Algoritmo de Análise de Falhas em Sequências de Bases Nitrogenadas e Sugestão de Correção

Tanira Campos Maidana[1], Sylvio André Garcia Vieira[2]

[1]Acadêmica do Curso de Ciência da Computação – Universidade Franciscana (UFN)
Caixa Postal 151– 97.010-032 – Santa Maria – RS – Brasil

[2]Docente dos Cursos de Ciência da Computação e Sistemas de Informação – Universidade Franciscana (UFN)

Caixa Postal 151– 97.010-032 – Santa Maria – RS – Brasil

maidana.tanira@unifra.edu.br, sylvio@unifra.br

Abstract. *The use of technologies in oncology and molecular biology are advanced, but the access to them has a high value, for study purposes is practically unfeasible. In this way, the proposal of this work is the development of a algorithm of analysis on nitrogen bases with correction suggestion and alignments. For this work purpose, the programming language used will be Python, It will also use some modules of the Biopython land Pandas libraries.*

Resumo. *A utilização de tecnologias na área da oncologia e biologia molecular estão avançadas, porém o acesso a elas possui um valor bem elevado, para fins de estudo é praticamente inviável. Dessa forma, a proposta desse trabalho é o desenvolvimento de um algoritmo de análise de falhas em bases nitrogenadas com sugestão de correção e alinhamentos. Para atingir esse objetivo, a linguagem de programação empregada é Python, utilizando também alguns módulos da biblioteca Biopython e Pandas.*

1. Introdução

A bioinformática se preocupa com a aquisição, armazenamento, processamento, distribuição, análise e interpretação das informações biológicas. Isto se dá por meio de aplicações e ferramentas procedentes da matemática, biologia e informática, com o objetivo de compreender o significado biológico de uma grande variedade de dados [Bibgen 2004]. A bioinformática pode atuar no campo da medicina, auxiliando no diagnóstico e no tratamento. Como também, na descoberta de uma mutação causal de uma doença. Mas para isso, é necessário reconhecer uma sequência, e poder compará-la com sequências da mesma natureza já tratadas e disponíveis em bancos de dados biológicos [Portal da Educação 2013]. Com isso, é possível identificar as mutações ocorridas em um organismo e descobrir se elas estão envolvidas com algum tipo doença ou síndrome.

De acordo com Mariano et al. (2013), atualmente, os laboratórios de biologia molecular efetuam análises de sequenciamento gênico por meio de *primers* (pequenos segmentos de ácidos nucléicos iniciadores) e associação, com o intuito de identificar sequências normais e sequências variantes. As sequências variantes identificam, ou

procuram identificar a presença de doenças genéticas hereditárias ou somáticas. Quando realizado o sequenciamento, há a possibilidade de ocorrerem ruídos, ou seja, pontos em que se encontra mais de uma base ou ainda algo que não representa base nenhuma. Assim, é necessária a identificação de qual base que deveria estar naquele local. Há grande dificuldade de fazer estas avaliações, pelo número de bases ser elevado, uma vez que um gene pequeno tem cerca de cinco mil pares de bases, podendo alcançar a centenas de milhares.

Como objetivo geral, este trabalho consiste no desenvolvimento de um algoritmo, que visa solucionar os problemas de ruídos em bases nitrogenadas por meio de análise de vizinhança e posteriormente alinhando o resultado com sequências da mesma natureza já tratadas, oriundas de banco de dados biológicos públicos.

Partindo do objetivo geral, podem-se elencar os objetivos específicos:

- ✓ Compreender conceitos da biologia molecular, funcionamento de equipamentos de sequenciamento NGS e suas metodologias particulares;
- ✓ Aprender a linguagem de programação Python;
- ✓ Pesquisar algoritmos ou métodos que sejam melhores aplicados no alinhamento das sequências;
- ✓ Desenvolver um algoritmo em linguagem Python que analisa uma sequência bruta examinando vizinhos das bases ruidosas;
- ✓ Recolher os resultados da analise e depositando-os em um *Data Frame* com o auxílio da biblioteca Pandas.
- ✓ Substituir as bases ruidosas por bases aceitáveis (sob avaliação do biólogo) gerando uma sequência tratada.
- ✓ Utilizar a biblioteca Biopython para abrir os arquivos das sequências e realizar o alinhamento.

2. Referencial Teórico

Nesta seção são apresentados conceitos relativos a área biológica, os ácidos nucleicos, aminoácidos, proteínas, síntese proteica, mutação, alinhamentos, banco de dados biológicos, nova geração de sequenciamento e ferramentas de bioinformática disponíveis. Na área da informática, as definições das ferramentas computacionais e Alfabeto IUPAC, que são necessários para a compreensão desse trabalho.

2.1. Ácidos Nucléicos

Os ácidos nucléicos são constituídos pela polimerização de unidades intituladas nucleotídeos. Cada nucleotídeo contém resíduos de: uma molécula de ácido fosfórico; uma pentose (açúcar, monossacarídeo de 5 carbonos) e uma base nitrogenadas púrica ou pirimídica, que variam de nucleotídeo a nucleotídeo [Junqueira e Carneiro 2012]. As bases púricas com maior número são a adenina e a guanina, em geral designadas pelas letras A e G. Em contrapartida, as bases pirimídicas são citosina, timina e uracila, indicadas pelas letras C, T e U, respectivamente.

Distingue-se dois tipos de ácidos nucléicos, o ácido desoxirribonucleico (DNA) e o ribonucleico (RNA). A pentose encontrada no DNA é a desoxirribose e no RNA é a

ribose. Esses ácidos também se diferenciam nas suas bases nitrogenadas, ambos podem ser combinados com A, C e G, mas a timina é exclusiva do DNA como a uracila é do RNA.

Em relação a estrutura, a molécula de DNA possui duas cadeias de nucleotídeos (dupla hélice), as bases unem-se por meio de ligações de hidrogênio. A função do desoxirribonucleico é comandar o funcionamento da célula e repassar informações genéticas para outras células. É nele que estão os genes, onde estão todas as heranças genéticas. Nas células eucariontes sua localização é no centro delas, mas há também fragmentos em outros lugares [Lopes 2004].

O RNA é formado a partir do DNA, possui uma única cadeia de nucleotídeo (fita simples) e participa principalmente do processo de síntese proteica. Há três gêneros: RNA mensageiro (RNAm), que por meio das suas bases, determina as posições dos aminoácidos nas proteínas; os RNA transportadores (RNAt) e o RNA ribossômico (RNAr) que estão relacionados na formação das proteínas.

2.2. Aminoácidos

Os aminoácidos são conhecidos também como monopeptídeos, que ligados entre si, formam uma cadeia denominada polipeptídeos, conhecida como proteína. Ademais, há vinte aminoácidos que podem participar da produção de proteínas.

2.3. Proteínas

As proteínas são compostos orgânicos, macromoléculas, e polipeptídeos. O funcionamento das células depende delas. Dessa forma, são fundamentais para o organismo. Elas possuem diversas funções para os seres vivos, principalmente estruturais, energética, defesa (anticorpos), enzimática, etc. Suas funcionalidades são de acordo com a sua forma espacial ou tridimensional a partir das ligações de hidrogênio. A mudança de temperatura e de pH (potencial hidrogeniônico), são os fenômenos que podem mudar o seu formato e consequentemente mudar sua função. Esse processo é conhecido como desnaturalização, é a quebra da ligação de hidrogênio [Lopes 2004].

2.4. Síntese Proteica

É o método natural pelo qual uma proteína é produzida. Nos seres eucariontes, como os seres humanos, ele é dividido em duas partes: a primeira é a transcrição e a segunda é a tradução ocorrem no interior da célula. A Figura 1 apresenta o esquema da síntese de uma proteína, que se inicia pela quebra da dupla hélice do DNA, iniciando a transcrição.

Esse processo é importantíssimo para o funcionamento do corpo humano. Caso alguma das etapas ocorra um erro, este é propagando podendo gerar algumas consequências sérias, com a produção errada de certas proteínas poderá ocorrer inibições de enzimas a mutações celulares malignas.

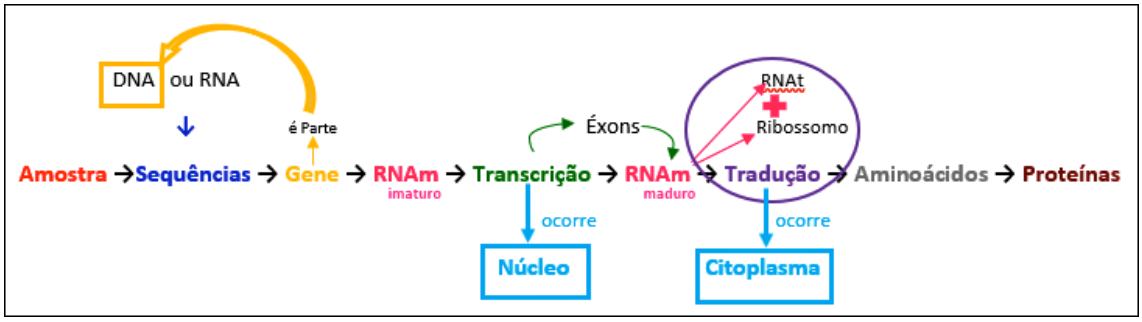


Figura 1. Esquematização de um processo de uma síntese proteica [Autor]

2.5. Mutação

Mutação é uma mudança que afeta a estrutura física de um cromossomo ou decorrente de uma alteração bioquímica em um gene [Terra 2014]. As mutações podem ser benéficas, como no caso para a seleção natural positiva determinante para evolução dos seres vivos, permitindo se adaptarem melhor a condições ambientais existentes. Ou mutações maléficas, em maioria, gerando doenças e tumores. A troca de um nucleotídeo dentro da região codificadora de um gene pode resultar na criação de um novo códon, que corresponderá a um aminoácido diferente no produto proteico [Zaha et. al 2014].

Algumas doenças humanas são causadas por alterações nos genes, não deixando as células executarem suas funções normalmente, como em casos pelo genótipo (efeito hereditário), ou fenótipo, efeito influenciado por fatores ambientais, como a exposição à radiação. Ambos muitas vezes estão ligados ao surgimento de cânceres. Além disso, a formação de células cancerígenas pode se dar com a participação de alguns vírus [Junqueira e Carneiro 2012].

Conforme Zaha et. al 2014, As mutações gênicas, conhecidas também como mutações pontuais podem envolver modificação em um único par de bases (substituição) ou em poucos pares de bases. Ainda podem ocorrer por adição ou deleção de bases no momento da duplicação do DNA. Há duas classificações para substituições, quando há uma troca entre bases da mesma família, como púricas com púricas, é chamado de substituição de transição, quando é por bases de família diferente, como púrica por pirimídicas é denominada substituição de transversão, geralmente mais perigosas.

2.6. Banco de Dados Biológicos

Conforme Portal da Educação (2013), os Bancos de Dados Biológicos (BDB) foram criados para armazenar essa quantidade imensa de dados moleculares com o intuito de integrar o maior número dessas informações por meio da Internet. Principalmente em BDB públicos, onde estas informações devem ser analisadas e corretamente interpretadas. Além de armazenar esse montante de informações, os BDB começam a desenvolver e disponibilizar programas de análises moleculares para facilitar o trabalho dos pesquisadores e suas interações.

Como a demanda aumentou nesse cenário científico mundial, necessitou-se de lugares confiáveis, com uma grande escalabilidade e de fácil acesso. Para isso, há o

Centro Nacional de Informações Biotecnológicas (NCBI — *National Center for Biotechnology Information*), localizado nos Estados Unidos da América e o Instituto de Bioinformática Europeu (EMBL-EBI — *European Bioinformatics Institute*) na Inglaterra. Interessante destacar, que esses servidores integram vários outros bancos importantes com dados específicos da biologia, como estrutura e função de genes e proteínas, por exemplo: o GenBank, SWISS-PROT, Omo, PIR, DDBJ, PDB, KEGG, etc.

2.7. Nova Geração de Sequenciamento

A Nova Geração de sequenciamento é baseada no sequenciamento de Sanger, um dos métodos, mais utilizados durante as últimas três décadas, porém era um trabalho muito árduo, já que com essa técnica era possível sequenciar fragmentos de 100 mil pares de base, sendo que um cromossomo humano possui 10^9pb. Por consequência disso, surgiu a Nova Geração de Sequenciamento (NGS – *Next-Generation Sequencing*), que reduziu os custos, ampliou drasticamente a velocidade de sequência, a partir do desenvolvimento de máquinas computadorizadas e outros tipos de método de sequenciamento, como Pirosequenciamento, Princípio de Sequenciamento por Síntese (*Sequencing By Snthesis* — SBS), SBS com detecção de íons, entre outros [Caten 2017]. Alguns exemplos dessas máquinas são a Ion Torrent, Illumina, Polonator, Roche, Oxford NanoporeMinIon, etc. Ressalta-se, que cada um desses equipamentos possui métodos, tamanho de leituras, volume de dados gerados e tempo de corrida distintos.

2.8. Alfabeto IUPAC

Em genética, o uso do termo amostra faz referência a sequências de nucleotídeos (DNA ou RNA) ou de proteínas, que são provenientes dos genes. A partir de sequências tratadas ou não, é possível retirar muitas informações importantes para o conhecimento do metabolismo de organismos.

As sequências brutas ou não tratadas são aquelas que provêm de equipamentos como NGS, podendo ser diretamente do sequenciador ou por meio de BDB, mas sem nenhuma análise. Geralmente estas sequências possuem ruídos, ou seja, bases nitrogenadas ambíguas ou faltantes. Isso ocorre, muitas vezes, porque a máquina no momento do sequenciamento os sensores do seu processador não conseguem captar algumas bases, ou no momento de converter os dados do sequenciamento em texto ocorre uma corrupção. Frisa-se que é aceito apenas as bases nitrogenadas púricas ou pirimídicas em uma sequência. Mas segundo a União Internacional de Química Pura e Aplicada (*International Union of Pure and Applied Chemistry* - IUPAC) foram encontras outras moléculas, derivadas de arranjos das bases púricas ou pirimídicas (Figura 2).

Porém essas demais moléculas estudadas pela IUPAC são consideradas ruídos, por serem equívocas, e dificultarem a análise e a compreensão de uma determinada sequência de bases nitrogenadas. Como mostrado na Figura 3 da Seção 2.8, que apresenta um fragmento de sequência de DNA com ruídos, representados pelas letras M e R, e o caractere -, destacadas em fundo branco.

Já as sequências aperfeiçoadas, são aquelas que passaram por uma vistoria e são tratadas suas bases faltantes ou ambíguas. Esse processo de aperfeiçoamento das sequências é caro e exaustivo, pois no mercado, os *softwares* encontrados que fazem esse processo são os embarcados nos sequenciadores de nova geração. Estes sequenciadores, por sua vez, são equipamentos de valores elevados. Outra forma é fazer o processo manualmente, por meio de planilhas eletrônicas ou programas que fazem uma leitura e visualização dos dados. Assim, com base nos dados contidos nos arquivos gerados pelos sequenciadores, pode-se atribuir novas bases que estão em falta ou são ambíguas. Este processo manual pode levar muito tempo.

IUPAC nucleotide code	Base
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
. or -	gap

Figura 2. Moléculas de Nucleotídeos aceitos pela IUPAC

Fonte: [Bioinformatics 2017]

2.9. Alinhamento

De acordo com Marco (2018), o alinhamento de sequências serve para organizar estruturas primárias de mesma natureza DNA, RNA e proteínas na busca de regiões análogas que possam ser oriundas de relações funcionais, estruturais ou evolutivas. Dessa forma, é possível reconhecer a evolução de determinada bactéria a partir da verificação das mutações em sua sequência.

Os alinhamentos podem ser realizados a partir de duas sequências, de forma global, na busca pelo melhor alinhamento, ocorrendo em toda extensão das sequências, ou de forma local, buscando somente regiões de alta similaridade (*hot spots*) entre elas, não considerando regiões adjacentes. As propriedades necessárias para realizar um

alinhamento são classificadas em pontualidades, penalidades e resultado (*score*) [Biopython 3].

As consideradas pontualidades são as combinações (*matches*) e as incompatibilidades (*mismatches*). Já as penalidades encontram-se as lacunas de abertura (*open-GAP*) e as lacunas de extensão (*extend-GAP*). Há muitas maneiras de calcular o *score*. Mas de forma simplificada, todas tendem para a Equação 1.

$$\textbf{Score} = \Sigma\ \textbf{(pontualidade)} - \Sigma\ \textbf{(penalidades)} \qquad (1)$$

Fonte: Elaborada pelo Autor

O valor associado a pontualidade *matches* são para bases que se correspondem, não há mudança desde a sua divergência. Ressalta-se que os valores de *matches* devem ser sempre positivos. No caso de *mismatches*, as bases são incompatíveis, houve uma substituição desde sua divergência. Já nas penalidades, um valor é atribuído ao o*pen-GAP* é quando há uma inserção ou deleção desde a divergência. Para isso insere-se uma lacuna. Valores de *Open-GAP* devem ser sempre negativos. Por fim, para *extend-GAP* é um valor inserido para lacunas após ser inicializada a lacuna *Open-GAP*. Aconselha-se para um melhor alinhamento inserir um valor mais baixo PARA *Extend-GAP* do valor de *GAP-Open*.

2.10. Ferramentas computacionais

Para desenvolver o algoritmo serão necessárias algumas ferramentas computacionais como o Python, que de acordo com Galvão (2017), é uma linguagem de programação interpretada, dinâmica e orientada à objetos. O Python permite que sejam incorporadas bibliotecas para que comandos mais específicos possam ser utilizados, como a *Biopython*, que é uma biblioteca voltada a computação biológica, escrita em Python [Biopython 2017]. Outra biblioteca utilizada neste trabalho é a Pandas, que fornece estruturas de alto desempenho e auxilia no trabalho com *Dataframes* [Pandas 2018].

Quando se trabalha com bioinformática, é comum a troca de dados entre sistemas diferentes e, por esta razão, faz-se o uso de padronização entre os formatos dos conteúdos dos dados contidos nos arquivos. Diferencia-se padrões dos dados dos arquivos com o uso de extensões que permitem a visualização prévia dos mesmos. As extensões mais utilizadas em bioinformática são o CSV (*Character-Separated Values*), que os dados são separados por vírgulas, o FASTA, cujo texto representa as sequências dos pares de bases, separadas em dois campos, o cabeçalho e a sequência em si [Pinto e Kremer 2017].

O formato GBK é utilizado em arquivos oriundos do banco de dados Genbank, contendo informações detalhadas de sequências de nucleotídeos, anotações biológicas e bibliográficas [Genebio 2018].

2.11. Ferramentas de bioinformática disponíveis

Algumas ferramentas desenvolvidas para apoiar os profissionais de bioinformática estão disponíveis para uso gratuito, como a BLAST (*Basic Local Alignment Search Tool*) que é desenvolvida e oferecida pelo NCBI. Ela utiliza algoritmos que permitem comparar uma determinada sequência com milhares de dados disponíveis em grandes bancos genéticos, fornecendo um valor de significância estatística associada a

comparação de similaridade. A BLAST pode ser usado para inferir relações funcionais e evolutivas entre as sequências, bem como ajudar a identificar membros de famílias gênicas [Feltes et al. 2014].

Outra ferramenta utilizada em bioinformática é o MEGA (*Molecular Evolutionary Genetics Analysis*), que é disponibilizada para sistemas operacionais Windows, Mac OS X e algumas versões do Linux. O MEGA analisa dezena de milhares de sequência de áreas como filogenéticas e fitomédicas e possui uma funcionalidade que permite construir gráficos e árvores evolutivas a partir de uma determinada sequência [Kumar et al. 2016]. Uma das suas aplicações que mais se destaca nesse *software* é a sua Interface Gráfica do Usuário (*Graphical User Interface* — GUI) por ser simples, de fácil visualização e localização das bases nitrogenadas, como mostra a Figura 3.

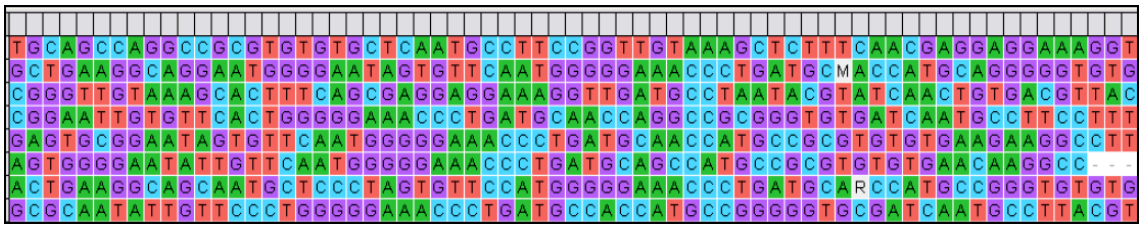


Figura 3. Tela do software MEGA 7 com um fragmento de sequência de DNA com ruídos

Fonte: Elaborada pelo Autor

3. Trabalhos Correlatos

Nesta seção, são descritos trabalhos com atributos similares ou que de alguma forma estão associados com a proposta do presente trabalho.

3.1. SIFT

O SIFT é uma das diversas ferramentas Web para analisar uma variedade de sequências. O método que utiliza para analisar essas sequências inteiras é o método matemático escore, sendo que quanto maior o valor do escore, maior o grau de homologia entre as sequências. Principalmente, atua no mecanismo de *splicing*, analisando os efeitos que as novas variantes possam ter sobre a estrutura proteica finalizada, ou seja, ele faz análise de possíveis substituições de aminoácidos, se podem ser danosos ou não para a estrutura [SIFT 2017].

A sua diferença com a proposta desse trabalho, é que serão analisados só com arquivos de sequências de genes brutas (sem nem um processo de aperfeiçoamento), ocorrerá uma busca para descobrir quais bases poderia ser substituída pelos ruídos, além da sugestão do IUPAC. Como será trabalhado com dados brutos a chance de acurácia é maior. Por fim, compará-la com outra sequência congênere curada e apresentá-las ao cientista com sugestão das possíveis trocas e alinhamentos com a sequência controle.

3.2. Detecção de Gene e Proteínas

Software desenvolvido por Machado et al. (2012) com o intuito de analisar amostras de sequências genéticas e proteicas específicas. Com a obtenção do seu sequenciamento, com a outra sequência da mesma natureza pré-definida em um banco de dados local,

para então identificar possíveis anomalias e suas futuras consequências em organismos de pessoas que possuem discordância nas suas amostras genéticas. Os pacientes cadastrados nesse software têm históricos de suas sequências genéticas em decorrência aos seus exames laboratoriais. Dessa forma, auxilia médicos a formularem um diagnóstico mais preciso.

Algumas diferenças desse software comparadas ao algoritmo desse projeto são: as amostras dos pacientes a serem analisadas devem ser digitadas, aliás, são segmentos de sequências, com isso não são sequências integrais.

3.3 FIFS: A data mining method for informative marker selection in high dimensional population genomic data

Esse método realizado por [Kavakiotis et al. 2017] utiliza uma técnica atual de mineração de conhecida como Seleção de Recursos por Item Frequente (FIFS - *Frequent Item Feature Selection*) para classificar um grande conjunto de dados genômicos de determinada população. É um método modular, que identifica e seleciona o genótipo mais frequente, a fim de criar subconjuntos polimorfismo de nucleotídeos simples (*Single Nucleotide Polymorphism* – SNPs). O método proposto foi aplicado em um conjunto de dados reais, de vários tipos de raças de porcos presentes na Grã-Bretanha. Essa pesquisa utiliza a técnica de mineração de dados associação, criando um próprio algoritmo que usufrui dos métodos matemático Delta e o de alinhamento Pairwise isso incentiva a melhorar este algoritmo futuramente.

4. Metodologia

O algoritmo desenvolvido nesse trabalho possui dois fluxos, como é possível visualizar na Figura 4, sendo o primeiro, como a opção 1 analisar uma sequência bruta de forma profunda na busca por ruídos e os substituí-los. Como segundo fluxo, a opção 2, um alinhamento entre a sequência de consulta com uma sequência controle. Para uma melhor compreensão do mecanismo deste algoritmo, há o Diagrama de Caso de Uso (Apêndice A) e de uma forma mais detalhada no Diagrama de Atividade (Apêndice B).

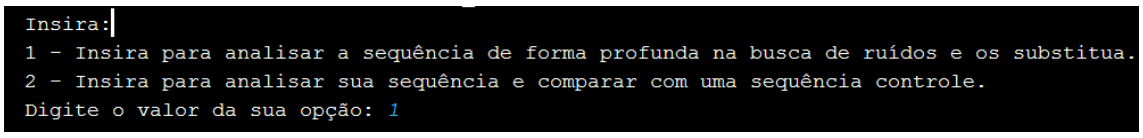


Figura 4 – Tela do terminal que apresenta o menu do algoritmo [Autor]

Os testes de verificação foram feitos com algumas sequências oriundas do BDB NCBI[1]. A sequência demonstrada com um exemplo, é uma sequência completa de DNA de uma cianobactéria Acaryochloris marina MBIC11017 contendo o plasmídeo pREB9[1] foi salva como arquivo com extensão fasta e genbank [NCBII 2008]. Essa sequência foi propositalmente alterada, inserindo nove ruídos (M, Y e 6 lacunas) ao longo de sua sequência. Estes ruídos com seus locais são visualizáveis na segunda linha da Figura 5.

[1] 1 - Disponível em: https://www.ncbi.nlm.nih.gov/nuccore/NC_009934.1,%20Fevereiro%202018.

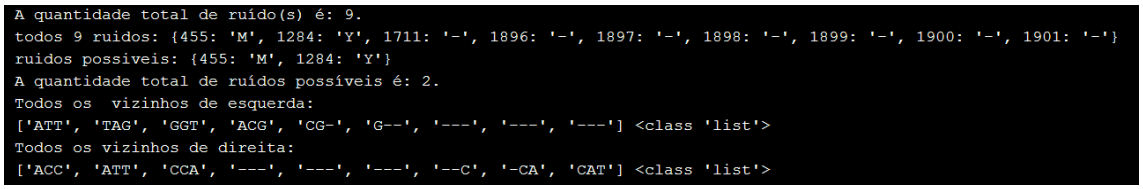


Figura 5 – Tela do terminal que apresenta alguns resultados da análise

Para o desenvolvimento deste trabalho foram verificados quais requisitos seriam necessários, e destacam-se o fato de que o algoritmo deve permitir que o usuário possa enviar arquivos de sequências de pares de bases com extensões .gbk ou .fasta. Esta funcionalidade foi desenvolvida fazendo uso da biblioteca *Biopython*, por meio da classe SeqIO pelo método *parse utilizado* no método determinaSequencia desse algoritmo, conforme apresentado nesse fragmento de código apresentado na Figura 6.

```
if extensao == 'fasta':
    for i in SeqIO.parse(nomeArquivo, extensao):
        sequencia = i.seq
        sequencia = sequencia.upper()
elif extensao == 'gbk':
    for i in SeqIO.parse(nomeArquivo, "genbank"):
        sequencia = i.seq
        sequencia = sequencia.upper()

else:
    sequencia = 'Advertência: Não foi inserido nome de arquivo válido,
ou a extensão. Impossível de ser aberto.'

return sequencia
```

Figura 6 – Fragmento de código de determinação da extensão do arquivo

Na Figura 6 pode-se observar a invocação do método SeqIO.parse, que recebe como parâmetro o nome do arquivo e a o nome da extensão desejada, e instancia o objeto i do tipo SeqRecord, atribuindo a sequência contida no arquivo para a variável do tipo String e a submetendo a ficar em maiúscula. Por fim, o retorno desse método determinaSequencia se dá pela variável sequência contendo unicamente a sequência do arquivo lido. Caso a extensão do arquivo não seja válida é atribuído uma mensagem de advertência como valor da variável sequência.

Quanto ao início da análise da sequência bruta, é dado após a detecção de algum ruído, caso seja encontrado, é solicitado ao usuário a quantidade de vizinhos que deseja analisar. Na busca por possíveis bases apropriadas para permutar com esses ruídos (Figura 7). Há grupos de bases que se repetem ao longo da sequência trazendo características específicas a amostra, com isso o algoritmo verifica os vizinhos do lado esquerdo e direito de cada ruído e os procura em toda a sequência de forma alinhada. Ressalta-se que ruídos sem ocorrência de vizinhos de bases aceitáveis e ruídos repetidos lado a lado não poderem ser analisados pelo algoritmo, na Figura 5 é possível visualizar quantidade total de vizinhos tanto do lado direito como esquerdo, todos os ruídos da sequência e os ruídos que podem ser analisados (ruídos possíveis).

Como o alfabeto IUPAC é fixo, ou seja, determinada base variante, conhecido como ruído, corresponde a um único conjunto de bases possíveis. Dessa forma, foi escolhida para armazenar esse alfabeto a estrutura de dados Dicionário, que são listas

mutáveis de associações compostas por uma chave, que deve ser única, e um valor correspondente [Mariano et al. 2016]. Na Figura 8 é possível visualizar no trecho do código no método alfabetoIUPAC, a verificação se um determinado ruído é reconhecido pelo alfabeto IUPAC (Figura 2), caso seja é retornado a sugestão das bases definida para aquele ruído, do contrário insere “sem sugestão”.

```
(vizinhosEsq, vizinhosDir) = defineVizinhos(sequencia, k, ruidos)
ruidosAninhados = organizaRuidos(sequencia, k, ruidos, vizinhosEsq,
vizinhosDir)
(df, ruidosPossiveis) = defineDataframe(ruidos, ruidosAninhados)
defineTabela(df, k, nomeArquivo)
print('ruidos: {}'.format(ruidos))
print('ruidos possiveis: {}'.format(ruidosPossiveis))
print(vizinhosEsq, type(vizinhosEsq))
print(vizinhosDir, type(vizinhosDir))
print(ruidosAninhados)
print('{}'.format(df))
print('{}'.format(df['base_modificada']))
```

Figura 7 – Fragmento de código do método da organização dos demais métodos da análise

```
alfabeto = {'R': 'A ou G', 'Y': 'C ou T', 'S': 'G ou C', 'W': 'A ou
T', 'K': 'G ou T', 'M': 'A ou C',
            'B': 'C ou G ou T', 'D': 'A ou G ou T', 'H': 'A ou C ou
T', 'V': 'A ou C ou G', '.': 'Nenhuma base',
            '-': 'Nenhuma Base', 'N': 'GAP'}
ruido = ruido.upper()
if ruido in alfabeto:
    return str(alfabeto[ruido])
return str('Sem Sugestao')
```

Figura 8 – Fragmento de código de determinação do alfabeto IUPAC

Para depositar todos os dados da análise, facilitar a criação de um arquivo com extensão CSV e possibilitar uma visualização desses dados de forma simples e de fácil entendimento, foi escolhido armazenar em uma *Data frame*. *Data frame* é uma estrutura bidimensional tabular mutável, potencialmente heterogênea, com eixos rotulados (linhas e colunas) [Pandas 2018]. Esta funcionalidade foi desenvolvida fazendo uso da biblioteca *Pandas,* por meio da classe *DataFrame*, conforme esse fragmento de código apresentado na Figura 9.

Na Figura 9 pode-se observar a criação da Data Frame, inserindo primeiro os nomes das colunas conteúdo a variável ruído, a localização da variável ruído, as bases aceitáveis, as localizações dessas bases, a quantidade dessas bases e a sugestão para o determinado ruído pelo alfabeto IUPAC(Figura 2). Como retorno desse método, é enviada uma tupla (similar à lista, porém é imutável) incluindo uma *Data Frame* (df) e a lista de ruídos, que contém os ruídos possíveis de serem analisados.

A geração de um arquivo com extensão CSV é visualizado no fragmento do código na Figura 10, que apresenta um fluxo, caso a opção do usuário seja o valor 1, é criado um nome para o arquivo formado pela palavra “Tabela_Detalhada”, mais o nome do arquivo da sequência bruta e a quantidade de vizinhos escolhidos (k) para análise.

Posteriormente, é criado o aquivo CSV a partir da Dataframe df. Se o valor da opção for diferente de 1, não será gerado o arquivo .csv.

```
 listaRuidos ={}
df = pd.DataFrame(columns=['ruido', 'local_ruido', 'base_modificada',
'localizacoes_base', 'quantidade', 'sugestao_IUPAC'])
for chave, ruido in ruidos.items():
    for local_ruido, local_base in quant.items():
        if local_ruido == chave:
            for base in local_base:
                df.loc[i, ['ruido']] = ruido
                df.loc[i, ['local_ruido']] = local_ruido
                df.loc[i, ['base_modificada']] = str(base)
                df.at[i, 'localizacoes_base'] = local_base[base]
                df.loc[i, ['quantidade']] = len(local_base[base])
                df.loc[i, ['sugestao_IUPAC']] =
str(alfabetoIUPAC(ruido))
                listaRuidos[int(local_ruido)] = str(ruido)
                i += 1
return (df, listaRuidos)
```

Figura 9 – Fragmento de código da criação de uma DataFrame [Autor]

```
if opcao == 1:
    nome =
'Tabela_Detalhada_'+nomeArquivo[0:int(nomeArquivo.index('.'))]+'_'+str
(k)+'vizinhos.csv'
    df.to_csv(nome, header=True, index=True, encoding='utf-8')
else:
    print('Não será gerado uma tabela detalhada.')
```

Figura 10– Fragmento de código de determinação a criação de um arquivo CSV

Com a visualização dos dados da DataFrame ou do arquivo CSV, o usuário insere novas bases no local dos ruídos possíveis (Figura 7). Com isso, será gerada uma nova sequência, visualizado no fragmento do código da Figura 11, onde é obtido as bases substitutas pelo método obtemNovasBases, criado uma nova sequência igual a sequência bruta, mas é modificado o seu local que contém ruídos trocado pelas bases substitutas.

```
bases = obtemNovasBases(ruidos)
novaSeq = []
novaSeq = list(sequencia)
for i in range(len(novaSeq)):
    for chave, base in bases.items():
        if chave == i:
            novaSeq[i] = base.upper()
            print(i, base)
novaSequencia = ''.join(novaSeq)
return novaSequencia.upper()
```

Figura 11 – Fragmento de código da criação de uma nova sequência

Após a criação da nova sequência, é possível criar um arquivo do tipo fasta contendo o cabeçalho advindo do arquivo da a sequência bruta, e o corpo do novo arquivo contendo a nova sequência já trata. O fragmento de código visualizado na Figura 12 apresenta a abertura de um arquivo saída onde é salvo primeiramente o

cabeçalho contendo o ID e a descrição em modo alteração. Posteriormente, é dada uma quebra de linha e inserido a nova sequência em fragmentos de 60 pares de base em cada linha, pois é formato mais frequente que as sequências são salvas em arquivos de extensão Fasta.

```
nome =
'ALTERACAO_'+nomeArq[0:int(nomeArq.index('.'))]+'_'+str(k)+'vizinhos'
nome = nome + '.fasta'
saida = open(nome, mode="a", encoding="utf-8")
saida.write('>'+id+' '+descricao)
saida.write('\n')
inicio = 0
fim = 60
for i in range(len(novaSequencia)):
    saida.write(novaSequencia[inicio:fim])
    saida.write('\n')
    inicio = fim
    fim += 60
saida.close()
```

Figura 12 – Fragmento de código da criação de um novo arquivo

O segundo fluxo do algoritmo constitui-se no alinhamento entre duas sequências, entre as sequências consulta e a controle. Antes de inicializar o alinhamento, é necessário que o usuário diminua o tamanho da sua sequência controle, também é possível diminuir sua sequência consulta. Essa redução e necessária, porque o local com melhor chance de conter regiões de alta similaridade está em pequenas frações de sequência de bases dentro da sequência [Zaha 2014].

Para realizar o alinhamento, esta funcionalidade foi desenvolvida fazendo uso da biblioteca *Biopython*, com o pacote Align, cria-se um objeto PairwiseAligner denominado Alinhador. Esse objeto armazena os valores das correspondências, incompatibilidade e as pontuações de lacuna. Foi criada uma estrutura dicionário denominada regra, contendo todas as possíveis combinações e incompatibilidades atribuindo determinando um valor para cada. Porém, o valor relacionado à pontuação de lacuna é necessário inserir separado a estrutura dicionário. O alinhador invoca o método substitution_matrix passando a regra. Após as inserções das pontuações o objeto Alinhamentos recebe do método align com o auxílio do objeto alinhador todos os melhores alinhamentos e as informações do alinhamento. Alinhamento passa por referência a sequência controle (novaSeqControle) e a sequência consulta (novaSeqBruta) ao método align. Por fim, são apresentados ao usuário todos os possíveis alinhamentos, as informações contendo a quantidade de melhores alinhamentos e o algoritmo escolhido para a execução desse alinhamento (mais detalhes podem ser vistos no Apêndice C).

Os seguintes valores atribuídos a estrutura regra, sendo 1 ponto para as combinações, há dois valores para incompatibilidade, considerando as substituições de transição valendo 0 e as substituições de transversão valendo -1, e atribuído -2 pontos para lacunas de abertura, não serão considerados valores para extensão de lacuna [Mariano et al. 2016].

O método substitution_matrix, diferente dos demais, possibilita atribuir regras específicas para cada propriedade do alinhamento, desconsiderando as pré-definidas já estabelecidas pelos métodos de pontuação, em muitos casos, não sendo aplicáveis há um

determinado problema [Biopython2 2018]. A classe PairwiseAligner seleciona automaticamente o algoritmo de alinhamento de acordo com as sequências recebidas. A classe trabalha com quatro tipos de algoritmos, sendo eles Needleman-Wunsch, Smith-Waterman, Gotoh ou Waterman-Smith-Beyer, suas características e diferenças podem ser visualizadas na Tabela 1. Essa escolha depende da quantidade das propriedades atribuídas. No caso desse trabalho, a escolha automática foi Gotoh [UNIFRIBURG 2018].

Tabela 1. Algoritmos utilizados no método substitution_matrix

	Alinhamento	Programação	Método	Pontuação Lacuna	Particularidades
Needleman-Wunsch	Global	Dinâmica	Matriz de Pontuação	Possui	A frequência de mutação para todas as bases é igual.
Smith-Waterman	Local	Dinâmica	Matriz de Pontuação	Possui	*Valor de terminação máximo para pontuação; *Complexidade O^2;
Gotoh	Global	Dinâmica	Matriz de Pontuação	Não Possui	*Penalidade diferente para buracos; *Valor de terminação canto inferior direito;
Waterman-Smith-Beyer	Local	Dinâmica	Matriz de Pontuação	Não Possui	* Complexidade O^3; *Penalidade diferente para buracos;

5. Resultados

O algoritmo realizou a análise dos dados oriundos do sequenciador, ainda sem tratamento, trazendo todos os ruídos da base de dados. Fez a varredura em toda a sequência procurando pelos fragmentos de bases chamadas de vizinhos da esquerda e da direita, localizando de acordo com a amostra fornecida, qual a frequência com que encontrou o mesmo fragmento de sequência e informado qual base estaria no local do ruído, podendo exibir esta informação diretamente no terminal ou gravando uma tabela no formato CSV.

O arquivo gerado pode ser visualizado na Figura 13, que se observa no conteúdo do arquivo a presença do ruído M sendo encontrado 5 vezes, duas sendo Timina (posições 810 e 1871) uma sendo Adenina(posição 1010) e mais duas vezes como Citosina (posições 1288 e 1450) e na última coluna a sugestão da IUPAC para o ruído, que poderia ser uma Adenina ou uma Citosina..

A	B	C	D	E	F	G
	ruido	local_ruido	base_modificada	localizacoes_base	quantidade	sugestao_IUPAC
1	M	455	T	[810, 1871]	2	A ou C
2	M	455	A	[1010]	1	A ou C
3	M	455	C	[1288, 1450]	2	A ou C
4	Y	1284	C	[640]	1	C ou T

Figura 13 – Tela do programa Excel que apresenta a Tabela Detalhada

Também como resultado do trabalho a sequência pode ser toda corrigida, substituindo o ruído pela base escolhida pelo usuário, que escolhe entre seguir a IUPAC ou as sugestões do algoritmo (Figura 14).

```
Substitua o ruido [M] no local[455]: T
Substitua o ruido [Y] no local[1284]: C
455 T
1284 C
Nova sequencia 2133
GCTAGATCCTGCTAACAGACCAGTGCCTGATTGCTGGATGAGGTTTCTTGTATTTGGCAAGATTGAAATAGGATGCGAATAGCAGCATAAAATTGATTACT
```

Figura 14 – Tela do terminal que apresenta a inserção das bases escolhidas

Outro resultado positivo do trabalho é a possibilidade escolher a posição inicial e final da sequência a ser analisada, que contribui consideravelmente com as pesquisas biológicas, visto que algumas das mutações pesquisadas estão em pontos específicos (hotspots).

O algoritmo permitiu a realização do alinhamento depois de que as bases ruidosas foram substituídas e os scores de semelhança são trazidos para o usuário, como se observa na Figura 15.

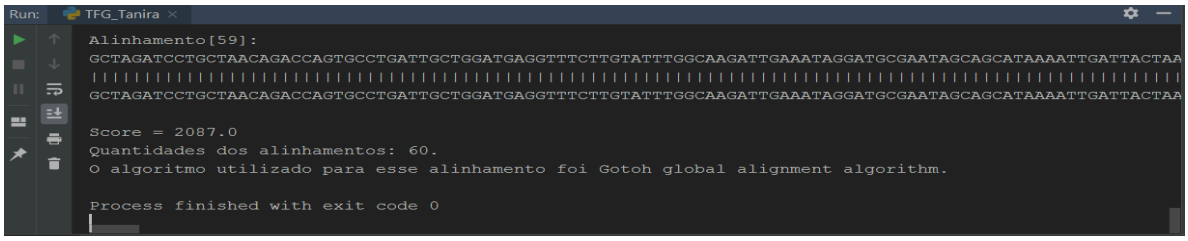


Figura 15 – Tela do terminal que mostra o alinhamento entre sequência tratada e controle bem como o score atingido

6. Considerações Finais e Trabalhos Futuros

A realização deste trabalho contribuiu de forma positiva com o conhecimento do aluno pois permitiu conhecer e ampliar estes conhecimentos referentes a biologia molecular. Também permitiu compreender como os equipamentos de sequenciamento funcionam e como fornecem os dados aos pesquisadores. Também permitiu o conhecimento da linguagem de programação Python, bem como suas bibliotecas Biopython e Pandas, que por serem necessárias ao desenvolvimento contribuíram para melhorar a visão do aluno sobre suas possibilidades de utilização.

A pesquisa dos algoritmos e dos métodos necessários ao trabalho permitiu que se compreendesse e aplicasse da melhor forma, produzindo os resultados almejados pelo presente estudo.

Ao realizar as funções de trocas e nova análise com sequências controle permitirá uma melhora na busca por novos pontos de mutação que venham a desencadear doenças genéticas.

Este trabalho, por ser pioneiro não somente na instituição, mas no mundo da bioinformática, precisa ainda de novos testes com novos organismos e doenças para comprovar a sua eficácia. Também pode servir de estímulo, para que outros alunos possam continuar já sabendo o que foi feito e o que foi possível desenvolver até aqui.

Referências

Bibgen (2004) “Introducción a la bioinformática”, /bvs.isciii.es/bibgen/actividades/curso_virtual/introduction/biinformatica.htm, Agosto 2017.

Bioinformatics (2017) “Screenshot das combinações de moléculas de nucleotídeos aceitas pelo International Union of Pure and Applied Chemistry (IUPAC)”, https://www.bioinformatics.org/sms/iupac.html, Agosto 2017.

Biopython (2017) “Biopython Introduction”, http://biopython.org/, Setembro 2017.

Biopython2 (2018) “Module pairwise2”, http://biopython.org/DIST/docs/api/Bio.pairwise2-module.html, Novembro 2018.

Biopython3(2018) “Biopython Cookbook - 6.5.3 Match and mismatch scores”, http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc69, Novembro 2018.

Buzolin et al. (2017) “Human Genomics 11:14 DOI 10.1186/s40246-017-0110-x”, http://pubmedcentralcanada.ca/pmcc/articles/PMC5485501/pdf/40246_2017_Article_110.pdf, Setembro 2017.

Caten, F. (2017) “Sequenciamento de DNA um guia completo”, http://marketing.neoprospecta.com/ebook-sequenciamento-dna, Outubro 2017.

Feltes et al. (2014) Bioinformática: da Biologia à Flexibilidade Molecular, Sociedade Brasileira de Bioquímica e Biologia Molecular – SBBq, 1ª edição.

Galvão F. (2017), Aprenda Python Básico - Rápido e Fácil de Entender, Felipe Galvão, 1ª edição.

Genebio (2018) “Genética e Bioinformática”, http://www.genebio.ufba.br/?page_id=303, Março 2018.

Junqueira, L. e Carneiro, J. (2012), Biologia Celular e Molecular, Guanabara Koogan, 9ª edição.

Kavakiotis et al. (2017) “FIFS: A data mining method for informative marker selection in high dimensional population genomic data”, Setembro 2017.

Kumar et al. (2016) “MEGA7: Molecular Evolutionary Genetics Analysis Version 7.0 for Bigger Datasets”, https://www.ncbi.nlm.nih.gov/pubmed/27004904, Novembro 2017.

Lopes, S. (2004), Bio – Volume Único, Saraiva S. A., 1ª edição.

Machado et al. (2012) “Detecção de Genes e Proteínas”, http://revistas.unifenas.br/index.php/RE3C/article/download/9/7, Agosto 2017.

Marco, R. (2018), “Alinhamento de Sequência”, http://www.ifsc.usp.br/~rdemarco/FFI0760/Alinhamento.pdf, Outubro 2018.

Mariano et al. (2016) “Introdução à Programação para bioinformática com Biopython”, 3ª Edição. Clube de Autores. Belo Horizonte-MG.

Pandas (2018), “Python Data Analysis Library”, https://pandas.pydata.org/, Outubro 2018.

Pinto L. e Kremer F. (2017) “Plataformas de Sequenciamento de Nova Geração & Pré-processamento de dados”, http://lvaruzza.com/files/apostila_bioinfo_2.0.1.pdf, Outubro 2017.

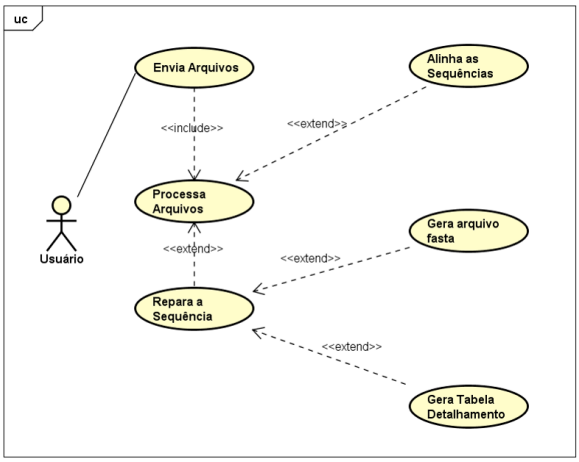
SIFT (2017) “SIFT Home”, http://sift.jcvi.org/, Outubro 2017.

Terra E. (2014) Dicionário da Língua Portuguesa, Rideel, 2ª edição.

UNIFRIBURG (2018) “Freiburg RNA Tools Teaching – Gotoh”, http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Gotoh, Outubro 2018.

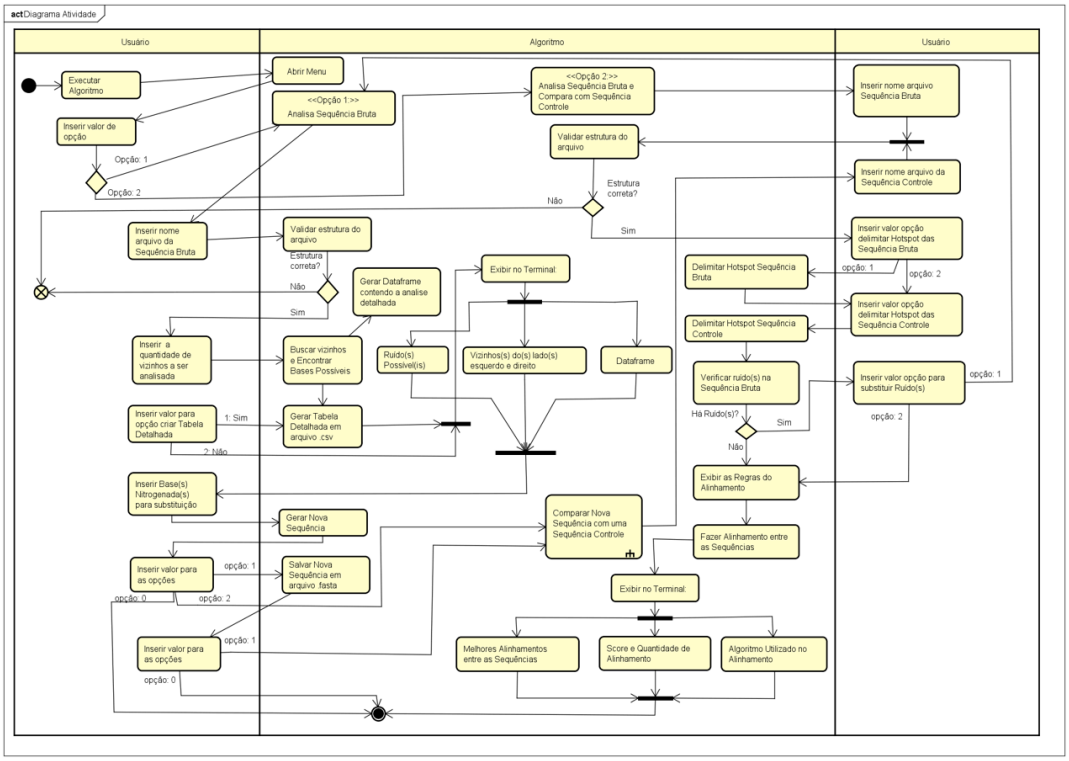
Apêndice A:

Diagrama de Caso de Uso



Apêndice B:

Diagrama de Atividade



Apêndice C:

Código do Alinhamento

```
alinhador = Align.PairwiseAligner()
print('Regras --> valor:\nBases da mesma família ex:(Pirimidicas x
Pirimidicas) = 0\n'
      'Bases de família ex:(Pirimidicas x Puricas): -1\n Match ex(A x
A): 1\nGAP ex.(-): -2')
#mismach por default eh 0
regra = {('A', 'C'): -1, ('A', 'T'): -1, ('A', 'U'): -1, ('A', 'G'):
0, ('G', 'C'): -1, ('G', 'T'): -1,
         ('G', 'U'): -1, ('G', 'A'): 0, ('C', 'A'): -1, ('C', 'G'): -
1, ('T', 'A'): -1, ('T', 'G'): -1,
         ('U', 'A'): -1, ('U', 'G'): -1, ('C', 'T'): 0, ('C', 'U'): 0,
('U', 'C'): 0, ('T', 'C'): 0,
         ('A', 'A'): 1, ('T', 'T'): 1, ('C', 'C'): 1, ('G', 'G'): 1,
('U', 'U'): 1}
alinhador.substitution_matrix = regra
alinhador.open_gap_score = -2
print(type(regra))
alinhamentos = alinhador.align(novaSeqControle, novaSeqBruta)


for i in range(len(alinhamentos)):
    print('Alinhamento[{}]:'.format(i))
    print(alinhamentos[i])

print("Score = %.1f" % alinhamentos.score)
print('Quantidades dos alinhamentos: {}.'.format(len(alinhamentos)))
print('O algoritmo utilizado para esse alinhamento foi
{}.'.format(alinhador.algorithm))
```