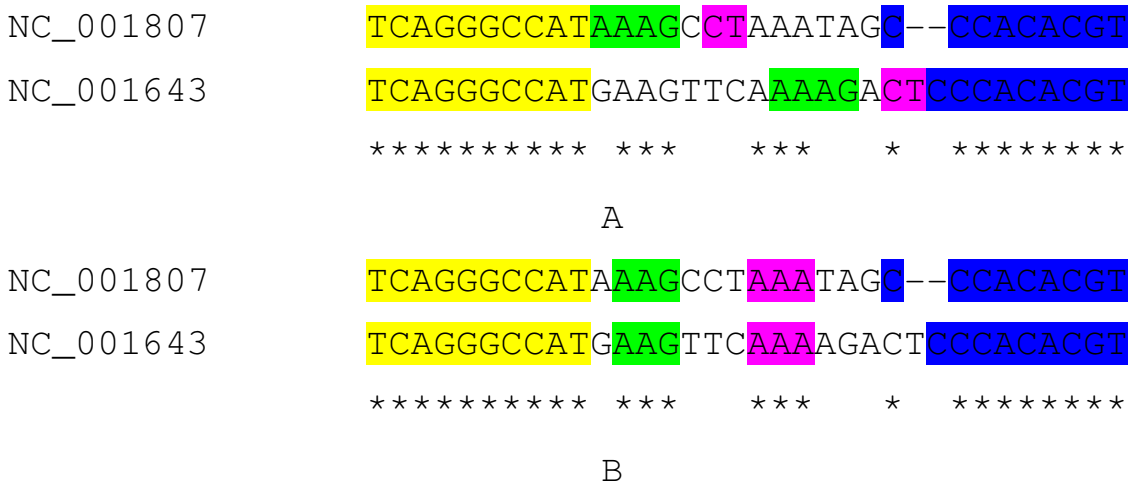


Figura 3.7 - A - Alinhamento efectuado pelo programa de alinhamento representado em blocos coloridos sobreposta ao alinhamento efectuado pelo CLUSTAL W. B- Alinhamento alternativo. As estrelas indicam identidade da base nas duas sequências no alinhamento CLUSTAL W.

4 - Conclusões

O programa de selecção de genes do mtDNA revelou-se útil para um estudo da heterogeneidade entre as diversas regiões do mtDNA presentes em mamíferos. Tem a funcionalidade de permitir a importação de qualquer ficheiro GenBank, e de selecção dos 38 genes/regiões de mtDNA existentes em mamíferos.

Por comparação à ferramenta interactiva descrita por Vasconcelos et al. (2005) que apresenta certas limitações, a ferramenta de selecção aqui desenvolvida tem algumas vantagens: (1) permite o importe de qualquer ficheiro, não estando limitados às espécies incorporadas pelos autores; (2) permite analisar, não só os 13 genes codificantes de proteínas, como também os genes que codificam os tRNA e os rRNA e a região de controlo; e (3) fornece acesso aos ficheiros das sequências dos genes isolados para que o utilizador possa usar independentemente nos programas de análise que entender.

De realçar que a aplicação desta ferramenta de selecção de genes do mtDNA ao estudo dos Primatas depositados na base RefSeq mostrou que, nesta fase do desenvolvimento do conhecimento científico, o operador não deve aceitar automaticamente os *outputs*. De facto, existem ainda muitas inconsistências nas bases de dados genéticas do GenBank, mesmo naquelas que foram desenhadas com critérios cuidadosos de anotação e revisão, como é o caso da RefSeq.

Comprovou-se que, estudando individualmente quase todas as regiões do mtDNA (excepto 16 dos 22 tRNAs), é possível distinguir as 22 espécies de Primatas. As diversidades haplotípica e nucleotídica são, em geral, bastante elevadas e constatou-se a elevada heterogeneidade entre as regiões.

Esta informação poderia ser usada, por exemplo, para seleccionar regiões pequenas, logo fáceis de analisar em laboratório e/ou em amostras difíceis (como pêlos ou fezes de alguns dos Primatas), mas que garantissem a máxima capacidade de distinção entre as espécies. O gene codificante da proteína ATP8 parece ser o mais indicado.

O ATP8 poderia também ser usado como relógio molecular rápido, no estudo filogenético de Primatas, enquanto um dos genes do RNA ribossómico funcionaria como relógio molecular lento.

Neste trabalho, estudou-se o comportamento de um algoritmo de alinhamento de pares de sequências sem restrições *a priori*, procurando apenas blocos de homologia

sequencialmente menores entre duas sequências. No extremo oposto, situam-se os algoritmos com restrições *a priori*, como o CLUSTAL W, o mais usado pela comunidade científica.

A comparação entre os resultados de alinhamento efectuados pelo algoritmo aqui estudado e os do CLUSTAL W mostrou que, para blocos de tamanho acima de 10bp, não há disparidades entre ambos os métodos. O que corresponderá a uma elevada percentagem da molécula de mtDNA em espécies próximas. Grandes diferenças são observadas, contudo, para alguns alinhamentos de blocos com menos de 10bp, se estes se localizarem em regiões com considerável diversidade genética. Será necessário implementar um alinhamento local para estes blocos com menos de 10bp. Neste alinhamento local terá que se atribuir *scores* às formas alternativas de organizar blocos de homologias, privilegiando a organização que minimizar o número de inclusões de *gaps*, e no caso de alternativas com o mesmo número de *gaps*, aquela que tiver menor número de *gaps* separadas, de forma semelhante ao incorporado à partida no CLUSTAL W.

O alinhamento por procura de blocos de tamanho sequencialmente menores, com algoritmo de força bruta, implica também um tempo de computação muito elevado, enquanto que o algoritmo do CLUSTAL W é um algoritmo muito eficiente. Mesmo estabelecendo valores máximos para o tamanho dos blocos a procurar, a eficiência está muito dependente dos valores reais de tamanho do maior bloco comum às duas sequências. O tempo de computação poderia ser ainda melhorado pela implementação de métodos mais eficientes de procura dos blocos, como por exemplo por programação dinâmica com uma matriz de N x M (tamanho das sequências), preenchida com valores 0 (para *mismatch*) e 1 (para *match*) e procurar a maior diagonal de 1s consecutivos. A eficiência resulta de numa só análise se saber os tamanhos de todos os blocos possíveis, podendo-se escolher o máximo (e respectiva localização) sem rever as sequências.

As operacionalidades de circularidade da molécula quando se alinha a sequência de mtDNA na sua totalidade e de alinhamento parcial de genes/regiões da molécula, ambas implementadas na ferramenta informática aqui desenvolvida são bastante práticas para estudos filogenéticos com base na diversidade de mtDNA. Não só permitem maior flexibilidade ao utilizador, como facilitam a implementação de parâmetros diferenciais de alinhamento para as diversas regiões do mtDNA, particularmente, zona codificante *versus* zona não codificante. De facto, a penalização das gaps implementada no CLUSTAL W pode ser demasiado restritiva para locais não codificantes, como a região controlo do mtDNA, onde a perda de certas bases possa ter efeito idêntico à substituição. Tal facto ainda não foi estudado.

Uma das opções a explorar será desenvolver esta ferramenta com a implementação do algoritmo de alinhamento do CLUSTAL W, resolvendo deste modo a eficiência computacional, a eficiência de alinhamento e o alinhamento múltiplo de sequências. Será contudo desejável avaliar mais profundamente o impacto de algumas modificações às restrições *a priori* do CLUSTAL W como: (1) relaxamento de penalizações dos *gaps* nas zonas não codificantes; (2) penalizações para transversões comparativamente a transições.

Referências bibliográficas:

Anderson S, Bankier AT, Barrell BG, de Bruijn MH, Coulson AR, Drouin J, Eperon IC, Nierlich DP, Roe BA, Sanger F, Schreier PH, Smith AJ, Staden R, Young IG (1981) Sequence and organization of the human mitochondrial genome. Nature. 290:457-465.

Andrews RM, Kubacka I, Chinnery PF, Lightowlers RN, Turnbull DM, Howell N (1999) Reanalysis and revision of the Cambridge reference sequence for human mitochondrial DNA. Nat Genet. 23:147.

Ayala FJ (1986) On the virtues and pitfalls of the molecular evolutionary clock. J Hered. 77:226-235.

Belle EM, Piganeau G, Gardner M, Eyre-Walker A (2005) An investigation of the variation in the transition bias among various animal mitochondrial DNA. Gene. 355:58-66.

Benson D, Lipman DJ, Ostell J (1993) GenBank. Nucleic Acids Res. 21:2963-2965.

Benson DA, Boguski M, Lipman DJ, Ostell J (1994) GenBank. Nucleic Acids Res. 22:3441-3444.

Benson DA, Boguski M, Lipman DJ, Ostell J (1996) GenBank. Nucleic Acids Res. 24:1-5.

Benson DA, Boguski MS, Lipman DJ, Ostell J (1997) GenBank. Nucleic Acids Res. 25:1-6.

Benson DA, Boguski MS, Lipman DJ, Ostell J, Ouellette BF (1998) GenBank. Nucleic Acids Res. 26:1-7.

Benson DA, Boguski MS, Lipman DJ, Ostell J, Ouellette BF, Rapp BA, Wheeler DL (1999) GenBank. Nucleic Acids Res. 27:12-17.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL (2000) GenBank. Nucleic Acids Res. 28:15-18.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL (2002) GenBank. Nucleic Acids Res. 30:17-20.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2003) GenBank. Nucleic Acids Res. 31:23-27.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2004) GenBank: update. Nucleic Acids Res. 32(Database issue):D23-26.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2005) GenBank. Nucleic Acids Res. 33(Database issue):D34-38.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2006) GenBank. Nucleic Acids Res. 34(Database issue):D16-20.

Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2007) GenBank. Nucleic Acids Res. 35(Database issue):D21-25.

Broughton RE, Reneau PC (2006) Spatial covariation of mutation and nonsynonymous substitution rates in vertebrate mitochondrial genomes. Mol Biol Evol. 23:1516-1524.

Giles RE, Blanc H, Cann HM, Wallace DC (1980) Maternal inheritance of human mitochondrial DNA. Proc Natl Acad Sci USA. 77:6715-6719.

Hall TA (1999) BioEdit: a user-friendly biological sequence alignment editor and analysis program for Windows 95/98/NT. Nucl Acids Symp Ser. 41:95-98.

Helm M, Brule H, Friede D, Giege R, Putz D, Florentz C (2000) Search for characteristic structural features of mammalian mitochondrial tRNAs. RNA. 6:1356-1379.

Howell N, Kubacka I, Mackey DA (1996) How rapidly does the human mitochondrial genome evolve? Am J Hum Genet. 59:501-509.

Ingman M, Kaessmann H, Paabo S, Gyllensten U (2000) Mitochondrial genome variation and the origin of modern humans. Nature. 408:708-713.

Kivisild T, Shen P, Wall DP, Do B, Sung R, Davis K, Passarino G, Underhill PA, Scharfe C, Torroni A, Scozzari R, Modiano D, Coppa A, de Knijff P, Feldman M, Cavalli-Sforza LL, Oefner PJ (2006) The role of selection in the evolution of human mitochondrial genomes. Genetics. 172:373-387.

Macaulay VA, Richards MB, Forster P, Bendall KE, Watson E, Sykes B, Bandelt HJ (1997) mtDNA mutation rates-no need to panic. Am J Hum Genet. 61:983-990.

Meyer S, Weiss G, von Haeseler A (1999) Pattern of nucleotide substitution and rate heterogeneity in the hypervariable regions I and II of human mtDNA. Genetics. 152:1103-1110.

Mishmar D, Ruiz-Pesini E, Golik P, Macaulay V, Clark AG, Hosseini S, Brandon M, Easley K, Chen E, Brown MD, Sukernik RI, Olckers A, Wallace DC (2003) Natural selection shaped regional mtDNA variation in humans. Proc Natl Acad Sci USA. 100:171-176.

Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48:443-453.

Nei M (1987) Molecular Evolutionary Genetics. Columbia University Press, New York, NY, USA.

Pruitt KD, Tatusova T, Maglott DR (2005) NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res. 33(Database issue):D501-504.

Rozas J, Sanchez-DelBarrio JC, Messeguer X, Rozas R. (2003) DnaSP, DNA polymorphism analyses by the coalescent and other methods. Bioinformatics. 19:2496-2497.

Saitou N, Nei M (1987) The Neighbor-Joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4: 406-425.

Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22:4673-4680.

Vasconcelos AT, Guimaraes AC, Castelletti CH, Caruso CS, Ribeiro C, Yokaichiya F, Armoa GR, Pereira G da S, da Silva IT, Schrago CG, Fernandes AL, da Silveira AR, Carneiro AG, Carvalho BM, Viana CJ, Gramkow D, Lima FJ, Correa LG, Mudado Mde A, Nehab-Hess P, Souza R, Correa RL, Russo CA. (2005) MamMiBase: a mitochondrial genome database for mammalian phylogenetic studies. Bioinformatics. 21:2566-2567.

Anexo 1

Ficheiro GenBank do Primata *Pan troglodytes*

(estão apenas representados o início e fim das 16554bp).

```
LOCUS       NC_001643              16554 bp    DNA     circular PRI 03-AUG-2004
DEFINITION  Pan troglodytes mitochondrion, complete genome.
ACCESSION   NC_001643
VERSION     NC_001643.1  GI:5835121
PROJECT     GenomeProject:11814
KEYWORDS    .
SOURCE      mitochondrion Pan troglodytes (chimpanzee)
  ORGANISM  Pan troglodytes
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Pan.
REFERENCE   1  (sites)
  AUTHORS   Horai,S., Hayasaka,K., Kondo,R., Tsugane,K. and Takahata,N.
  TITLE     Recent African origin of modern humans revealed by complete
            sequences of hominoid mitochondrial DNAs
  JOURNAL   Proc. Natl. Acad. Sci. U.S.A. 92 (2), 532-536 (1995)
   PUBMED   7530363
REFERENCE   2  (sites)
  AUTHORS   Horai,S., Satta,Y., Hayasaka,K., Kondo,R., Inoue,T., Ishida,T.,
            Hayashi,S. and Takahata,N.
  TITLE     Man's place in Hominoidea revealed by mitochondrial DNA genealogy
  JOURNAL   J. Mol. Evol. 35 (1), 32-43 (1992)
   PUBMED   1518083
REFERENCE   3  (sites)
  AUTHORS   Foran,D.R., Hixson,J.E. and Brown,W.M.
  TITLE     Comparisons of ape and human sequences that regulate mitochondrial
            DNA transcription and D-loop DNA synthesis
  JOURNAL   Nucleic Acids Res. 16 (13), 5841-5861 (1988)
   PUBMED   3399380
REFERENCE   4  (sites)
  AUTHORS   Hixson,J.E. and Brown,W.M.
  TITLE     A comparison of the small ribosomal RNA genes from the
            mitochondrial DNA of the great apes and humans: sequence,
            structure, evolution, and phylogenetic implications
  JOURNAL   Mol. Biol. Evol. 3 (1), 1-18 (1986)
   PUBMED   3444394
REFERENCE   5  (bases 1 to 16554)
  AUTHORS   Hayasaka,K.
  TITLE     Direct Submission
  JOURNAL   Submitted (02-SEP-1994) Human Genetics, National Institute of
            Genetics, 1,111 Yata, Mishima, Shizuoka 411, Japan
COMMENT     REVIEWED REFSEQ: This record has been curated by NCBI staff. The
            reference sequence was derived from D38113.
FEATURES             Location/Qualifiers
     source          1..16554
                     /organism="Pan troglodytes"
                     /organelle="mitochondrion"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:9598"
     tRNA            1..71
                     /product="tRNA-Phe"
     tRNA            1..71
                     /product="tRNA-Phe"
```

/note="Calculated by tRNAscan-SE"
rRNA 72..1020
/product="12S ribosomal RNA"
tRNA 1021..1089
/product="tRNA-Val"
tRNA 1021..1089
/product="tRNA-Val"
/note="Calculated by tRNAscan-SE"
rRNA 1090..2647
/product="16S ribosomal RNA"
STS 1481..1622
/standard_name="D3S2893E"
/db_xref="UniSTS:150884"
tRNA 2648..2730
/product="tRNA-Leu"
/note="Calculated by tRNAscan-SE"
tRNA 2648..2722
/product="tRNA-Leu"
/note="codons recognized: UUR"
gene 2725..3681
/gene="ND1"
/db_xref="GeneID:807867"
CDS <2725..3681
/gene="ND1"
/codon_start=1
/transl_table=2
/product="NADH dehydrogenase subunit 1"
/protein_id="NP_008186.1"
/db_xref="GI:5835127"
/db_xref="GeneID:807867"
/translation="TPMTNLLLLIVPILIAMAFLMLTERKILGYMQLRKGPNIVGPYG
LLQPFADAMKLFTKEPLKPSTSTITLYITAPTLALTIALLLWTPLPMPNPLVNLNLGL
LFILATSSLAVYSILWSGWASNSNYALIGALRAVAQTISYEVTLAIILLSTLLMSGSF
NLSTLVTTQEHLWLILPTWPLAMMWFISTLAETNRTPFDLTEGESELVSGFNIEYAAG
PFALFFMAEYMNIIMMNTLTATIFLGATYNTHSPELYTTYFVTKALLLTSLFLWIRTA
YPRFRYDQLMHLLWKNFLPLTLASLMWYISMPTTISSIPPQT"
tRNA 3681..3749
/product="tRNA-Ile"
tRNA 3681..3749
/product="tRNA-Ile"
/note="Calculated by tRNAscan-SE"
tRNA complement(3747..3818)
/product="tRNA-Gln"
tRNA complement(3747..3818)
/product="tRNA-Gln"
/note="Calculated by tRNAscan-SE"
tRNA 3820..3887
/product="tRNA-Met"
tRNA 3820..3887
/product="tRNA-Met"
/note="Calculated by tRNAscan-SE"
gene 3888..4931
/gene="ND2"
/db_xref="GeneID:807865"
CDS 3888..4931
/gene="ND2"
/codon_start=1
/transl_table=2
/product="NADH dehydrogenase subunit 2"
/protein_id="NP_008187.1"
/db_xref="GI:5835126"
/db_xref="GeneID:807865"

/translation="MNPLAQPIIYSTILTGTLITALSSHWFFTWVGLEMNMLAFIPIL
TKKMSPRSTEAAIKYFLTQATASMILLMAILSNSMLSGQWTMTNTTNQYSSLMIMMAM
AMKLGMAPFHFWVPEVTQGTPLMSGLLLLTWQKLAPISIMYQISSSLNVNLLLTLSIL
SIMAGSWGGLNQTQLRKILAYSSITHMGWMMAVLPYNPNMTILNLTIYIILTTTAFLL
LNLNSSTTTLLLSRTWNKLTWLTPLIPSTLLSLGGLPPLTGFLPKWVIIEEFTKNNSL
IIPTIMAIITLLNLYFYLRLIYSTSITLLPMSNNVKMKWQFEHTKPTPFLPTLITLTT
LLLPISPFMLMIL"
tRNA 4930..4997
/product="tRNA-Sec"
/note="Calculated by tRNAscan-SE"
/selenocysteine
tRNA 4930..4997
/product="tRNA-Trp"
tRNA complement(5005..5073)
/product="tRNA-Ala"
tRNA complement(5005..5073)
/product="tRNA-Ala"
/note="Calculated by tRNAscan-SE"
tRNA complement(5075..5147)
/product="tRNA-Asn"
tRNA complement(5075..5147)
/product="tRNA-Asn"
/note="Calculated by tRNAscan-SE"
tRNA complement(5179..5244)
/product="tRNA-Cys"
tRNA complement(5179..5244)
/product="tRNA-Cys"
/note="Calculated by tRNAscan-SE"
tRNA complement(5244..5309)
/product="tRNA-Tyr"
tRNA complement(5244..5309)
/product="tRNA-Tyr"
/note="Calculated by tRNAscan-SE"
gene 5321..6862
/gene="COX1"
/db_xref="GeneID:807866"
CDS 5321..6862
/gene="COX1"
/codon_start=1
/transl_table=2
/product="cytochrome c oxidase subunit I"
/protein_id="NP_008188.1"
/db_xref="GI:5835128"
/db_xref="GeneID:807866"
/translation="MFTDRWLFSTNHKDIGTLYLLFGAWAGVLGTALSLLIRAELGQP
GNLLGNDHIYNVIVTAHAFVMIFFMVMPIMIGGFGNWLVPLMIGAPDMAFPRMNNMSF
WLLPPSLLLLLASAMVEAGAGTGWTVYPPLAGNYSHPGASVDLTIFSLHLAGISSILG
AINFITTIINMKPPAMTQYQTPLFVWSVLITAVLLLLSLPVLAAGITMLLTDRNLNTT
FFDPAGGGDPILYQHLFWFFGHPEVYILILPGFGMISHIVTYYSGKKEPFGYMGMVWA
MMSIGFLGFIVWAHHMFTVGMDVDTRAYFTSATMIIAIPTGVKVFSWLATLHGSNMKW
SAAVLWALGFIFLFTVGGLTGIVLANSSLDIVLHDTYYVVAHFHYVLSMGAVFAIMGG
FIHWFPLFSGYTLDQTYAKIQFAIMFIGVNLTFFPQHFLGLSGMPRRYSDYPDAYTTW
NVLSSVGSFISLTAVMLMIFMIWEAFASKRKVLMVEEPSANLEWLYGCPPPYHTFEEP
VYMKS"
tRNA complement(6862..6933)
/product="tRNA-Ser"
/note="codons recognized: UCN"
tRNA complement(6863..6931)
/product="tRNA-Ser"
/note="Calculated by tRNAscan-SE"
STS 6900..7801
/standard_name="PMC20756P2"

/db_xref="UniSTS:271962"
STS 6900..7782
/standard_name="PMC109173P1"
/db_xref="UniSTS:270154"
tRNA 6935..7002
/product="tRNA-Asp"
tRNA 6935..7002
/product="tRNA-Asp"
/note="Calculated by tRNAscan-SE"
gene 7003..7686
/gene="COX2"
/db_xref="GeneID:807864"
CDS 7003..7686
/gene="COX2"
/codon_start=1
/transl_table=2
/product="cytochrome c oxidase subunit II"
/protein_id="NP_008189.1"
/db_xref="GI:5835122"
/db_xref="GeneID:807864"
/translation="MAHAAQVGLQDATSPIMEELIIFHDHALMIIFLICFLVLYALFL
TLTTKLTNTSISDAQEMETVWTILPAIILVLIALPSLRILYMTDEVNDPSFTIKSIGH
QWYWTYEYTDYGGLIFNSYMLPPLFLEPGDLRLLDVDNRVVLPVEAPVRMMITSQDVL
HSWAVPTLGLKTDAIPGRLNQTTFTATRPGVYYGQCSEICGANHSFMPIVLELIPLKI
FEMGPVFTL"
tRNA 7713..7782
/product="tRNA-Lys"
tRNA 7713..7782
/product="tRNA-Lys"
/note="Calculated by tRNAscan-SE"
gene 7784..7990
/gene="ATP8"
/db_xref="GeneID:807862"
CDS 7784..7990
/gene="ATP8"
/codon_start=1
/transl_table=2
/product="ATP synthase F0 subunit 8"
/protein_id="NP_008190.1"
/db_xref="GI:5835123"
/db_xref="GeneID:807862"
/translation="MPQLNTAVWPTMITPMLLTLFLVTQLKMLNSNYHLPPSPKPMKM
KNYNKPWEPKWTKIYSLHSLPPQS"
gene 7945..8625
/gene="ATP6"
/db_xref="GeneID:807859"
CDS 7945..8625
/gene="ATP6"
/codon_start=1
/transl_table=2
/product="ATP synthase F0 subunit 6"
/protein_id="NP_008191.1"
/db_xref="GI:5835129"
/db_xref="GeneID:807859"
/translation="MNENLFASFAAPTILGLPAAVLIILFPPLLVPTSKHLINNRLIT
TQQWLIQLTSKQMMTMHSTKGRTWSLMLVSLIIFITTTNLLGLLPHSFTPTTQLSMNL
AMAIPLWAGAVVMGFRFKTKNALAHFLPQGTPTPLIPMLVIIETISLLIQPMALAVRL
TANITAGHLLMHLIGSATLALSTINLPYALIIFTILILLTILEIAVALIQAYVFTLLV
SLYLHDNT"
gene 8625..9408
/gene="COX3"
/db_xref="GeneID:807869"

```
CDS             8625..9408
                /gene="COX3"
                /note="TAA stop codon is completed by the addition of 3' A
                residues to the mRNA"
                /codon_start=1
                /transl_except=(pos:9408,aa:TERM)
                /transl_table=2
                /product="cytochrome c oxidase subunit III"
                /protein_id="NP_008192.1"
                /db_xref="GI:5835130"
                /db_xref="GeneID:807869"
                /translation="MTHQSHAYHMVKPSPWPLTGALSALLMTSGLAMWFHFYSTTLLT
                LGLLTNTLTMYQWWRDVMREGTYQGHHTPPVQKGLRYGMILFITSEVFFFAGFFWAFY
                HSSLAPTPQLGGHWPPTGITPLNPLEVPLLNTSVLLASGVSITWAHHSLMENNRNQMI
                QALLITILLGLYFTLLQASEYFESPFTISDGIYGSTFFVATGFHGLHVIIGSTFLTIC
                LIRQLMFHFTSKHHFGFQAAAWYWHFVDVVWLFLYVSIYWWGS"
tRNA            9409..9476
                /product="tRNA-Gly"
tRNA            9409..9476
                /product="tRNA-Gly"
                /note="Calculated by tRNAscan-SE"
gene            9477..9822
                /gene="ND3"
                /db_xref="GeneID:807870"
CDS             9477..9822
                /gene="ND3"
                /note="TAA stop codon is completed by the addition of 3' A
                residues to the mRNA'"
                /codon_start=1
                /transl_except=(pos:9822,aa:TERM)
                /transl_table=2
                /product="NADH dehydrogenase subunit 3"
                /protein_id="NP_008193.1"
                /db_xref="GI:5835131"
                /db_xref="GeneID:807870"
                /translation="MNFVLILMTNTLLALLLMIITFWLPQLNSYMEKSTPYECGFDPM
                SPARVPFSMKFFLVAITFLLFDLEIALLLPLPWALQTANLPLMVTSSLLLITILALSL
                AYEWLQKGLDWTE"
tRNA            9823..9887
                /product="tRNA-Arg"
tRNA            9823..9887
                /product="tRNA-Arg"
                /note="Calculated by tRNAscan-SE"
gene            9888..10184
                /gene="ND4L"
                /db_xref="GeneID:807868"
CDS             9888..10184
                /gene="ND4L"
                /codon_start=1
                /transl_table=2
                /product="NADH dehydrogenase subunit 4L"
                /protein_id="NP_008194.1"
                /db_xref="GI:5835124"
                /db_xref="GeneID:807868"
                /translation="MPLIYMNIMLAFTISLLGMLVYRSHLMSSLLCLEGMMLSLFIMA
                TLMTLNTHSLLANIVPITMLVFAACEAAVGLALLVSISNTYGLDYVHNLNLLQC"
gene            10178..11555
                /gene="ND4"
                /db_xref="GeneID:807863"
CDS             10178..11555
                /gene="ND4"
                /note="TAA stop codon is completed by the addition of 3' A
```

residues to the mRNA"
/codon_start=1
/transl_except=(pos:11555,aa:TERM)
/transl_table=2
/product="NADH dehydrogenase subunit 4"
/protein_id="NP_008195.1"
/db_xref="GI:5835132"
/db_xref="GeneID:807863"
/translation="MLKLIIPTIMLLPLTWFSKKRMIWINTTTHSLIISTIPLLFFNQ
INNNLFSCSLPFSSDPLTTPLLMLTAWLLPLTIMASQRHLSNEPLSRKKLYLSMLISL
QISLIMTFSATELIMFYIFFETTLIPTLAIITRWGNQPERLNAGTYFLFYTLVGSLPL
LIALIYTHNTLGSLNILLLTLTTQELSNTWANNLMWLAYTMAFMVKMPLYGLHLWLPK
AHVEAPIAGSMVLAAVLLKLGGYGMMRLTLILNPLTKHMAYPFLMLSLWGMIMTSSIC
LRQTDLKSLIAYPSVSHMALVVTAILIQTPWSFTGAIILMIAHGLTSSLLSCLANSNY
ERTHSRIMILSQGLQTLLPLMAFWWLLASLANLALPPTINLLGELSVLVTSFSWSNTT
LLLTGFNMLITALYSLYMFTTTQWGSLTHHINSMKPSFTRENTLMFLHLSPILLLSLN
PDIITGFTS"
tRNA 11556..11624
/product="tRNA-His"
tRNA 11556..11624
/product="tRNA-His"
/note="Calculated by tRNAscan-SE"
tRNA 11625..11683
/product="tRNA-Ser"
/note="codons recognized: AGY"
tRNA 11684..11754
/product="tRNA-Leu"
/note="codons recognized: CUN"
tRNA 11684..11754
/product="tRNA-Leu"
/note="Calculated by tRNAscan-SE"
gene 11755..13566
/gene="ND5"
/db_xref="GeneID:807860"
CDS 11755..13566
/gene="ND5"
/codon_start=1
/transl_table=2
/product="NADH dehydrogenase subunit 5"
/protein_id="NP_008196.1"
/db_xref="GI:5835125"
/db_xref="GeneID:807860"
/translation="MTMYATMTTLALTSLIPPILGALINPNKKNSYPHYVKSIIASTF
IISLFPTTMFMCLDQETIISNWHWATTQTTQLSLSFKLDYFSMTFIPVALFVTWSIME
FSLWYMDSDPNINQFFKYLLIFLITMLILVTANNLFQLFIGWEGVGIMSFLLISWWYA
RTDANTAAIQAILYNRIGDIGFVLALAWFLLHSNSWDPQQMILLSTNTDLTPLLGFLL
AAAGKSAQLGLHPWLPSAMEGPTPVSALLHSSTMVVAGIFLLIRFYPLAENNPLIQTL
TLCLGAITTLFAAVCALTQNDIKKIVAFSTSSQLGLMMVTIGINQPHLAFLHICTHAF
FKAMLFMCSGSIIHNLNNEQDIRKMGGLLKTMPLTSTSLTIGSLALAGMPFLTGFYSK
DLIIETANMSYTNAWALSITLIATSLTSAYSTRMILLTLTGQPRFPTLTNINENNPTL
LNPIKRLTIGSLFAGFLITNNILPMSTPQVTIPLYLKLTALGVTSLGLLTALDLNYLT
SKLKMKSPLYTFHFSNMLGFYPNIMHRSIPYLGLLTSQNLPLLLLDLTWLEKLLPKTI
SQYQISASITTSTQKGMIKLYFLSFFFPLILTLLLIT"
gene complement(13567..14091)
/gene="ND6"
/db_xref="GeneID:807861"
CDS complement(13567..14091)
/gene="ND6"
/codon_start=1
/transl_table=2
/product="NADH dehydrogenase subunit 6"
/protein_id="NP_008197.1"

/db_xref="GI:5835133"
/db_xref="GeneID:807861"
/translation="MTYALFLLSVSLVMGFVGFSSKPSPIYGGLVLIVSGVVGCAIIL
NYGGGYMGLMVFLIYLGGMMVVFGYTTAMAIEEYPEAWGSGVEVLVSVLVGLAMEVGL
VLWVKGYDGMVVVVNFNSVGSWMIYEGEGPGLIREDPIGAGALYDYGRWLVVVTGWTL
FVGVYIVIEIARGN"
tRNA complement(14092..14160)
/product="tRNA-Glu"
tRNA complement(14092..14160)
/product="tRNA-Glu"
/note="Calculated by tRNAscan-SE"
gene 14165..15305
/gene="CYTB"
/db_xref="GeneID:807858"
CDS 14165..15305
/gene="CYTB"
/note="TAA stop codon is completed by the addition of 3' A
residues to the mRNA"
/codon_start=1
/transl_except=(pos:15305,aa:TERM)
/transl_table=2
/product="cytochrome b"
/protein_id="NP_008198.1"
/db_xref="GI:5835134"
/db_xref="GeneID:807858"
/translation="MTPTRKINPLMKLINHSFIDLPTPSNISAWWNFGSLLGACLILQ
ITTGLFLAMHYSPDASTAFSSIAHITRDVNYGWIIRYLHANGASMFFICLFLHIGRGL
YYGSFLYLETWNIGIILLLTTMATAFMGYVLPWGQMSFWGATVITNLLSAIPYIGTDL
VQWVWGGYSVDSPTLTRFFTFHFILPFIITALTTLHLLFLHETGSNNPLGITSHSDKI
TFHPYYTIKDILGLFLFLLILMTLTLFSPGLLGDPDNYTLANPLNTPPHIKPEWYFLF
AYTILRSIPNKLGGVLALLLSILILTAIPVLHTSKQQSMMFRPLSQLLYWLLATDLLI
LTWIGGQPVSYPFITIGQMASVLYFTTILILMPIASLIENKMLEWT"
STS 14230..14339
/gene="CYTB"
/standard_name="D5S1608E"
/db_xref="UniSTS:151029"
tRNA 15306..15371
/product="tRNA-Thr"
tRNA 15306..15371
/product="tRNA-Thr"
/note="Calculated by tRNAscan-SE"
tRNA complement(15374..15441)
/product="tRNA-Pro"
tRNA complement(15374..15441)
/product="tRNA-Pro"
/note="Calculated by tRNAscan-SE"
STS 15396..15837
/standard_name="PMC16238P1"
/db_xref="UniSTS:271472"
D-loop 15442..16554
ORIGIN
1 gtttatgtag cttacccct caaagcaata cactgaaaat gtttcgacgg gtttacatca
61 ccccataaac aaacaggttt ggtcctagcc tttctattag ctcttagtaa gattacacat
121 gcaagcatcc ccgccccgtg agtcaccctc taaatcgcca tgatcaaaag gaacaagtat
181 caagcacgca gcaatgcagc tcaaaacgct tagcctagcc acacccccac gggagacagc
241 agtgataaac ctttagcaat aaacgaaagt ttaactaagc catactaacc tcagggttgg
301 tcaatttcgt gctagccacc gcggtcatac gattaaccca agtcaataga aaccggcgta
361 aagagtgttt tagatcaccc ccccataaag ctaaaattca cctgagttgt aaaaaactcc
…
16261 acatcataac aaaaaattcc cacaaacccc cccttccccc cggccacagc actcaaacaa
16321 atctctgcca aaccccaaaa acaaagaacc cagacgccag cctagccaga cttcaaattt
16381 catctttagg cggtatgcac ttttaacagt cacccctcaa ttaacatgcc ctccccccctc

16441 aactcccatt ctactagccc cagcaacgta acccccctact caccctactc aacacatata
16501 ccgctgctaa ccccataccc tgaaccaacc aaaccccaaa gacaccccta caca
//

Anexo 2:

Output do programa de selecção do gene tRNA-Gln dos Primatas

> NC_001643 (tRNA-Gln)
CTAGGACTATAAGAATCGAACTCATCCCTGAGAATCCAAAATTCTCCGTGCCACCTATCA
CACCCCATCCTA
> NC_001644 (tRNA-Gln)
CTAGGACTATGAGAGTCGAACCCATCCCTGAGAATCCAAAATTCTCCGTGCCACCTATCA
CACCCCATCCTA
> NC_001645 (tRNA-Gln)
CTAGGACTATGAGAATTGAACCCATCCCTGAGAATCCAAAATTCTCCGTGCCACCTGTCA
CACCCCATCCTA
> NC_001646 (tRNA-Gln)
CTAGGACTATGGGAATTGAACCCACCCCTGAGAATCCAAAATTCTCCGTGCCACCCATCA
CACCCCATCCTA
> NC_001807 (tRNA-Gln)
CTAGGACTATGAGAATCGAACCCATCCCTGAGAATCCAAAATTCTCCGTGCCACCTATCA
CACCCCATCCTA
> NC_001992 (tRNA-Gln)
CTAGGATTACAGGCATCGAACCTGCTCCTGAGAATCCAAACTTCTCCGTGCTACCCCTCA
CACTCCATCCTA
> NC_002082 (tRNA-Gln)
CTAGAACCATAGGAGTCGAACCCATCCCTGAGAACCCAAAACTCTCCGTGCCACCCGTCG
CACCCTGTTCTA
> NC_002763 (tRNA-Gln)
CTAGAATTATAGGACTCGAACCTATTCCTAAGAACTCAAAATCCTCCGTGCTACCTATTA
CACCATACTCTA
> NC_002764 (tRNA-Gln)
CTAGGACTGTAGGTATCGAACCTACCCCTGAGAACCCAAACTTCTCCGTGCTACCTCATA
CACCCCATCCTA
> NC_002765 (tRNA-Gln)
CTAGAACCTTAGGGCTTGAACCTACTCTTAAGAATTCAAAATTCTTCGTGCTACCAAATA
CACCAAGTCCTA
> NC_002811 (tRNA-Gln)
CTAGAATTACAGGAATTGAACCTATACTTAAGAATCCAAAATTCTTCGTGCTACCCAATT
ACACAATATTCTA
> NC_004025 (tRNA-Gln)
CTAGAATCACAGGGATCGAACCTGCTCTTAAGGATTCAAAGTCCTTCGTGCTACCTAAAA
CACACCCTATTCTA
> NC_005943 (tRNA-Gln)
CTAGGACTGTAGGTATCGAACCTACCCCTGAGAATCCAAACTTCTCCGTGCTACCCTACA
CACTCTATCCTA
> NC_006900 (tRNA-Gln)
CTAGAATTATAGGCATCGAACCTACCCCTGAGAATTCAAATTCCTCCGTGCTACCTATCA
CACCTCATTCTA
> NC_006901 (tRNA-Gln)
CTAGAACCATAGGTATCGAACCTACTCCTGAGAACCCAAATTTCTCCGTGCTACCCATTA
CACCTTGTTCTA
> NC_007009 (tRNA-Gln)
CTAAGATCGCAGGCATTGAACCTACCCCTGAGAATCCAAACTTCTCCGTGCTACCTAACA
CACCCTATCCTA
> NC_008066 (tRNA-Gln)
CTAAGATTGCAGGCATTGAACCTACCCCTGAGAATCCAAACTTCTCCGTGCTACCTAACA
CACCCCATCCTA
> NC_008215 (tRNA-Gln)
CTAGAAATATAGGCATCGAACCTACTCCTGAGAATTCAAATTTCTCCGTGCCACCTATTA
CACCTCCTTCTA
> NC_008216 (tRNA-Gln)

CTAGAATCATAGGTATTGAACCTACTCCTGAGAATCCAAATTTCTCCGTGCTACCTATTA
CACCCAACTCTA
> NC_008217 (tRNA-Gln)
CTAGAATCATAGGCATTGAACCTACCCCTGAGAATTCAAACTTCTCCGTGCTACCTATTA
CACCCAATTCTA
> NC_008218 (tRNA-Gln)
CTAGAATTATAGGTATCGAACCTACCCCTGAGAATTCAAATTCCTCCGTGCTACCTATCA
CACCCCATTCTA
> NC_008219 (tRNA-Gln)
CTAGAATTATAGGCATTGAACCTACCCCTGAGGATCCAAACTCCTCCGTGCTACCTATTA
CACCCCATTCTA

Anexo 3:

Código fonte do programa de selecção de genes do mtDNA

```
/*
*
* GeneSelection.h
*
*/

//---------------------------------------------------------------------

#ifndef GeneSelectionH
#define GeneSelectionH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <Dialogs.hpp>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <fstream>
#include <stdio.h>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *File1;
        TMenuItem *Open1;
        TMenuItem *Exit1;
        TMenuItem *Help1;
        TMenuItem *About1;
        TLabel *Label1;
        TButton *Button1;
        TComboBox *ComboBox1;
        TOpenDialog *OpenDialog1;
        TSaveDialog *SaveDialog1;
        TListBox *ListBox1;
        TLabel *Label2;
        TMenuItem *NewSelection1;
        void __fastcall Exit1Click(TObject *Sender);
        void __fastcall ComboBox1Change(TObject *Sender);
        void __fastcall Open1Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall NewSelection1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------
```

```
#endif

/*
*
* GeneSelection.cpp
*
*/

//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "GeneSelection.h"

//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

using namespace std;

const int N_GENES=38;

char * genes[N_GENES] = {"Control Region", "tRNA-Phe", "12S rRNA",
"tRNA-Val", "16S rRNA", "tRNA-Leu(UUR)", "ND1", "tRNA-Ile", "tRNA-Gln",
"tRNA-Met", "ND2", "tRNA-Trp", "tRNA-Ala", "tRNA-Asn", "tRNA-Cys", "tRNA-
Tyr", "COX1", "tRNA-Ser(UCN)", "tRNA-Asp", "COX2", "tRNA-Lys", "ATP8",
"ATP6", "COX3", "tRNA-Gly", "ND3", "tRNA-Arg", "ND4L", "ND4", "tRNA-His",
"tRNA-Ser(AGY)", "tRNA-Leu(CUN)", "ND5", "ND6", "tRNA-Glu", "CYTB", "tRNA-
Thr", "tRNA-Pro"};

char line[200], kind[50], location[100], location_f[100];
char seq_gene[10000], seq_gene2[10000], accession[20];
char prefix1[100], prefix2[100], prefix3[100], prefix4[100];
char s_aux[200], filename[300], outfilename[300];
char path[100][300];

int n_gene=-1, num_seq=0, num_files=0;
int k, j, i, s, begin, end, start, num_c, ns;
int first_line1, last_line1, first_line2, last_line2;
bool success=false;
char *fn, *begin_str, *end_str, *fe, *line2;

FILE *fd;
ofstream outf;

//---------------------------------------------------------------------
int position_split(char position[], int *end){
  int begin, tmp;
  begin_str=strtok(position,"..(");
  if(begin_str!=NULL && begin_str[0]=='>') begin_str = begin_str+1;
  begin=atoi(begin_str);
  if(begin==0){
    begin_str=strtok(NULL,"..(");
    if(begin_str!= NULL && begin_str[0]=='>') begin_str = begin_str+1;
      begin=atoi(begin_str);
    }

  end_str=strtok(NULL,"..(");
  if(end_str!=NULL && end_str[0]=='>') end_str = end_str+1;
  *end = atoi(end_str);
  if(*end==0 || begin==0) return -1;
  return begin;
```

```
}

//-----------------------------------------------------------------------
int position_split_CR(char position[]){
  int begin1, end1, begin2, end2, tmp;
  char *part1, *nums2, *nums1;

  if(strstr(location,"complement")!=NULL){
    part1=strtok(position,",");
    nums2=strtok(NULL,",");
    strtok(part1,"(");

    if(strstr(location,"complement")!=NULL){
      strtok(NULL,"(");
    }

    nums1=strtok(NULL,"(");

    begin1 =atoi(strtok(nums1,".."));
    end1 =atoi(strtok(NULL,".."));
    begin2 =atoi(strtok(nums2,".."));
    end2 =atoi(strtok(NULL,".."));
  }

  if(begin1 < begin2){
    tmp=begin1;
    begin1=begin2;
    begin2=tmp;
    tmp=end1;
    end1=end2;
    end2=tmp;
  }

  //begin2..end2
  first_line2 = ((begin2-1)/60)*60+1;
  last_line2 = ((end2-1)/60)*60+1;

  while (strcmp(kind,"ORIGIN")!=0 && !feof(fd) ) {
    fgets(line, 200, fd);
    sscanf(line, "%s", kind);
  }
  if(feof(fd)) return 0;

  while (start!=first_line2){
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }

  while(1){
    strtok(line," ");
    line2=strtok(NULL,"");
    for(j=0; line2[j]!='\0'; j++ )
      if(isalpha(line2[j]) || line2[j]=='-')
        seq_gene2[ns++]=line2[j];

    if(start==last_line2) break;
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }
  seq_gene2[ns]='\0';

  //end1..begin1
  first_line1 = ((begin1-1)/60)*60+1;
```

```
  last_line1 = ((end1-1)/60)*60+1;

  while (start!=first_line1){
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }
  ns=0;
  while(1){
    strtok(line," ");
    line2=strtok(NULL,"");
    for(j=0; line2[j]!='\0'; j++)
      if(isalpha(line2[j]) || line2[j]=='-')
         seq_gene[ns++]=line2[j];

    if(start==last_line1) break;
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }
  seq_gene[ns]='\0';

  outf << "> " << accession << "  (" << genes[n_gene] << ")\n";
  for(j=begin1-first_line1; j<=end1-first_line1; j++){
    outf << (char) toupper(seq_gene[j]); num_c++;
    if(num_c%60==0) outf << endl;
  }
  for(j=begin2-first_line2; j<=end2-first_line2; j++){
    outf << (char) toupper(seq_gene2[j]); num_c++;
    if(num_c%60==0) outf << endl;
  }

  if(num_c%60!=0) outf << endl;
  num_seq++;
  fclose(fd);

  return 0;
}

//---------------------------------------------------------------------
void select_gene(){

  strcpy(prefix1,"/gene=");
  strcpy(prefix2,"/product=");
  strcpy(prefix3,"/gene=");
  strcpy(prefix4,"/product=");

  num_seq=0;

  if(n_gene==0){//D-loop
    prefix1[0]='\0';
    prefix2[0]='\0';
    strcat(prefix1,"/note=\"control");
    strcat(prefix2,"/note=\"D-loop\"");
  } else {
    if(n_gene==17 || n_gene==30){ //tRNA-Ser
      strcat(prefix1, "\"tRNA-Ser\"");
      strcat(prefix2, "\"tRNA-Ser\"");
    } else if(n_gene==5 || n_gene==31){//tRNA-Leu
      strcat(prefix1, "\"tRNA-Leu\"");
      strcat(prefix2, "\"tRNA-Leu\"");
    } else {
      strcat(prefix1,"\"");
      strcat(prefix2,"\"");
      strcat(prefix1, genes[n_gene]);
```

```
      strcat(prefix2, genes[n_gene]);
      strcat(prefix1,"\"");
      strcat(prefix2,"\"");
    }
   }

  if(n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31){
    strcat(prefix3,"\"");
    strcat(prefix4,"\"");
    strcat(prefix3, genes[n_gene]);
    strcat(prefix4, genes[n_gene]);
    strcat(prefix3,"\"");
    strcat(prefix4,"\"");
  }

  outf.open(outfilename);

  for(k=0; k < num_files; k++){

    kind[0]='\0';
    seq_gene[0]='\0';
    start=ns=num_c=begin=0;
    success = false;

    fd=fopen(path[k],"r");

    while (strcmp(kind,"ORIGIN")!=0 && !feof(fd)) {
      fgets(line, 200, fd);
      sscanf(line, "%s %s", kind, location);

    if(strcmp(kind,"ACCESSION")==0){
      strcpy(accession, location);
      continue;
    }

    if(strcmp(kind, "tRNA")==0 || strcmp(kind, "gene")==0 || strcmp(kind,
"rRNA")==0){
      strcpy(location_f, location);
      continue;
    } else if(n_gene==0 && (strcmp(kind, "D-loop")==0 || strcmp(kind,
"misc_feature")==0)){
      success=true;
      break;
    }

    if(n_gene==2){//12S rRNA
      if(strcmp(kind, "/product=\"12S")==0 || strcmp(kind, "/product=\"s-
rRNA\"")==0){
        begin=position_split(location_f, &end);
        success=true;
        break;
      }
    } else if(n_gene==4){//16S rRNA
        if(strcmp(kind, "/product=\"16S")==0 || strcmp(kind, "/product=\"l-
rRNA\"")==0){
          begin=position_split(location_f, &end);
          success=true;
          break;
        }
    }

    if(n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31){
      if(strcmp(prefix3,kind)==0 || strcmp(prefix4,kind)==0){
```

```
    begin=position_split(location_f, &end);
    success=true;
    break;
  }
}

if(strcmp(prefix1,kind)==0 || strcmp(prefix2,kind)==0){
  if( n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31 ){
    fgets(line, 200, fd);
    sscanf(line, "%s %*s %s",kind,location);
  }
  if(n_gene==17){ //tRNA-Ser UCN
    if(strcmp(location,"UCN\"")!=0) continue;
  } else if(n_gene==30){//tRNA-Ser AGY
    if(strcmp(location,"AGY\"" )!=0) continue;
  } else if(n_gene==5){//tRNA-Leu UUR
    if(strcmp(location,"UUR\"" )!=0) continue;
  } else if(n_gene==31){//tRNA-Leu CUN
    if(strcmp(location,"CUN\"" )!=0) continue;
  }
    begin=position_split(location_f, &end);
    success=true;
    break;
  }
}

if(feof(fd)) continue;
if(!success || begin==-1) continue;

if(n_gene==0){//D-loop
  if(strstr(location,"join")!=NULL){//join é substring de location
    position_split_CR(location);
    continue;
  } else{
    begin=position_split(location, &end);
  }
}

first_line1 = ((begin-1)/60)*60+1;
last_line1 = ((end-1)/60)*60+1;

while (strcmp(kind,"ORIGIN")!=0 && !feof(fd)) {
  fgets(line, 200, fd);
  sscanf(line, "%s", kind);
}
if(feof(fd)) continue;

while (start!=first_line1){
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}

while(1){
  strtok(line," ");
  line2=strtok(NULL,"");
  for(j=0; line2[j]!='\0'; j++ )
    if(isalpha(line2[j]) || line2[j]=='-')
      seq_gene[ns++]=line2[j];

  if(start==last_line1) break;
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}
```

```
    seq_gene[ns]='\0';

    outf << "> " << accession << "  (" << genes[n_gene] << ")\n";
    for(j=begin-first_line1; j<=end-first_line1; j++){
      outf << (char) toupper(seq_gene[j]);
      num_c++;
      if(num_c%60==0) outf << endl;
    }
    if(num_c%60!=0) outf << endl;
    num_seq++;
    fclose(fd);

  }
  outf.close();
}

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

//---------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Label1->Caption="Gene:";
  for(i=0; i < N_GENES; i++)
    ComboBox1->Items->Add(genes[i]);
}

//---------------------------------------------------------------------------
--
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
  Form1->Close();
}

//---------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  ComboBox1->ItemHeight=32;
  n_gene=ComboBox1->ItemIndex;
}

//---------------------------------------------------------------------
void __fastcall TForm1::Open1Click(TObject *Sender)
{
  if(OpenDialog1->Execute()){
    sprintf(filename, "%s", OpenDialog1->FileName);
    strcpy(path[num_files++],filename);
    fn = strrchr(filename, '\\');
    fn=fn+1;
    ListBox1->Items->Add(fn);
  }
}

//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)//Run
{
  Label2->Caption=n_gene;
  if(n_gene!=-1){
    Label2->Caption="";
    if(SaveDialog1->Execute()){
```

```
      sprintf(outfilename, "%s", SaveDialog1->FileName);
      fe=strrchr(outfilename,'.');
      if(fe!=NULL){
        if(strcmp(fe,".txt")!=0)
          strcat(outfilename,".txt");
      } else strcat(outfilename,".txt");
      if(n_gene!=-1){
        select_gene();
        if(num_seq!=num_files) Label2->Caption="It was not possible to take
information from all files.";
      }
    }
  } else {
     Label2->Caption="You have to choose a gene.";
  }
}

//---------------------------------------------------------------------
void __fastcall TForm1::NewSelection1Click(TObject *Sender)
{
  num_files=0;
  ListBox1->Clear();
}
//---------------------------------------------------------------------
```

Anexo 4:

Output* (reorganizado em quatro colunas) do programa de alinhamento dos genes do mtDNA para as sequências totais de *Homo sapiens* e *Pan troglodytes

Not aligned
symbols:
a= 1470
b= 1453

7 15992 22
30 16015 10
40 16026 1
42 16027 52
95 16080 7
104 16089 42
147 16132 4
152 16137 31
183 16173 2
190 16175 20
211 16196 15
227 16212 9
237 16222 1
238 16242 5
245 16247 1
263 16248 11
274 16260 1
276 16261 18
295 16280 3
299 16283 11
311 16295 7
324 16302 12
337 16315 5
343 16321 31
375 16352 1
376 16354 2
379 16357 8
388 16366 5
394 16372 9
404 16382 35
440 16418 3
444 16422 5
451 16427 1
459 16429 7
466 16442 8
475 16451 6
483 16458 1
485 16461 2
488 16464 3
491 16469 1
494 16470 7
502 16479 1
504 16481 2
507 16484 5
513 16490 2
517 16492 5
523 16498 1
525 16500 21
547 16522 27
575 16551 20
596 18 27
624 46 6
631 52 1
633 55 19
653 75 59
714 134 2
717 138 5
723 143 14
738 158 3
742 162 10
753 173 5
759 179 56
816 236 11
828 248 33
862 282 9
872 292 21
894 314 14
909 329 23
933 353 30
964 383 2
969 385 12
982 398 25
1008 424 3
1012 428 8
1021 437 21
1043 459 7
1051 467 57
1109 525 13
1123 539 119
1243 659 49
1293 709 12
1306 722 38
1345 761 42
1388 804 7
1396 812 12
1409 825 8
1418 834 22
1441 857 79
1521 937 1
1523 939 6
1529 946 3
1533 950 8
1542 959 16
1559 976 62
1622 1039 34
1657 1074 9
1667 1084 8
1676 1093 18
1695 1112 7
1703 1120 8
1712 1129 9
1722 1139 41
1763 1181 5
1769 1186 11
1781 1198 32
1814 1231 12
1827 1244 17
1845 1262 6
1852 1269 33
1886 1303 4
1891 1308 167
2059 1476 2
2062 1479 27
2090 1507 32
2123 1540 22
2146 1563 78
2225 1642 4
2229 1647 3
2234 1651 12
2247 1664 3
2251 1668 8
2260 1677 22
2283 1700 4
2288 1705 57
2346 1763 1
2348 1765 5
2354 1771 5
2360 1777 1
2362 1778 1
2364 1781 18
2383 1800 23
2407 1824 11
2419 1835 1
2421 1838 4
2426 1843 18

2445 1862 2
2448 1865 37
2486 1903 39
2526 1943 26
2553 1970 6
2560 1977 67
2628 2045 24
2653 2070 16
2670 2087 18
2689 2106 19
2709 2126 3
2713 2130 1
2715 2132 31
2747 2164 13
2761 2178 3
2765 2182 9
2775 2192 15
2792 2209 41
2834 2251 17
2852 2269 13
2866 2283 17
2884 2301 5
2890 2307 1
2892 2308 1
2893 2310 13
2907 2324 50
2958 2375 149
3109 2525 3
3112 2529 5
3118 2535 27
3146 2563 27
3175 2592 10
3186 2603 7
3194 2611 4
3199 2619 2
3205 2622 2
3208 2625 8
3217 2634 9
3227 2644 28
3256 2673 22
3279 2696 23
3303 2720 6
3310 2727 7
3318 2735 28
3347 2764 11
3359 2776 14
3374 2791 23
3398 2815 17
3416 2833 6
3423 2840 7
3430 2848 2
3434 2851 9
3444 2861 4
3449 2866 29
3479 2896 5
3485 2902 3
3489 2906 8
3498 2915 4
3503 2920 2
3506 2923 5
3512 2929 20
3533 2950 3
3537 2954 4
3542 2959 8
3551 2968 1
3552 2971 4
3558 2975 34
3593 3010 5
3599 3016 2
3602 3019 3
3606 3023 28
3635 3052 47
3683 3100 9
3693 3109 1
3695 3112 5
3701 3118 5
3707 3124 26
3734 3151 8
3743 3160 8
3752 3169 2
3755 3172 5
3761 3178 5
3768 3185 19
3788 3205 2
3791 3208 5
3797 3214 6
3804 3221 13
3818 3235 9
3828 3245 2
3831 3248 8
3840 3257 8
3849 3266 22
3872 3289 23
3896 3313 12
3909 3326 1
3911 3328 5
3917 3334 2
3920 3337 2
3923 3340 17
3941 3358 2
3944 3361 5
3950 3367 2
3953 3370 14
3968 3385 8
3977 3394 16
3994 3411 27
4023 3440 17
4041 3458 7
4050 3466 1
4054 3469 3
4057 3474 18
4076 3493 17
4094 3510 1
4095 3512 1
4097 3514 21
4119 3536 19
4139 3556 8
4148 3565 11
4160 3577 46
4207 3624 3
4211 3628 2
4214 3631 6
4221 3638 12
4234 3651 15
4250 3667 35
4286 3703 27
4314 3730 2
4316 3733 3
4320 3737 20
4341 3758 10
4352 3769 103
4456 3873 26
4483 3900 9
4493 3910 17
4510 3928 2
4514 3931 7
4522 3939 5
4528 3945 14
4543 3960 11
4555 3972 23
4579 3996 2
4582 3999 11
4594 4011 3
4598 4015 1
4600 4017 16
4617 4034 3
4621 4038 2
4624 4041 11
4636 4053 8
4645 4062 2
4647 4065 3
4651 4068 5
4657 4074 8
4666 4083 2
4669 4086 8
4680 4094 3
4684 4101 13
4698 4115 5
4705 4122 59
4765 4182 5
4771 4188 2
4774 4191 2
4777 4194 32
4810 4227 5

4816 4233 5
4822 4239 17
4840 4257 2
4843 4259 1
4845 4262 9
4855 4272 2
4858 4275 2
4861 4278 15
4877 4294 7
4885 4302 2
4888 4305 5
4894 4311 14
4908 4326 2
4912 4329 5
4918 4335 7
4926 4343 12
4939 4356 2
4942 4359 8
4951 4368 5
4957 4374 11
4969 4386 5
4975 4392 3
4979 4396 13
4993 4410 12
5006 4423 16
5023 4440 11
5035 4452 14
5050 4467 5
5056 4473 2
5059 4476 29
5089 4506 5
5095 4512 2
5098 4515 20
5119 4536 2
5122 4539 26
5149 4566 39
5189 4606 4
5194 4611 3
5198 4615 34
5233 4650 5
5239 4656 5
5245 4662 5
5250 4668 2
5254 4671 8
5262 4680 1
5266 4683 2
5269 4686 20
5290 4707 31
5322 4739 6
5329 4746 14
5344 4761 32
5377 4794 8
5386 4803 11
5398 4815 20
5419 4836 2
5422 4839 20
5443 4860 8
5452 4869 9
5462 4879 10
5475 4889 3
5478 4893 1
5479 4896 5
5485 4902 8
5493 4911 2
5497 4914 31
5530 4947 28
5559 4976 5
5565 4982 21
5587 5004 76
5665 5082 23
5689 5106 15
5705 5122 39
5745 5163 1
5748 5165 74
5823 5240 18
5842 5259 51
5894 5311 5
5901 5317 10
5912 5328 7
5920 5336 26
5947 5363 18
5966 5382 7
5974 5390 17
5992 5408 8
6001 5417 5
6007 5423 11
6019 5434 1
6024 5435 4
6028 5444 5
6034 5450 11
6046 5462 8
6055 5471 14
6070 5486 2
6073 5489 20
6094 5510 23
6118 5534 2
6121 5537 2
6124 5540 23
6148 5564 9
6158 5573 1
6160 5576 5
6166 5582 11
6178 5594 5
6184 5600 2
6187 5603 26
6221 5630 4
6225 5636 1
6226 5642 11
6238 5654 2
6241 5657 2
6244 5660 8
6253 5669 5
6259 5675 2
6262 5678 5
6268 5684 2
6271 5687 32
6304 5719 3
6307 5723 8
6316 5732 2
6319 5735 38
6358 5774 2
6361 5777 5
6368 5784 11
6380 5796 4
6385 5801 5
6391 5807 2
6394 5810 17
6412 5828 11
6424 5840 8
6433 5849 14
6448 5864 36
6485 5901 13
6499 5915 26
6526 5942 6
6533 5948 1
6535 5951 5
6541 5957 2
6544 5960 11
6556 5972 14
6571 5987 5
6577 5993 11
6589 6005 8
6598 6014 6
6605 6021 13
6619 6035 2
6622 6038 5
6628 6044 38
6667 6083 5
6673 6089 11
6685 6101 29
6715 6131 5
6721 6137 5
6727 6143 8
6736 6152 56
6793 6209 14
6808 6224 5
6814 6230 26
6841 6257 5
6846 6263 2
6850 6266 17
6868 6284 11
6880 6296 5
6886 6302 26
6913 6329 5

6919 6335 14
6934 6350 2
6937 6353 8
6946 6362 17
6964 6380 2
6967 6383 9
6977 6393 14
6992 6408 10
7003 6419 11
7015 6431 8
7024 6440 5
7030 6446 11
7042 6458 2
7045 6461 20
7066 6482 29
7096 6512 17
7114 6530 17
7132 6548 11
7143 6560 3
7146 6564 1
7148 6565 1
7150 6566 5
7156 6572 5
7162 6578 8
7171 6587 5
7177 6593 11
7189 6605 5
7195 6611 2
7198 6614 8
7207 6623 2
7210 6626 47
7259 6675 10
7270 6686 11
7282 6698 1
7285 6699 4
7289 6705 1
7291 6707 41
7333 6749 5
7339 6754 2
7342 6758 23
7366 6782 6
7374 6790 9
7384 6800 88
7473 6889 26
7500 6916 22
7523 6939 12
7536 6952 28
7565 6981 3
7569 6985 3
7574 6988 2
7578 6994 41
7620 7036 23
7644 7060 5
7650 7066 1
7652 7068 6
7659 7075 2
7662 7078 2
7665 7081 11
7677 7093 2
7679 7096 4
7684 7100 1
7686 7102 8
7695 7111 8
7704 7120 2
7707 7123 40
7749 7165 2
7752 7168 8
7761 7177 50
7812 7228 2
7815 7231 5
7821 7237 8
7830 7246 2
7833 7249 29
7863 7279 6
7870 7286 4
7875 7291 2
7878 7294 8
7887 7303 5
7893 7309 5
7899 7315 2
7902 7318 11
7914 7330 17
7932 7348 23
7956 7372 11
7968 7384 11
7980 7396 2
7983 7399 2
7986 7402 17
8004 7420 2
8007 7423 5
8013 7429 2
8016 7432 5
8021 7438 1
8023 7439 11
8035 7451 25
8061 7477 1
8062 7479 3
8067 7483 14
8082 7498 12
8095 7511 10
8106 7522 14
8121 7537 23
8145 7561 8
8153 7570 2
8157 7573 11
8169 7585 8
8178 7594 26
8205 7621 2
8208 7624 20
8229 7644 3
8232 7648 20
8253 7669 8
8262 7678 2
8265 7681 10
8277 7693 9
8286 7703 3
8290 7707 5
8296 7713 15
8312 7729 32
8345 7762 3
8349 7766 35
8386 7803 7
8394 7811 26
8421 7838 8
8430 7847 3
8434 7851 22
8458 7875 4
8463 7880 5
8469 7886 5
8475 7892 11
8487 7904 17
8505 7922 2
8508 7925 2
8511 7928 31
8543 7960 12
8557 7974 19
8577 7994 16
8594 8011 11
8606 8023 9
8616 8033 2
8619 8036 7
8627 8044 2
8630 8047 3
8634 8051 22
8657 8074 20
8677 8095 1
8679 8096 4
8684 8101 14
8699 8116 3
8703 8120 1
8705 8122 7
8713 8130 24
8738 8155 27
8766 8183 7
8774 8191 1
8775 8208 3
8779 8211 1
8795 8212 8
8804 8221 38
8843 8260 6
8850 8267 7
8858 8275 8
8867 8283 1
8868 8285 1

8870 8287 13
8884 8301 5
8890 8307 27
8918 8335 38
8957 8374 8
8966 8383 2
8969 8386 24
8994 8410 1
8996 8413 8
9005 8422 56
9062 8479 6
9069 8486 7
9077 8494 2
9080 8497 2
9089 8499 5
9094 8511 3
9098 8515 2
9101 8518 17
9119 8536 11
9131 8548 5
9137 8554 5
9143 8560 26
9170 8587 11
9182 8599 49
9232 8649 2
9235 8652 41
9277 8694 11
9289 8706 11
9301 8718 5
9307 8724 3
9316 8727 7
9323 8735 1
9326 8736 2
9331 8738 4
9335 8752 3
9339 8756 7
9347 8764 7
9355 8772 6
9362 8778 1
9364 8781 8
9373 8790 5
9379 8796 2
9382 8799 8
9390 8808 1
9393 8810 7
9401 8818 28
9430 8847 11
9442 8859 2
9445 8862 5
9451 8868 8
9460 8877 2
9463 8880 26
9490 8907 11
9502 8919 5
9508 8924 3
9511 8928 29
9541 8958 8
9550 8967 17
9568 8985 2
9571 8988 5
9577 8994 35
9613 9030 26
9640 9057 2
9643 9060 5
9651 9065 4
9655 9072 2
9658 9074 1
9659 9076 10
9670 9087 35
9706 9123 3
9710 9127 4
9715 9132 5
9721 9138 26
9748 9165 2
9751 9168 5
9757 9174 2
9760 9176 5
9766 9183 11
9778 9195 20
9799 9216 26
9826 9243 14
9841 9258 21
9863 9280 25
9889 9306 17
9907 9324 5
9912 9330 1
9914 9331 19
9934 9351 2
9936 9354 2
9940 9357 8
9949 9366 2
9952 9369 11
9964 9381 8
9976 9389 3
9979 9396 8
9988 9405 19
10008 9425 62
10070 9488 2
10073 9490 12
10086 9503 3
10090 9507 5
10096 9513 6
10103 9520 7
10111 9528 8
10120 9537 5
10126 9543 18
10145 9562 16
10162 9579 11
10174 9591 2
10177 9594 32
10210 9627 14
10225 9642 3
10229 9646 7
10237 9654 2
10240 9657 6
10247 9664 10
10258 9675 17
10275 9693 1
10276 9699 5
10283 9704 1
10288 9705 14
10304 9721 1
10306 9723 5
10312 9729 11
10324 9741 1
10327 9744 17
10345 9762 1
10347 9763 1
10348 9765 14
10363 9780 2
10366 9783 5
10371 9792 3
10375 9795 1
10379 9796 10
10390 9807 9
10400 9817 22
10423 9840 56
10480 9897 5
10486 9903 62
10549 9966 5
10555 9972 32
10588 10005 8
10597 10014 5
10603 10020 2
10606 10023 14
10621 10038 2
10624 10041 23
10648 10065 2
10651 10068 2
10655 10072 16
10672 10089 14
10686 10107 3
10693 10110 5
10699 10116 8
10708 10125 8
10717 10134 12
10730 10147 46
10777 10194 4
10782 10199 20
10803 10220 6
10811 10226 3
10814 10230 1
10815 10232 5
10822 10237 2

10824 10241 23
10848 10265 14
10863 10280 4
10868 10285 2
10870 10288 4
10876 10293 19
10896 10313 14
10911 10328 10
10921 10340 1
10924 10341 4
10929 10346 12
10942 10359 7
10950 10367 12
10963 10380 5
10969 10386 1
10971 10388 5
10977 10394 17
10995 10412 8
11004 10421 6
11011 10428 6
11018 10435 33
11052 10469 2
11055 10472 5
11061 10478 5
11067 10484 24
11092 10509 1
11094 10511 8
11103 10520 5
11109 10526 39
11149 10566 19
11169 10585 2
11172 10589 5
11178 10595 20
11199 10616 17
11217 10634 17
11235 10652 20
11255 10673 1
11256 10675 3
11260 10678 1
11262 10679 23
11286 10703 2
11289 10706 3
11293 10710 10
11304 10721 2
11308 10725 17
11326 10743 28
11355 10772 5
11361 10778 2
11364 10781 5
11370 10787 2
11373 10790 5
11379 10796 5
11385 10802 15
11401 10818 31
11433 10850 2
11436 10853 2
11439 10856 8
11448 10865 8
11457 10874 15
11473 10890 4
11478 10895 8
11487 10904 20
11508 10925 11
11520 10937 8
11529 10946 8
11538 10955 8
11546 10965 3
11551 10968 1
11553 10970 3
11557 10974 7
11565 10982 5
11571 10988 20
11592 11009 33
11626 11043 5
11632 11049 65
11700 11114 3
11703 11120 20
11724 11141 12
11737 11154 3
11741 11158 18
11760 11177 2
11763 11180 8
11772 11189 17
11790 11207 5
11796 11213 32
11829 11246 11
11841 11258 2
11844 11261 21
11866 11283 7
11874 11291 2
11877 11294 5
11882 11302 4
11888 11306 3
11892 11309 8
11901 11318 12
11914 11331 1
11916 11333 16
11933 11350 12
11946 11363 6
11953 11370 11
11965 11382 10
11976 11393 14
11991 11408 17
12009 11426 23
12033 11450 1
12035 11452 6
12042 11459 23
12066 11483 5
12072 11489 2
12076 11493 16
12093 11510 5
12099 11516 11
12110 11528 2
12114 11531 2
12117 11534 5
12123 11540 2
12126 11543 2
12129 11546 2
12133 11550 4
12138 11555 52
12191 11608 26
12218 11635 1
12220 11637 15
12235 11654 2
12240 11657 6
12247 11664 38
12286 11703 61
12348 11765 1
12351 11768 4
12356 11773 9
12366 11783 2
12369 11786 4
12374 11791 6
12381 11798 16
12399 11814 1
12400 11817 1
12402 11819 5
12408 11825 34
12443 11860 2
12446 11863 9
12456 11873 16
12473 11890 2
12476 11893 2
12479 11896 2
12482 11899 20
12503 11920 15
12520 11937 24
12545 11962 17
12563 11980 8
12572 11989 5
12578 11995 11
12590 12007 10
12601 12018 3
12605 12022 5
12611 12028 6
12618 12035 13
12632 12049 20
12653 12070 9
12663 12080 10
12674 12091 5
12680 12097 2
12683 12100 2
12686 12103 11
12698 12114 1

12699 12116 4
12704 12121 2
12707 12124 11
12719 12136 9
12729 12146 4
12734 12151 23
12758 12175 14
12773 12190 20
12793 12211 2
12797 12214 5
12803 12220 2
12806 12223 5
12812 12229 9
12822 12239 22
12845 12262 6
12852 12269 7
12860 12277 11
12872 12289 5
12878 12295 5
12885 12302 4
12890 12307 3
12894 12311 11
12906 12323 22
12929 12346 12
12943 12360 3
12947 12364 4
12953 12368 1
12954 12371 5
12960 12377 2
12963 12379 1
12964 12381 3
12968 12385 2
12971 12388 12
12984 12401 28
13013 12430 3
13017 12434 4
13022 12439 2
13025 12442 32
13058 12475 5
13064 12481 2
13067 12484 23
13091 12508 5
13097 12514 15
13113 12530 14
13128 12545 13
13142 12559 4
13147 12564 18
13166 12583 2
13169 12586 6
13176 12593 16
13193 12610 2
13196 12613 20
13217 12634 26
13244 12661 17
13262 12679 29
13292 12709 26
13319 12736 8
13328 12745 8
13337 12754 20
13358 12775 2
13361 12778 8
13370 12787 5
13376 12793 2
13379 12796 8
13388 12805 2
13391 12808 5
13397 12814 5
13403 12820 18
13422 12839 16
13439 12856 26
13466 12883 23
13490 12907 28
13519 12936 15
13535 12952 47
13583 13000 5
13589 13006 14
13604 13021 5
13610 13027 8
13619 13036 2
13622 13039 29
13652 13069 5
13658 13075 2
13661 13078 14
13676 13093 8
13685 13102 2
13689 13106 4
13694 13111 14
13710 13128 2
13715 13132 6
13722 13139 4
13727 13144 17
13745 13162 9
13754 13172 5
13762 13179 3
13765 13201 6
13790 13207 3
13794 13211 7
13802 13219 8
13811 13229 2
13814 13231 2
13817 13234 4
13822 13239 30
13853 13270 10
13864 13281 3
13868 13285 2
13871 13288 19
13892 13309 6
13899 13316 1
13901 13318 5
13907 13324 2
13910 13327 19
13930 13347 3
13934 13351 1
13936 13353 7
13944 13361 1
13946 13363 23
13970 13387 11
13982 13399 8
13991 13408 2
13994 13411 8
14003 13420 11
14014 13432 3
14018 13435 24
14043 13460 11
14055 13472 1
14057 13474 5
14063 13480 1
14065 13482 21
14087 13504 2
14090 13507 8
14099 13516 11
14111 13528 8
14120 13537 6
14127 13544 5
14133 13550 41
14175 13592 5
14181 13598 20
14202 13619 11
14214 13631 8
14223 13640 17
14241 13658 2
14244 13661 31
14276 13693 5
14282 13699 6
14289 13706 17
14307 13724 5
14313 13730 2
14316 13733 8
14325 13742 16
14342 13759 10
14353 13770 4
14358 13775 5
14364 13781 1
14366 13783 3
14370 13787 17
14388 13805 1
14390 13808 2
14394 13811 11
14406 13823 5
14412 13829 47
14460 13877 2
14463 13880 8
14472 13889 5
14478 13895 8

14487 13904 8
14496 13913 17
14514 13931 8
14523 13940 15
14539 13956 6
14546 13963 3
14550 13967 2
14554 13971 7
14562 13979 2
14565 13982 5
14571 13988 10
14582 13999 18
14601 14018 32
14634 14051 2
14637 14054 7
14645 14062 9
14657 14072 3
14661 14078 8
14671 14088 85
14756 14174 1
14759 14176 15
14775 14192 17
14793 14210 8
14802 14219 26
14829 14246 11
14841 14258 14
14856 14273 2
14859 14276 11
14871 14288 5
14877 14294 5
14883 14300 9
14893 14310 10
14904 14321 32
14937 14354 2
14940 14357 2
14943 14360 14
14958 14375 11
14970 14387 5
14976 14393 2
14979 14396 14
14994 14411 5
15000 14417 2
15003 14420 14
15018 14435 26
15045 14462 5
15051 14468 11
15063 14480 12
15076 14492 1
15077 14494 15
15093 14510 12
15106 14523 4
15112 14529 4
15117 14534 14
15132 14549 2
15135 14552 14
15150 14567 14
15165 14582 8
15174 14591 18
15193 14610 4
15198 14615 5
15204 14621 11
15216 14633 2
15219 14636 8
15227 14649 3
15235 14652 2
15238 14655 25
15264 14680 3
15267 14684 5
15273 14690 11
15285 14702 5
15291 14708 5
15297 14714 14
15312 14729 2
15316 14733 8
15325 14742 2
15328 14745 4
15333 14750 2
15336 14753 11
15348 14765 8
15357 14774 8
15366 14783 23
15390 14807 5
15396 14813 5
15402 14819 11
15414 14831 17
15434 14851 3
15438 14855 6
15450 14861 6
15456 14869 1
15458 14870 5
15463 14880 27
15491 14908 12
15504 14921 8
15513 14930 11
15525 14942 6
15532 14949 10
15543 14960 8
15552 14969 2
15555 14972 5
15561 14978 5
15566 14984 3
15570 14987 5
15576 14993 11
15588 15005 9
15598 15015 4
15603 15020 20
15624 15041 6
15631 15048 7
15640 15055 1
15645 15056 6
15652 15069 2
15656 15073 6
15664 15081 7
15672 15089 1
15674 15091 40
15715 15131 1
15717 15134 5
15723 15140 12
15736 15153 13
15750 15167 35
15785 15203 2
15787 15207 1
15789 15209 6
15798 15215 6
15805 15222 11
15817 15234 33
15851 15267 1
15854 15271 3
15858 15275 5
15864 15281 14
15880 15297 4
15886 15303 4
15891 15308 24
15916 15333 15
15932 15349 8
15941 15357 3
15944 15361 26
15972 15389 6
15979 15396 6
15986 15403 57
16044 15461 4
16049 15466 9
16059 15476 5
16065 15482 2
16068 15485 4
16074 15491 3
16079 15495 46
16126 15542 4
16131 15547 8
16141 15555 1
16149 15556 4
16153 15564 1
16155 15570 12
16169 15583 3
16173 15588 11
16184 15600 2
16186 15604 1
16187 15606 1
16194 15608 15
16210 15624 1
16212 15626 3
16216 15630 8
16226 15638 2
16230 15640 1

16241 15641 6
16247 15661 3
16251 15665 5
16256 15672 1
16259 15673 8
16268 15682 5
16276 15690 10
16287 15701 5
16292 15707 1
16293 15709 1
16296 15710 5
16302 15716 4
16307 15721 12
16319 15734 2
16323 15737 3
16328 15742 26
16355 15769 4
16360 15774 4
16365 15779 5
16371 15785 5
16377 15790 1
16378 15792 14
16394 15808 7
16403 15817 38
16443 15855 26
16469 15882 1
16471 15883 43
16515 15927 10
16525 15945 4
16530 15950 2
16538 15952 40

Anexo 5:

Código fonte do programa de alinhamento de genes do mtDNA

```
/*
*
* UnitDNA.h
*
*/

//---------------------------------------------------------------------
#ifndef UnitDNAH
#define UnitDNAH

//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Dialogs.hpp>
#include <vcl\condefs.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream.h>
#include <math.h>
#include <fcntl.h>
#include <conio.h>
#include <iostream.h>
#include <assert.h>
#include <cassert>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TMemo *Memo1;
        TEdit *Edit2;
        TEdit *Edit3;
        TLabel *Label2;
        TLabel *Label3;
        TOpenDialog *OpenDialog1;
        TLabel *Label4;
        TEdit *Edit4;
        TLabel *Label5;
        TEdit *Edit5;
        TLabel *Label6;
        TGroupBox *GroupBox1;
        TButton *Button1;
        TButton *Button2;
        TGroupBox *GroupBox2;
        TEdit *Edit1;
        TLabel *Label1;
        TButton *Button3;
        TLabel *Label7;
        TGroupBox *GroupBox3;
        TRadioButton *RadioButton1;
        TRadioButton *RadioButton2;
        TGroupBox *GroupBox4;
```

```
        TRadioButton *RadioButton3;
        TRadioButton *RadioButton4;
        TComboBox *ComboBox1;
        TGroupBox *GroupBox5;
        TEdit *Edit6;
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall ComboBox1Change(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:    // User declarations
public:           // User declarations

        __fastcall TForm1(TComponent* Owner);

};
//---------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------
#endif



/*
*
* UnitDNA.cpp
*
*/

//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "UnitDNA.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------
const int N_GENES=38;

char * genes[N_GENES] = {"Control Region", "tRNA-Phe", "12S rRNA",
"tRNA-Val", "16S rRNA", "tRNA-Leu(UUR)", "ND1", "tRNA-Ile", "tRNA-Gln",
"tRNA-Met", "ND2", "tRNA-Trp", "tRNA-Ala", "tRNA-Asn", "tRNA-Cys", "tRNA-
Tyr", "COX1", "tRNA-Ser(UCN)", "tRNA-Asp", "COX2", "tRNA-Lys", "ATP8",
"ATP6", "COX3", "tRNA-Gly", "ND3", "tRNA-Arg", "ND4L", "ND4", "tRNA-His",
"tRNA-Ser(AGY)", "tRNA-Leu(CUN)", "ND5", "ND6", "tRNA-Glu", "CYTB", "tRNA-
Thr", "tRNA-Pro"};

char line[200], kind[50], location[100], location_f[100];
char accession[20], seq_gene[10000], seq_gene2[10000];
char prefix1[100], prefix2[100], prefix3[100], prefix4[100];
char s_aux[200], filename[100];;
char  a[20000], b[20000];

int n_gene=-1, num_seq=0;
int i, k, j, s, begin, end, start, num_c, ns;
int first_line1, last_line1, first_line2, last_line2;
int mm=0, t, tt, m1, indc=0, sss, sss1, ind=1, Paa1, Pbb1;
int N, M, ia, ib, Ai, Af, Bi, Bf, aii, bii;
int p1a=0, p1b=0, p2a=0, p2b=0;
int m, Paa, Pbb, Lcomab, stla, stlb, Position0=0;
```

```
int countt=0, S, gene1, unitc, maxLen;
int Ar[20000][2], Br[20000][2];
int UnitedA[20000][3];

FILE *fd;

char *fn, *begin_str, *end_str, *line2, *aa, *bb;


long horas;
bool file1, file2;
bool success = false, intire;

string first_line;
AnsiString out;

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

//---------------------------------------------------------------------
int position_split(char position[], int *end){
  int begin, tmp;

  begin_str=strtok(position,"..(");
  if(begin_str!=NULL && begin_str[0]=='>') begin_str = begin_str+1;
  begin=atoi(begin_str);

  if(begin==0){
    begin_str=strtok(NULL,"..(");
    if(begin_str!= NULL && begin_str[0]=='>') begin_str = begin_str+1;
    begin=atoi(begin_str);
  }

  end_str=strtok(NULL,"..(");
  if(end_str!=NULL && end_str[0]=='>') end_str = end_str+1;
  *end = atoi(end_str);
  M=*end;
  if(*end==0 || begin==0) return -1;
  return begin;
}

bool position_split_CR(char position[], char ab[]){
  int begin1, end1, begin2, end2, tmp;
  char *part1, *nums2, *nums1;

  if(strstr(location,"complement")!=NULL){
    part1=strtok(position,",");
    nums2=strtok(NULL,",");

    strtok(part1,"(");

    if(strstr(location,"complement")!=NULL){
      strtok(NULL,"(");
    }

    nums1=strtok(NULL,"(");
    begin1 =atoi(strtok(nums1,".."));
    end1 =atoi(strtok(NULL,".."));
    begin2 =atoi(strtok(nums2,".."));
    end2 =atoi(strtok(NULL,".."));
  }
```

```
if(begin1 < begin2){
  tmp=begin1;
  begin1=begin2;
  begin2=tmp;
  tmp=end1;
  end1=end2;
  end2=tmp;
}

//begin2..end2
first_line2 = ((begin2-1)/60)*60+1;
last_line2 = ((end2-1)/60)*60+1;

while (strcmp(kind,"ORIGIN")!=0 && !feof(fd)) {
  fgets(line, 200, fd);
  sscanf(line, "%s", kind);
}
if(feof(fd)) return false;

while (start!=first_line2){
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}

while(1){
  strtok(line," ");
  line2=strtok(NULL,"");
  for(j=0; line2[j]!='\0'; j++ ){
    if(isalpha(line2[j]) || line2[j]=='-'){
      seq_gene2[ns++]=line2[j];
    }
  }
  if(start==last_line2) break;
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}
seq_gene2[ns]='\0';

//end1..begin1
first_line1 = ((begin1-1)/60)*60+1;
last_line1 = ((end1-1)/60)*60+1;

while (start!=first_line1){
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}
ns=0;
while(1){
  strtok(line," ");
  line2=strtok(NULL,"");
  for(j=0; line2[j]!='\0'; j++ ){
    if(isalpha(line2[j]) || line2[j]=='-'){
      seq_gene[ns++]=line2[j];
    }
  }
  if(start==last_line1) break;
  fgets(line, 200, fd);
  sscanf(line, "%d", &start);
}
seq_gene[ns]='\0';

for(j=begin1-first_line1; j<=end1-first_line1; j++){
```

```
    ab[num_c++] = toupper(seq_gene[j]);
  }
  for(j=begin2-first_line2; j<=end2-first_line2; j++){
    ab[num_c++] = toupper(seq_gene2[j]);
  }

  ab[num_c]='\0';

  fclose(fd);

  return success;
}

//-------------------------------------------------------------
bool read_file_gene(char filename[], char ab[]){

  strcpy(prefix1,"/gene=");
  strcpy(prefix2,"/product=");
  strcpy(prefix3,"/gene=");
  strcpy(prefix4,"/product=");

  if(n_gene==0){//D-loop
    prefix1[0]='\0';
    prefix2[0]='\0';
    strcat(prefix1,"/note=\"control");
    strcat(prefix2,"/note=\"D-loop\"");
  } else {
    if(n_gene==17 || n_gene==30){ //tRNA-Ser
      strcat(prefix1, "\"tRNA-Ser\"");
      strcat(prefix2, "\"tRNA-Ser\"");
    } else if(n_gene==5 || n_gene==31){//tRNA-Leu
              strcat(prefix1, "\"tRNA-Leu\"");
              strcat(prefix2, "\"tRNA-Leu\"");

            } else {
              strcat(prefix1,"\"");
              strcat(prefix2,"\"");
              strcat(prefix1, genes[n_gene]);
              strcat(prefix2, genes[n_gene]);
              strcat(prefix1,"\"");
              strcat(prefix2,"\"");
            }
  }

  if(n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31){
    strcat(prefix3,"\"");
    strcat(prefix4,"\"");
    strcat(prefix3, genes[n_gene]);
    strcat(prefix4, genes[n_gene]);
    strcat(prefix3,"\"");
    strcat(prefix4,"\"");
  }

  kind[0]='\0';
  seq_gene[0]='\0';
  start=ns=num_c=begin=0;
  success = false;

  fd=fopen(filename,"r");

  while (strcmp(kind,"ORIGIN")!=0 && !feof(fd)) {

    fgets(line, 200, fd);
```

```
    sscanf(line, "%s %s", kind, location);

    if(strcmp(kind,"ACCESSION")==0){
      strcpy(accession, location);
      continue;
    }
    if(strcmp(kind, "tRNA")==0 || strcmp(kind, "gene")==0 || strcmp(kind,
"rRNA")==0){
      strcpy(location_f, location);
      continue;
    } else if(n_gene==0 && (strcmp(kind, "D-loop")==0 || strcmp(kind,
"misc_feature")==0)){
              success=true;
              break;
            }

    if(n_gene==2){//12S rRNA
      if(strcmp(kind, "/product=\"12S")==0 || strcmp(kind, "/product=\"s-
rRNA\"")==0){
        begin=position_split(location_f, &end);
        success=true;
        break;
      }
    } else if(n_gene==4){//16S rRNA
              if(strcmp(kind, "/product=\"16S")==0 || strcmp(kind,
"/product=\"l-rRNA\"")==0){
                begin=position_split(location_f, &end);
                success=true;
                break;
              }
            }

  if(n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31){
    if(strcmp(prefix3,kind)==0 || strcmp(prefix4,kind)==0){
      begin=position_split(location_f, &end);
      success=true;
      break;
    }
  }

  if(strcmp(prefix1,kind)==0 || strcmp(prefix2,kind)==0){
    if( n_gene==17 || n_gene==30 || n_gene==5 || n_gene==31 ){
      fgets(line, 200, fd);
      sscanf(line, "%s %*s %s",kind,location);
    }
    if(n_gene==17){ //tRNA-Ser UCN
      if(strcmp(location,"UCN\"")!=0) continue;
      } else if(n_gene==30){//tRNA-Ser AGY
         if(strcmp(location,"AGY\"" )!=0) continue;
      } else if(n_gene==5){//tRNA-Leu UUR
         if(strcmp(location,"UUR\"" )!=0) continue;
      } else if(n_gene==31){//tRNA-Leu CUN
         if(strcmp(location,"CUN\"" )!=0) continue;
      }
      begin=position_split(location_f, &end);
      success=true;
      break;
    }
  }//while end
  if(feof(fd)) return false;
  if(!success || begin==-1){
    return false;
  }
```

```
  if(n_gene==0){//D-loop
    if(strstr(location,"join")!=NULL){//join é substring de location
      return position_split_CR(location, ab);
    } else {
      begin=position_split(location, &end);
    }
  }

  first_line1 = ((begin-1)/60)*60+1;
  last_line1 = ((end-1)/60)*60+1;

  while (strcmp(kind,"ORIGIN")!=0 && !feof(fd)) {
    fgets(line, 200, fd);
    sscanf(line, "%s", kind);
  }
  if(feof(fd)) return false;

  while (start!=first_line1){
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }

  while(1){
    strtok(line," ");
    line2=strtok(NULL,"");
    for(j=0; line2[j]!='\0'; j++ )
      if(isalpha(line2[j]) || line2[j]=='-') seq_gene[ns++]=line2[j];

    if(start==last_line1) break;
    fgets(line, 200, fd);
    sscanf(line, "%d", &start);
  }
  seq_gene[ns]='\0';

  for(j=begin-first_line1; j<=end-first_line1; j++)
    ab[num_c++] = toupper(seq_gene[j]);

  ab[num_c]='\0';
  N=begin;
  fclose(fd);
  return success;
}

void bubblesort(int a[][2], int b[][2]){
  int i,j,tmp;
  for(i=0; i<countt; i++)
    for(j=0; j<countt; j++)
      if(a[i][0]<a[j][0]){
        tmp=a[i][0];
        a[i][0]=a[j][0];
        a[j][0]=tmp;
        tmp=a[i][1];
        a[i][1]=a[j][1];
        a[j][1]=tmp;
        //--b-
        tmp=b[i][0];
        b[i][0]=b[j][0];
        b[j][0]=tmp;
        tmp=b[i][1];
        b[i][1]=b[j][1];
        b[j][1]=tmp;
      }
}
```

```
int read_file_genbank(char filename[], char a[]){
  char kind[50], line[200];
  char *line2;
  FILE *fd;
  kind[0]='\0';
  int ns=0, j;

  fd=fopen(filename,"r");

  while (strcmp(kind,"ORIGIN")!=0) {
    fgets(line, 200, fd);
    sscanf(line, "%s", kind);
  }

  fgets(line, 200, fd);
  while (strstr(kind,"//")==NULL) {
    strtok(line," ");
    line2=strtok(NULL,"");
    for(j=0; line2[j]!='\0'; j++ )
      if(isalpha(line2[j]) || line2[j]=='-')
        a[ns++]=toupper(line2[j]);

    fgets(line, 200, fd);
    sscanf(line, "%s", kind);
  }
  a[ns]='\0';
  fclose(fd);
  return ns;
}

int read_file(char seq[], string filename){

  char c, filenm[200];
  int i_seq=0;
  ifstream fd;
  fd.open(filename.c_str());
  if(!fd){
    cerr << "Error: file " << filename <<" could not be opened.\n";
    exit(EXIT_FAILURE);
  }

  getline(fd,first_line);
  if(first_line[0]!='>'){
    fd.close();
    strcpy(filenm,filename.c_str());
    return read_file_genbank(filenm,seq);
  }

  while(!fd.eof()){
    fd >> c;
    if( isalpha(c) || c=='-') seq[i_seq++]=c;
  }

  seq[i_seq-1]='\0';
  fd.close();
  return i_seq-1;
}

//-----------------------------------------------------------------
int Find(int Na, int Nb, int Ma, int Mb){
  int mm=0, t, tt, m1, indc, sss, sss1;
  int Paa1, Pbb1;
```

```
  sss=stla+stlb;
  m1=m;
  if (Mb-Nb>=Ma-Na) mm=Ma-Na; else mm=Mb-Nb;
  if(mm<m) m=mm;

  a1: indc=0;
    t=Ma-Na-m;
    tt=Mb-Nb-m;

    for (int j=0; j<=t; j++){
      for (int i=0; i<=tt; i++){
      int ind=1;
      for (int k=0; k<m; k++) if(aa[j+k]!=bb[i+k]){ ind=0; continue; }
        if (ind==1) {
          indc=1;
          Paa1= j+Na; Pbb1= i+Nb; Lcomab=m;
          if( Paa1 > Pbb1 ) sss1=Paa1; else sss1=Pbb1;
          if( sss1 < sss ) { sss=sss1; Paa=Paa1; Pbb=Pbb1;}
        }
      }
    }

    if (indc==0) {
      m--;
      goto a1;
    }
  m=m1;
  return 0;
}

//---------------------------------------------------------------------
void FindNM(int Na, int Nb, int Ma, int Mb)
{

  aa = (char*) calloc(Ma-Na+1,sizeof(char));
  bb = (char*) calloc(Mb-Nb+1,sizeof(char));

  for (int ia=0; ia<Ma-Na+1;ia++)  aa[ia]=a[Na+ia];
  for (int ib=0; ib<Mb-Nb+1;ib++)  bb[ib]=b[Nb+ib];
  Find(Na,Nb,Ma,Mb);

  free (aa); aa = NULL;
  free (bb); bb = NULL;
  return;
}

//---------------------------------------------------------------------
int recursive_FindNM(int P1a, int P1b, int P2a, int P2b, int Com){
  int ai, af, bi, bf;

  if(Com < 1 || (P2a-P1a < 1) || (P2b-P1b < 1)) return 0;

  FindNM(P1a, P1b, P2a, P2b);
  ai=Paa; bi=Pbb;
  af=Paa+Lcomab; bf=Pbb+Lcomab;

  if(Lcomab < 1) return 0;
  Ar[countt][0]=ai;
  Ar[countt][1]=af;
  Br[countt][0]=bi;
  Br[countt][1]=bf;
  countt++;
  S+=Lcomab;
```

```
  Com=Lcomab;

  recursive_FindNM(P1a,P1b,ai,bi,Com-1);
  recursive_FindNM(af,bf,P2a,P2b,Com);
}

//---------------------------------------------------------------------
bool check_position(int ai, int bi, int af, int bf){
  if( ai >= af || bi >= bf ) return false;
  if( ai < Ai || ai >= Af || af <= Ai || af > Af) return false;
  if( bi < Bi || bi >= Bf || bf <= Bi || bf > Bf) return false;
  return true;
}

//---------------------------------------------------------------------
void manual_mode(int ai, int bi, int af, int bf){
  int c,k,k2,k1,soma, flag_cont=0,x1,x2;
  recursive_FindNM(ai,bi,af,bf,m);

  out = "Not aligned symbols: a= "+IntToStr((af-ai+1)-S)+"   b=
"+IntToStr((bf-bi+1)-S);
  out +="\n\n";
  bubblesort(Ar,Br);

  ofstream outfile("alignment.txt");
  outfile << "Not aligned symbols: a= " << (af-ai+1)-S << "  b= "<< (bf-
bi+1)-S<<endl<<endl;

  unitc=0;
  for(c=0; c < countt; ){
    flag_cont=0;
    soma=Ar[c][1]-Ar[c][0];
    for(k=0; k<10; k++){
      k1=c+k; k2=c+k+1;
      if(Ar[k2][0]-Ar[k1][0]==Ar[k1][1]-Ar[k1][0]){
        if(Br[k2][0]-Br[k1][0]==Br[k1][1]-Br[k1][0]){
          soma=soma+(Ar[k2][1]-Ar[k2][0]); flag_cont=1; continue;
        }
      }
      x1=Ai+Ar[c][0]; x2= Bi+Br[c][0];
      out += IntToStr(x1)+ " "+IntToStr(x2)+" "+ soma +"\n";
      outfile << x1 << " " << x2 << " " << soma << endl ;
      UnitedA[unitc][0]=Ar[c][0];
      UnitedA[unitc][1]=Br[c][0];
      UnitedA[unitc++][2]=soma;
      if(soma > maxLen) maxLen=soma;

      if(flag_cont) c=c+k+1; else c++;
      break;
    }
  }
  outfile.close();
}

//---------------------------------------------------------------------
void first_alignment(int ai, int bi, int af, int bf){
  char AA[20000], BB[20000];

  FindNM(ai,bi,af,bf);

  for (int i=Paa; i<stla;i++) AA[i-Paa]=a[i];
  for (int i=0; i<Paa;i++) AA[stla-Paa+i]=a[i];
  for(j=0; j<stla; j++) a[j]=AA[j];
```

```
  for (int i=Pbb; i<stlb;i++) BB[i-Pbb]=b[i];
  for (int i=0; i<Pbb;i++) BB[stlb-Pbb+i]=b[i];
  for(j=0; j<stlb; j++) b[j]=BB[j];

  Ar[countt][0]=0;
  Ar[countt][1]=Lcomab;
  Br[countt][0]=0;
  Br[countt][1]=Lcomab;
  countt++;
  S+=Lcomab;
}

//---------------------------------------------------------------------
void intire_seq(){
  int c, k, k2, k1, soma, flag_cont=0, x1, x2;
  out = "Not aligned symbols: a= "+IntToStr(stla-S)+"   b= "+IntToStr(stlb-
S);
  out +="\n\n";

  for (j=0; j<countt; j++){
    if(Ar[j][0]<stla-aii){ Ar[j][0]=Ar[j][0]+aii; Ar[j][1]=Ar[j][1]+aii; }
    else {Ar[j][0]=Ar[j][0]-(stla-aii); Ar[j][1]=Ar[j][1]-(stla-aii); }

    if(Br[j][0]<stlb-bii){ Br[j][0]=Br[j][0]+bii; Br[j][1]=Br[j][1]+bii; }
    else{ Br[j][0]=Br[j][0]-(stlb-bii); Br[j][1]=Br[j][1]-(stlb-bii); }
  }

  bubblesort(Ar,Br);

  ofstream outfile("alignment.txt");
  outfile << "Not aligned symbols: a= " << (stla)-S << "  b= "<< (stlb)-
S<<endl<<endl;

  unitc=0;
  for(c=0; c < countt; ){
    flag_cont=0;
    soma=Ar[c][1]-Ar[c][0];
    for(k=0; k<10; k++){
      k1=c+k; k2=c+k+1;
      if(Ar[k2][0]-Ar[k1][0]==Ar[k1][1]-Ar[k1][0]){
        if(Br[k2][0]-Br[k1][0]==Br[k1][1]-Br[k1][0]){
          soma=soma+(Ar[k2][1]-Ar[k2][0]); flag_cont=1; continue;
        }
      }
      x1=Ar[c][0]+1; x2=Br[c][0]+1;
      out += IntToStr(x1)+ " "+IntToStr(x2)+" "+ soma +"\n";
      outfile << x1 << " " << x2 << " " << soma << endl ;
      UnitedA[unitc][0]=Ar[c][0];
      UnitedA[unitc][1]=Br[c][0];
      UnitedA[unitc++][2]=soma;
      if(soma > maxLen) maxLen=soma;

      if(flag_cont) c=c+k+1; else c++;
      break;
    }
  }
  outfile.close();
}

//---------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)  //RUN
{
```

```
  countt=0; S=0; maxLen=1;
  if(file1){
    if(file2){
      if(RadioButton3->Checked){//automatic mode
        if(RadioButton1->Checked){//intire seq
          Label7->Caption="Calculating...";
          m=StrToInt(Edit6->Text);
          first_alignment(0,0,stla,stlb);
          aii=Paa; bii=Pbb;
          m=StrToInt(Edit1->Text);
          recursive_FindNM(Lcomab,Lcomab,stla,stlb,m);
          intire_seq();
          Memo1->Lines->Text = out;
          Label7->Caption="Done.";
      } else { //gene seq
          Label7->Caption="Calculating...";
          m=StrToInt(Edit1->Text);
          manual_mode(0,0,stla,stlb);
          Memo1->Lines->Text = out;
          Label7->Caption="Done.";
      }
    } else if(RadioButton4->Checked){//manual mode
             m=StrToInt(Edit1->Text);

             p1a=StrToInt(Edit2->Text);
             p1b=StrToInt(Edit4->Text);
             p2a=StrToInt(Edit3->Text);
             p2b=StrToInt(Edit5->Text);

             if(check_position(p1a,p1b,p2a,p2b)){
               Label7->Caption="Calculating...";
               manual_mode(p1a-Ai,p1b-Bi,p2a-Ai,p2b-Bi);
               Memo1->Lines->Text = out;
               Label7->Caption="Done.";
             } else Label7->Caption="The numbers are wrong.";
      } else Label7->Caption="You must select the mode.";
    } else Label7->Caption="You have to select File 2 sequence.";
 } else Label7->Caption="You have to select Files sequences.";
}

//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender) //File 1
{
  if(RadioButton2->Checked){//gene
    Label7->Caption="";
    if(n_gene!=-1){
      if(OpenDialog1->Execute()){
        sprintf(filename, "%s", OpenDialog1->FileName);
        file1=read_file_gene(filename,a);
        Ai=N; Af=M; stla=strlen(a);
        if(file1){
          file2=false;
          gene1=n_gene;
          intire=false;
        } else Label7->Caption="It was not possible to take information
from file.";
      }
    } else Label7->Caption="You have to choose a gene.";
  } else if(RadioButton1->Checked) {//intire seq
           Label7->Caption="";
           if(OpenDialog1->Execute()){
              sprintf(filename, "%s", OpenDialog1->FileName);
              stla=read_file(a,filename);
```

```
              file1=true;
              intire=true;
            }
          } else Label7->Caption="You must select a range.";
}

//---------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender) //File 2
{
  if(file1){
    if(RadioButton2->Checked){//gene
      if(intire==false){
        Label7->Caption="";
        if(n_gene==gene1){
          if(OpenDialog1->Execute()){
            sprintf(filename, "%s", OpenDialog1->FileName);
            file2=read_file_gene(filename,b);
            Bi=N; Bf=M; stlb=strlen(b);
            if(file2){
              gene1=-1;
            } else Label7->Caption="It was not possible to take information
from file.";
          }
        } else {
           out="The gene must be the same as the one taken from File 1: ";
           Label7->Caption=out+genes[gene1];
        }
      } else Label7->Caption="You must select intire sequence.";

    } else if(RadioButton1->Checked){
        Label7->Caption="";
        if(intire){
          if(OpenDialog1->Execute()){
            sprintf(filename, "%s", OpenDialog1->FileName);
            stlb=read_file(b,filename);
            file2=true;
          }
        }else {
           out="The gene must be the same as the one taken from File 1: ";
           Label7->Caption=out+genes[gene1];
        }
    } else Label7->Caption="You must select a range.";
  } else Label7->Caption="You have to select File 1 sequence first.";
}

//---------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  ComboBox1->ItemHeight=32;
  n_gene=ComboBox1->ItemIndex;
}

//---------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for(i=0; i < N_GENES; i++)
    ComboBox1->Items->Add(genes[i]);
  RadioButton1->Checked=true;
  RadioButton3->Checked=true;
}
//---------------------------------------------------------------------
```