UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Eduardo Munari

# Migração para ambiente gráfico e refinamento no mecanismo de pré-processamento para limpeza de sequências em ferramenta localizadora de quasiespécies

São José do Rio Preto
2017

Eduardo Munari

# Migração para ambiente gráfico e refinamento no mecanismo de pré-processamento para limpeza de sequências em ferramenta localizadora de quasiespécies

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Orientador: Prof. Dr. Geraldo Francisco Donegá Zafalon

São José do Rio Preto
2017

Eduardo Munari

# Migração para ambiente gráfico e refinamento no mecanismo de pré-processamento para limpeza de sequências em ferramenta localizadora de quasiespécies

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Prof. Dr. Geraldo Francisco Donegá Zafalon

Eduardo Munari

Banca avaliadora:
Prof. Dr. Carlos Roberto Valêncio
Prof. Dr. Leandro Alves Neves

São José do Rio Preto
2017

# AGRADECIMENTOS

Agradeço primeiramente aos meu pais, por todo o esforço que fizeram para me dar todo o apoio possível durante todos os anos que se passaram enquanto eu estava na faculdade, tirando muitas vezes deles para dar para mim.

Agradeço, também, ao Prof. Geraldo e ao Anderson, por toda a paciência e apoio que me deram, por todos os conselhos e orientações, sempre acreditando no meu esforço e no meu trabalho.

Aos meus amigos, os quais não vou citar pois posso me esquecer de alguém, que foram parte importantíssima na minha vida na faculdade, que sempre me ajudaram nos mais diversos momentos, tenham sido eles de alegria ou de tristeza.

# RESUMO

No presente trabalho é apresentado a migração da ferramenta LOCQSPEC – Localizador de Quasiespécies do Laboratório GENOMA – Unesp/IBILCE – para uma interface gráfica mais eficiente. Atualmente acessada por linhas de comando, decidiu-se, então, criar uma interface gráfica melhorada para que se possa acoplar essa ferramenta ao conjunto de ferramentas existente no Laboratório. A linguagem C#, utilizada para o desenvolvimento de outras ferramentas do Laboratório de Bioinformática, foi escolhida como nova linguagem para a implementação dessa interface. Com isso, pretende-se contribuir para uma melhoria na usabilidade da ferramenta, tendo em vista a geral dificuldade, por parte dos biólogos, de se executar o programa por meio de instruções e de *scripts*. Para atingir essa melhoria focada na usabilidade, foram aplicadas também as Heurísticas de Jakob Nielsen, além de uma pesquisa com usuários externos para garantir a eficiência do método utilizado na interface gráfica criada. Além disso, foram implementadas melhorias no mecanismo de pré-processamento da LOCQSPEC, a fim de realizar uma limpeza mais eficiente nas sequências de entrada. Com isso, foi possível oferecer um resultado biologicamente mais significante para que os biólogos possam realizar análises mais precisas. A constituição desta interface e o refinamento do mecanismo de pré-processamento para limpeza de sequências destacam-se com contribuições reais do presente trabalho.

**Palavras-chave:** Bioinformática, Quasiespécies, Interfaces Gráficas, Usabilidade.

# ABSTRACT

The actual work presents the migration of the LOCQSPEC - Quasispecies Locator tool from the GENOMA - Unesp / IBILCE Laboratory to a more efficient graphical interface. Currently being accessed by command lines, it was decided to create an improved graphical interface so that it could couple this tool to the already existing toolset in the Laboratory. The C # language, which is already in use by the Bioinformatics Laboratory to develop another tools, was chosen as the new language for the implementation of this new interface. After applying these changes, it is intended to contribute to an improvement in the usability of the tool, in view of the general difficulty, on the part of Biologists, to run the program through instructions and scripts. In order to achieve this improvement focused on usability, we also applied the Jakob Nielsen Heuristics, as well as a search with external users to ensure the efficiency of the method used in the graphical interface created. In addition, improvements were made to the preprocessing mechanism of LOCQSPEC in order to perform a more efficient cleaning of the input sequences. Having this improvement done, it was possible to offer a more biologically significant result so that biologists can perform more accurate analyzes. The development of the graphical user interface and refinement of the mechanism for pre-processing in the sequence cleaning present as the main contribution of the work.

**Keywords:** Bioinformatics, Quasispecies, Graphical Interface, Usability.

*May the bridges I burn light the way.*

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE ABREVIAÇÕES

LOCQSPEC  Localizador de Quasiespécies

DNA Ácido desoxirribonucleico

RNA Ácido ribonucleico

CDC *Centers for Disease Control and Prevention*

HMM *Hidden Markov Models*

FFT *Fast Fourier Transformation*

IHC Interação Humano-Computador

IBILCE Instituto de Biociências, Letras e Ciências Exatas

# CAPÍTULO 1 – INTRODUÇÃO

## 1.1 Considerações iniciais

Na época de Teofrasto, aluno de Aristóteles, haviam 500 tipos de plantas catalogados. No século XVI, devido ao aumento das viagens dos europeus pelo mundo, Casper Bauhin observou mais de 6.000 tipos de plantas. No começo do século XVII, Carolus Linnaeus catalogou cerca de 18.000 espécies de plantas e 4.000 espécies de animais. No fim do mesmo século, o barão Cuvier relatou cerca de 50.000 espécies de plantas catalogadas (GIBAS; JAMBECK, 2001).

Em função desses avanços, evoluções consideráveis no aspecto cientifico possibilitaram o entendimento de diversos organismos. A grande quantidade de dados biológicos disponíveis tornou infactível a análise manual dessa quantidade de informações (ZAFALON et al., 2015).

Devido ao montante de informações e à preocupação dos biólogos em organizar, acessar e ampliar o número de espécies catalogadas, surgiram os primeiros problemas enfrentados pela informática na área biológica, que resultaram na Bioinformática. Porém, a bioinformática só aparece de fato no século passado, quando, depois de Watson e Crick desvendarem a estrutura química do DNA, tem-se o início do Projeto Genoma Humano (PROSDOCIMI, 2003) e, logo, surgiram os sequenciadores automáticos.

Em resposta ao aumento das pesquisas na área e, ainda, ao grande volume de dados produzidos, a localização de uma série de padrões em um banco de dados se torna um obstáculo à pesquisa. Atualmente, as buscas em bancos de dados são meramente procedimentos rotineiros, mas há ainda muito o que fazer para aumentar a precisão dos procedimentos em alinhamento de sequências, comparação de genomas, modelagem de estruturas proteicas, entre outras coisas (KATOH, 2016; WÅNGGREN, BILLETER, KEMP, 2016).

## 1.2 Objetivos

O presente trabalho tem como principal objetivo realizar a migração de toda a ferramenta LOCQSPEC para uma linguagem que suporte a construção de uma interface gráfica eficiente. Neste caso, essa migração foi realizada para a plataforma C#, devido a sua ampla flexibilidade e vasto montante de recursos visuais. Além disso, objetiva-se implementar melhorias no mecanismo de pré-processamento da LOCQSPEC, a fim de se realizar a limpeza nas sequências de entrada. Com isso, é possível oferecer um resultado biologicamente mais

significante para que os biólogos possam realizar análises mais precisas.

## 1.3 Justificativa

O projeto Genoma, e tantos outros, ganharam tamanha proporção e reconhecimento devido ao sucesso e à importância que alcançaram (PROSDOCIMI, 2003). A bioinformática tem papel muito importante, pois, sem o uso de técnicas computacionais, torna-se difícil realizar inferências sobre as biossequências (sequências de nucleotídeos ou aminoácidos) produzidas pelos sequenciadores e montadas pelos programas (EDGAR, 2004).

A fim de amenizar tais problemas e contribuir para o avanço científico biológico, surgiu o LOCQSPEC, que é um *software* para análise de *quasiespécies* (MARUCCI et al., 2008). Essa ferramenta utiliza o reconhecimento de padrões, que nada mais é do que a busca por conjuntos especiais de ácidos nucleicos ou de aminoácidos, em determinados conjuntos de sequências (LEMOS et al., 2003).

No entanto, atualmente a LOCQSPEC não oferece uma interface gráfica intuitiva e, muitas vezes, é executada por meio de instruções em linha de código e de *scripts*. Esse fato pode ser um agravante, visto que os usuários da ferramenta são majoritariamente biólogos, os quais apontam problemas de usabilidade (O'DONOGHUE et al., 2010).

Nesse contexto, a migração para uma interface agradável, intuitiva e funcional contribuirá para sua ampla utilização por parte de profissionais não ligados diretamente à área de computação. Além disso, foi possível integrar a ferramenta ao conjunto de soluções desenvolvido no laboratório de Bioinformática e que compõe um pacote de recursos para a área de Bioinformática.

Outros problemas que podem ser visualizados junto à LOCQSPEC é que, muitas vezes, as sequências dadas como entrada para o processamento contêm trechos de informações irrelevantes advindos da etapa de sequenciamento. O processamento dessas informações pode prejudicar a significância biológica dos resultados obtidos pela ferramenta, o que terá implicações diretas na qualidade da análise biológica. Assim, o refinamento da funcionalidade de limpeza das sequências será útil, de modo a permitir que avaliações mais precisas sejam realizadas e a qualidade dos resultados obtidos seja elevada (CHOU, 2001).

## 1.4 Metodologia

A primeira etapa deste trabalho envolve o estudo de técnicas computacionais voltadas aos alinhamentos de sequências, que processam grandes quantidades de dados e que têm bom desempenho. Após isso, será realizada a identificação das melhorias que podem ser adicionadas à LOCQSPEC, de acordo com as metodologias anteriormente estudadas.

Em seguida, foi realizada uma avaliação sobre quais melhorias em nível de interface de operação podem ser realizadas para atender às necessidades dos biólogos. Essa avaliação foi feita a partir de experiências obtidas em parceria com o CDC (*Centers for Disease Control and Prevention*) de Atlanta (EUA), com o qual o Laboratório de Bioinformática de que o candidato faz parte mantém um projeto em andamento. São verificadas principalmente melhorias de usabilidade e na simplicidade da apresentação dos resultados.

Após o estudo das técnicas que foram aplicadas, foram realizadas as migrações de todas as funcionalidades do LOCQSPEC, utilizando a linguagem de programação C#. Posteriormente, foi necessário o estudo do código fonte do LOCQSPEC para detectar em quais pontos da ferramenta serão anexadas as novas funcionalidades e, assim, realizar o acoplamento.

Como últimas etapas, foram feitos testes e eventuais correções no LOCQSPEC, além da sua documentação completa.

## 1.5 Organização da monografia

O trabalho está dividido em 5 capítulos.

O capítulo 1 está definido como a introdução, em que é apresentada uma visão geral de métodos, os objetivos do experimento e é introduzido o contexto.

O capítulo 2 apresenta a revisão bibliográfica, mostrando toda a base do trabalho e o estado da arte da área de Ciência da Computação, principalmente em Bioinformática.

No capítulo 3 define-se os procedimentos e as técnicas utilizadas, bem como a produção do projeto.

O capítulo 4 apresenta testes e mostra os resultados obtidos.

Por fim, o capítulo 5 apresenta a conclusão baseada nos resultados obtidos.

# CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA

## 2.1 Considerações Iniciais

São apresentados conteúdos necessários para o entendimento de todo o trabalho, desde conceitos da parte biológica, na seção 2.2, até conceitos da área computacional, apresentados nas seções seguintes.

## 2.2 Fundamentação Biológica

O trabalho em questão aborda um *software* de Localização de Quasiespécies, portanto a fundamentação biológica é importante antes de se voltar a atenção à parte computacional, para que seja possível entender como funcionam as sequências a serem analisadas.

### 2.2.1 A célula

A célula é a unidade estrutural dos seres vivos e pode ser analisada sozinha ou em conjunto com outras células, formando assim o ser vivo ou parte dele. A célula possui tudo o que é necessário para realizar as funções de um ser vivo, como nutrição, produção de energia e reprodução (ALBERTS et al., 2016).

Os organismos se equiparam a uma grande sociedade de células, que cooperam entre si, de modo a dividir as funções. Juntas, as células garantem a execução de inúmeras tarefas que são responsáveis por diversão funções, além da manutenção da vida.

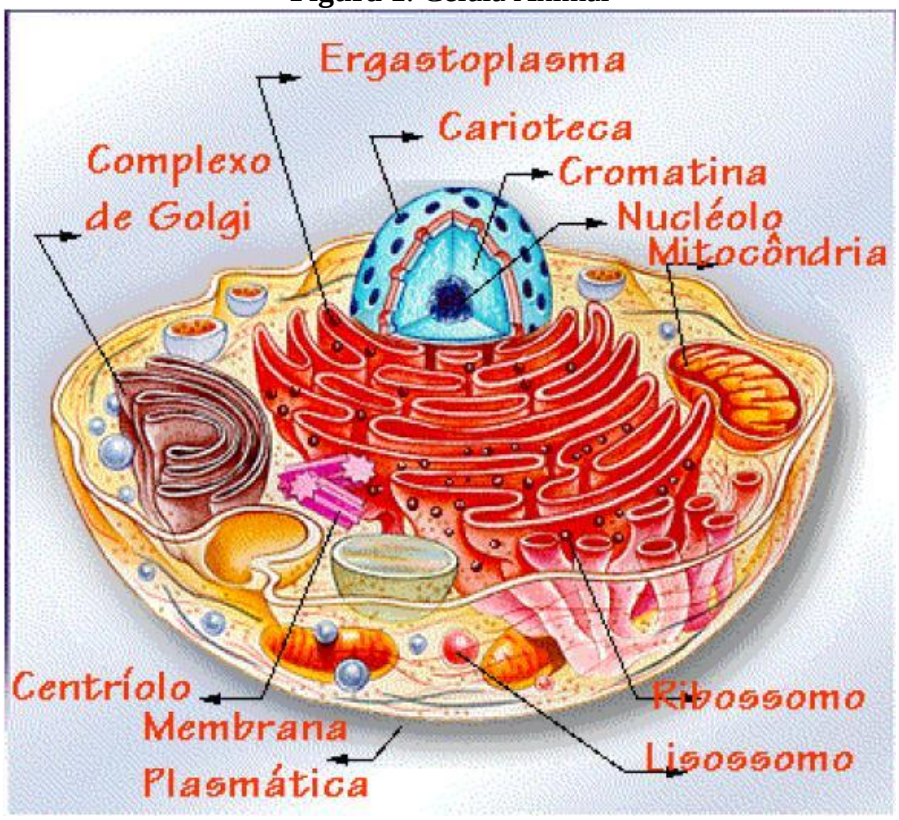
Cada célula possui uma função específica, e as células que formam o organismo da maioria dos seres vivos apresentam uma membrana que envolve seu núcleo, e são chamadas células eucarióticas (ALBERTS et al., 2016). As células eucarióticas possuem membrana celular, citoplasma e núcleo, conforme observa-se na Figura 1.

### 2.2.2 Ácidos nucleicos

Os ácidos nucleicos são moléculas formadas por nucleotídeos que possuem extensas cadeias carbônicas e que contêm um grupamento fosfórico (fosfato), uma base nitrogenada (purina ou pirimidina) e um glicídio (pentose), conforme pode ser observado na Figura 2. Nos eucariontes, localizam-se no núcleo das células e podem ser de dois tipos: ácido desoxirribonucleico (DNA) e ácido ribonucleico (RNA), de modo que ambos estão relacionados com controle metabólico celular e com transmissão hereditária (ALBERTS et al,

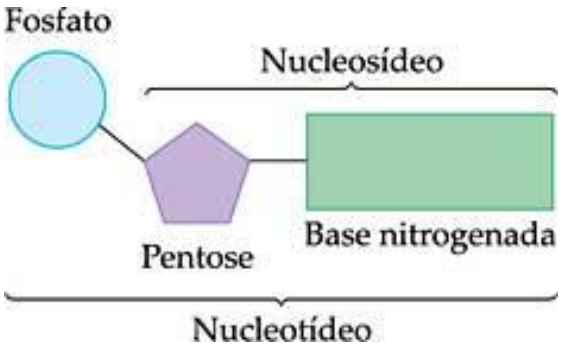
2016).

**Figura 1**. Célula Animal




Fonte: (LEMOS et al., 2003)

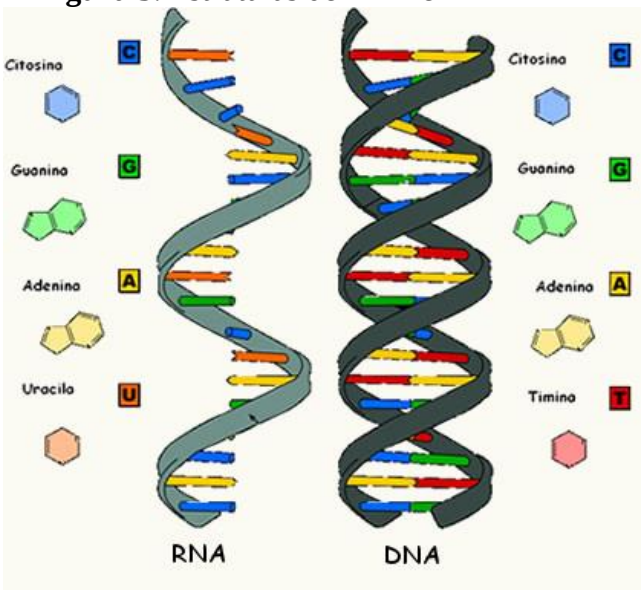
**Figura 2**. Ácido Nucleico




Fonte: *http://www.sobiologia.com.br/*

Além de diferenças específicas, como peso molecular, existem também diferenças estruturais entre as moléculas de DNA e RNA, conforme observa-se na Figura 3. Essas diferenças estão relacionadas com as bases nitrogenadas e a disposição dos nucleotídeos, que implicam na distância mantida entre os filamentos e a duplicidade das fitas contidas no DNA, diferenciando-se da unicidade do RNA (ALBERTS et al, 2016).

**Figura 3**. Estruturas de RNA e DNA




Fonte: www.alunosonline.uol.com.br/

### 2.2.3 DNA

O DNA é responsável por carregar características hereditárias. Assim, uma célula gerada possui características herdadas de sua célula geradora, o que ocorre através da divisão celular, quando parte (meiose) ou todo (mitose) o material genético é passado à célula filha. Apenas por causa dessa característica é que se tem a possibilidade de passar vantagens genéticas para futuras gerações, o que é um fato vital para a história evolutiva (ALBERTS et al, 2016).

Além disso, possui a forma de duas hélices entrelaçadas, o que vem a ser um fator essencial na sua habilidade de replicação, ou seja, sua reprodução, que consiste em gerar uma nova molécula de DNA.

### 2.2.4 RNA

O RNA constitui uma fita simples de ácidos nucleicos, que pode ser encontrada em três formas e tipos diferentes. Os RNAs trabalham em conjunto com os DNAs, de modo a ordenar funções moleculares e fazer a síntese de proteínas.

As informações que são codificadas no DNA são transcritas em RNA para sintetizar as proteínas. Após isso, ocorre o processo de tradução, em que o RNA construído é transformado em proteína. As proteínas, em si, são formadas por aminoácidos, constituídos por *códons*, que são nada mais que um trecho de RNA com uma sequência de três nucleotídeos.

### 2.2.5 Proteínas

As proteínas têm relação com quase tudo que ocorre nas células, afinal, o papel principal das proteínas é expressar a informação genética (ALBERTS et al., 2016).

Além disso, as proteínas também têm como função serem agentes estruturais das células, responsáveis pelo armazenamento, mobilidade, catalisação de funções biológicas, além de regular e defender o organismo, como os anticorpos. As proteínas estão entre as macromoléculas mais abundantes e versáteis, e são formadas pelo mesmo grupo de 20 aminoácidos. Relacionadas com o controle de todas funções celulares, em cada proteína existe um gene para codificá-la (ALBERTS et al, 2016).

### 2.2.6 Genoma e gene

A sequência completa de DNA que codifica um ser vivo é chamada de genoma (GIBAS et al., 2001), que consiste no conjunto de cromossomos de uma célula em um organismo (ALBERTS et al., 2016). Apesar disso, o genoma não trabalha de forma total, e é divido em partes definidas menores, chamadas de genes, que possuem função específica e exclusiva na produção de certa proteína.

O gene, por sua vez, é formado por sequências específicas de ácidos nucleicos, e é a unidade fundamental da hereditariedade. Além disso, localize-se intercalado com as sequências de DNA não codificado por proteínas, o designado “DNA inútil”, o que caracteriza uma parte não codificada (ALBERTS et al., 2016).

### 2.2.7 Quasiespécies

Quasiespécies consistem em populações que sofreram variabilidades genéticas, definindo sua classificação em classes, genótipos, subtipos, isolados e quasiespécies. O vírus é bastante usado como exemplo quando se fala de quasiespécies, pois sua variabilidade genética tem consequência direta em seu comportamento (MENG et al., 2016).

O vírus da hepatite C (HCV), maior causados de mortes do mundo é um exemplo de quasiespécie. As quasiespécies desse vírus circulam pelo corpo do infectado como populações diferentes, porém estreitamente relacionadas (MENG et al., 2016).

A análise dessas quasiespécies não é viável sem a bioinformática. O vírus da HCV, por exemplo, possui diferença média de nucleotídeos no RNA de 20% a 30% em genótipos, 10% a 20% em subtipos e menos de 10% nas quasiespécies (MENG et al., 2016).

## 2.3 Alinhamentos de sequências

O alinhamento de sequências consiste, basicamente, em determinar o grau de similaridades entre sequências, podendo ser duas ou mais. Fazer um alinhamento consiste em encontrar trechos semelhantes nas sequências de entrada, de modo a se buscar por uma configuração que reflita o maior nível de similaridade possível entre as sequências envolvidas (LEMOS et al., 2003).

Alinhamentos podem ser classificados a partir de seus tipos, os quais podem ser: par-a-par ou múltiplo, global ou local, determinístico ou probabilístico. Na análise mais simples, de apenas duas sequências, o alinhamento é chamado de par-a-par. Analogamente, ao alinharmos mais de duas sequências, usamos o alinhamento múltiplo. Ao alinharmos cadeias inteiras, utilizamos o chamado alinhamento global, portanto, ao alinharmos partes das sequências utilizamos o alinhamento local.

No alinhamento determinístico, sempre se encontra o melhor alinhamento possível para dadas sequências, no entanto, devido ao custo computacional dessa estratégia, geralmente o tempo de execução é alto para conjuntos com mais de três sequências, o que, geralmente, torna o processo infactível (LEMOS et al., 2003). Por outro lado, alinhamentos probabilísticos utilizam diferentes heurísticas, como Algoritmos Genéticos, Alinhamento Progressivo, entre outras, de modo que se viabiliza a execução de alinhamentos de múltiplas sequências em um tempo de execução factível e que produz resultados com boa significância biológica.

Basicamente, no alinhamento de duas sequências *a* e *b*, deve-se (EIDHAMMER et al., 2004):

a) parear todos os resíduos na mesma ordem em que aparecem na sequência original.

b) é permitido alinhar um caractere de *a* com um de *b*.

c) caso haja algum espaço (*gap*, representado por '-') nas sequências, este pode ser alinhado com qualquer resíduo da outra sequência.

d) dois *gaps* não podem ser alinhados.

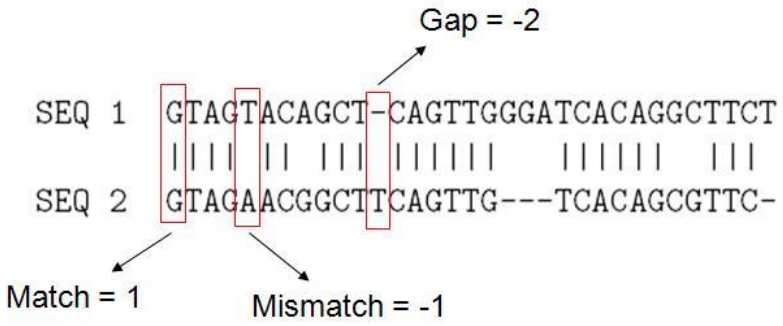
## 2.4 O alinhamento simples

A função do algoritmo consiste em, dadas duas sequências, computar o melhor alinhamento entre elas.

Um alinhamento é definido como a inserção de buracos em pontos arbitrários ao longo das sequências de modo que elas fiquem com o mesmo comprimento. Não é permitido que um *gap* esteja alinhado com outro *gap* (PROSDOCIMI et al., 2003).

Dado um alinhamento, deve-se atribuir uma pontuação, conforme nota-se na Figura 4. A pontuação total do alinhamento será a soma das pontuações das colunas. Se a coluna possui bases iguais, recebe 1 ponto; se possui bases diferentes, -1; e se possuir um buraco e uma base, -2 pontos. O alinhamento ótimo é o que possui maior pontuação, e esse fato é chamado de similaridade.

**Figura 4.** Alinhamento simples




Fonte: (PROSDOCIMI et al., 2003)

Caracteres diferentes (*mismatches*) e espaços (*gaps)* dispostos a fim de trazerem uma maior quantidade de similaridades (*matches*) consistem em um alinhamento ótimo.

## 2.5 O alinhamento múltiplo

O alinhamento múltiplo de sequências é a comparação simultânea de um conjunto de sequências, e é utilizado na identificação de regiões conservadas, no sentido de examinar similaridades globais e relações evolucionárias entre um grupo de biossequências (KANEHISA, 2000; BAWONO et al., 2017).

O método de alinhamento par-a-par é muito eficiente e preciso, no entanto, em problemas reais, em que geralmente é necessário o processamento de grandes conjuntos de sequências, o seu comportamento extrapola os tempos hábeis de processamento, então, para tal processamento, surgiram heurísticas chamadas de algoritmos de alinhamento múltiplo de sequências (ZAFALON, 2009).

Para realização do alinhamento múltiplo, recorremos a algoritmos de aproximação de otimização combinatória, e os dividimos em dois grupos: progressivos e iterativos.

Iterativo: as sequências são separadas em subgrupos, que são alinhados repetidamente e, em seguida, tais subgrupos são alinhados globalmente com todas as sequências (ZAFALON, 2009).

Progressivos: é usada a técnica de programação dinâmica, na qual alinha-se primeiro às sequências mais correlatas e, então, às menos correlatas (SIEVERS; HIGGINS, 2014).

Os mais utilizados são os progressivos, em função de sua relativa simplicidade e dos bons resultados obtidos.

## 2.6 Reconhecimento de padrões

O objetivo do reconhecimento de padrões é classificar objetos em categorias ou em classes (LEMOS et al., 2003). Em função do surgimento dos computadores, a demanda por aplicações práticas capazes de reconhecer padrões aumentou. Reconhecer padrões é fundamental em todos sistemas que envolvem a tomada de decisões.

Um sistema de reconhecimento de padrões consiste em observações a serem classificadas ou descritas, um método de extração de características e um modo de classificá-las após o término das fases. O esquema de classificação é baseado em um conjunto de padrões previamente designado, chamado de conjunto de treinamento, cujo resultado é caracterizado como supervisionado (LEMOS et al., 2003). Há também o aprendizado não supervisionado, no qual não ocorre o contato com o conjunto de treinamento, estabelecendo, então, sua análise através de padrões estatísticos (LEMOS et al., 2003).

Enumerar todos os padrões que satisfaçam as restrições e atribuir características probabilísticas para a aparição de cada um deles é útil apenas para padrões pequenos e simples, pois seu tempo de execução aumenta exponencialmente de acordo com o comprimento do padrão (LEMOS et al., 2003). O tempo de execução cresce linearmente ao tamanho da sequência de entrada. A vantagem é a garantia da descoberta do melhor padrão, e podem ser permitidos descasamentos, inserções e remoções (BREJOVA et al., 2000).

Pode ser usado, também, como padrão de classificação de proteínas, por exemplo, relacionando-as com padrões de famílias de proteínas já existentes (COUTO; ZIPFEL, 2016).

### 2.6.1 Enumeração de padrões com buracos

Em alguns casos, é melhor procurar padrões que contenham *gaps*. O sistema MOTIF faz esse trabalho. Ele descobre padrões com 3 aminoácidos separados por 2 buracos fixos, nos quais os buracos podem ter tamanhos de *0* a *d*, em que *d* é uma constante definida pelo usuário. Não se permite *mismatches*, mas não se exige que o padrão ocorra em todas as sequências. Assim, remove-se todos os padrões que ocorrem próximos, depois todos os descasamentos de um padrão são alinhados e, baseado no consenso das colunas desse alinhamento, o padrão é

estendido (LEMOS et al., 2003).

### 2.6.2 Enumeração de padrões com podas

A principal vantagem do método de padrões com podas é a busca de padrões maiores e mais complexos do que as buscas exaustivas simples. Exemplos dessa estratégia são o algoritmo *Pratt* (LEMOS et al., 2003), WINNOVER (LEMOS et al., 2003) e a fase de varredura do algoritmo TEIRESIAS (LEMOS et al., 2003).

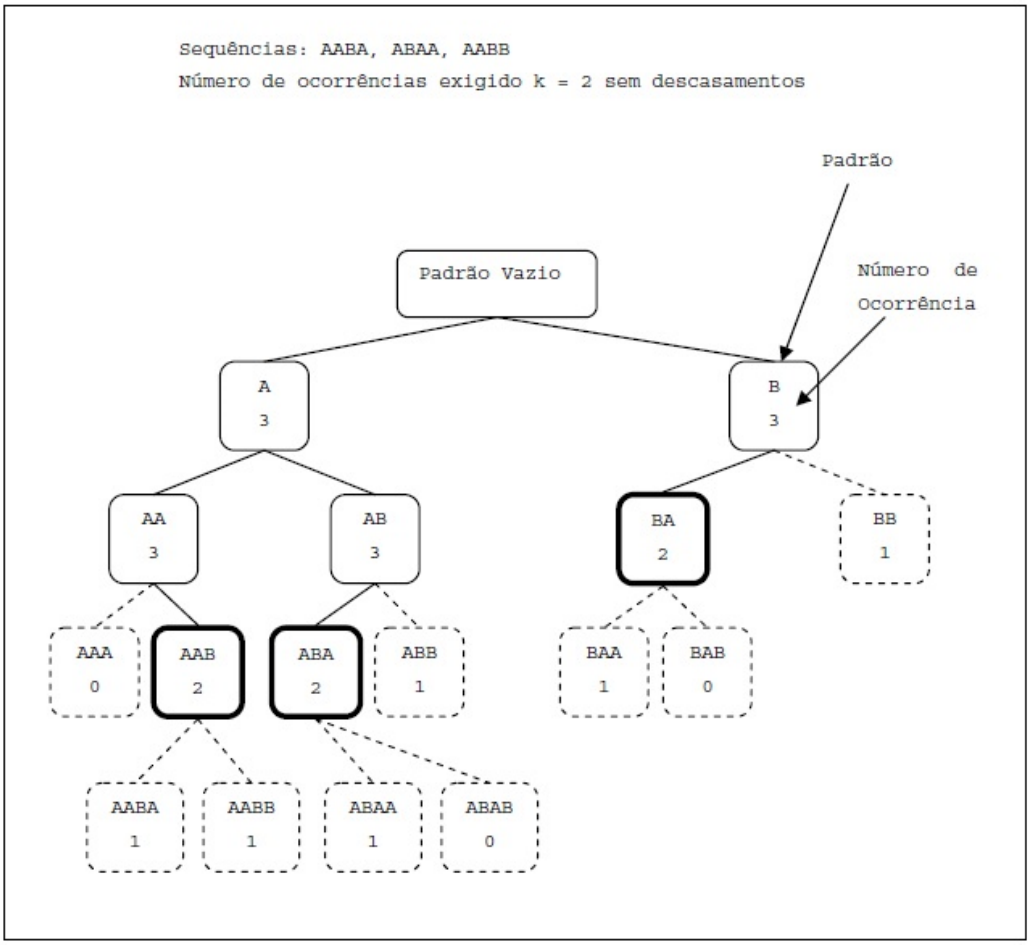
No algoritmo *Pratt*, o objetivo é descobrir padrões mais interessantes que se casam com um número mínimo de sequências dadas. O algoritmo consiste em fornecer um conjunto de sequências desalinhadas, limites que especificam a classe de padrões e um número mínimo de sequências com as quais o padrão deve se casar. Um grafo de padrões é construído a partir do tipo de busca que o usuário quiser fazer, como ilustrado na Figura 5.

No algoritmo *Pratt*, o objetivo é descobrir padrões mais interessantes que se casam com um número mínimo de sequências dadas. O algoritmo consiste em fornecer um conjunto de sequências desalinhadas, limites que especificam a classe de padrões e um número mínimo de sequências com as quais o padrão deve se casar. Um grafo de padrões é construído a partir do tipo de busca que o usuário quiser fazer:

1) Se o usuário não impuser restrições, as menores sequências são concatenadas em uma única cadeia e o grafo é construído;
2) Caso exista uma sequência e o usuário queira procurar padrões que se casem com tal sequência e que ocorram em pelo menos *k* sequências, então um grafo é construído a partir da sequência escolhida;
3) Se houver um alinhamento múltiplo de um subconjunto de sequências, um grafo pode ser construído a partir dele, e a busca é restringida a padrões que consistam com o alinhamento.

O WINNOVER trata-se de um algoritmo que faz podas em suas estruturas de dados para aumentar o desempenho. Desse modo, o algoritmo encontra padrões desconhecidos que aparecem em diferentes regiões em cada sequência pertencente ao conjunto de entrada, pois os padrões estão sujeitos a mutações e não aparecem exatamente iguais.

**Figura 5.** Representação do método de enumeração de padrões com podas.




Fonte: (LEMOS et al., 2003)

Dado um conjunto de sequência, o WINNOVER constrói um grafo G cujos vértices são as subcadeias de comprimento *l*, e os arcos correspondem às subcadeias similares (LEMOS et al., 2003).

O algoritmo TEIRESIAS baseia-se na comparação de padrões pequenos. É um método de enumeração de padrões que utiliza um método combinatorial e relata todos os padrões maximais que possuem um suporte suficiente sem enumerar todo o espaço de padrões. Trata padrões de qualquer comprimento (LEMOS et al., 2003).

O algoritmo TEIRESIAS é exato, garante que todos os padrões maximais que ocorrem em pelo menos *K* sequências sejam encontrados. No entanto, esse número de padrões pode ser alto. A desvantagem é que os descasamentos não são flexíveis, o único descasamento permitido são os caracteres coringas, caracteres usados para substituir caracteres desconhecidos. Além disso, outra desvantagem é que o tempo de execução é exponencial, o que compromete a viabilidade de seu uso (HUSSEIN; RASHID; ABDULAH, 2016).

### 2.6.3 Métodos Heurísticos Iterativos

Um método heurístico iterativo para busca de padrões conhecido em Bioinformática é o *Gibbs Sampling* (LEMOS et al., 2003). Esse algoritmo procura o melhor padrão conservado, sem buracos, de tamanho fixo *W* em uma matriz de pontuações. Funciona com iterações, mas não garante que os conjuntos encontrados serão os melhores, porém converge para um máximo local. O método é rápido, o que o torna viável em várias aplicações.

### 2.6.4 Métodos de Aprendizado de Máquina

Um dos métodos que utiliza estratégias de aprendizado de máquina para reconhecimento de padrões é o *Expectation Maximization* (LEMOS et al., 2003), o qual procura estimar parâmetros do modelo estocástico do padrão, que ocorre uma vez em uma posição desconhecida de cada sequência em uma dada família de sequências. Além disso, essa estratégia usa todas as possíveis ocorrências do padrão para obter uma matriz, em vez de apenas uma ocorrência escolhida em cada cadeia (LEMOS et al., 2003).

Outra estratégia utilizada em Bioinformática para reconhecimento de padrões em biossequências é o *Hidden Markov Model* (HMM) (LEMOS et al., 2003). Basicamente, esse algoritmo pode assumir três tipos de estados: casamentos ou principais, inserção e remoção. Os estados de casamento representam a linha de baixo e modelam as colunas conservadas do alinhamento e especificam a distribuição de probabilidade. Os estados de inserção modelam as regiões altamente variáveis no alinhamento para modelar possíveis buracos entre os estados de casamento. Os estados de remoção permitem pular alguns estados de casamento para modelar a situação em que poucas sequências tem um buraco em uma posição do alinhamento múltiplo.

## 2.7 Ferramentas de Bioinformática

Devido à crescente quantidade de dados genômicos, o envolvimento de técnicas computacionais para comparação de biossequências e especialmente o desenvolvimento de algoritmos eficientes torna-se indispensável para uma análise correta acerca dos dados gerados. Desse modo, a interação entre as áreas de informática e biologia criou a necessidade de comunicação entre especialistas de computação e biólogos.

Assim, nas próximas subseções serão apresentadas algumas ferramentas conhecidas em Bioinformática que auxiliam o trabalho de análises dos biólogos.

### 2.7.1 Ferramenta MUSCLE

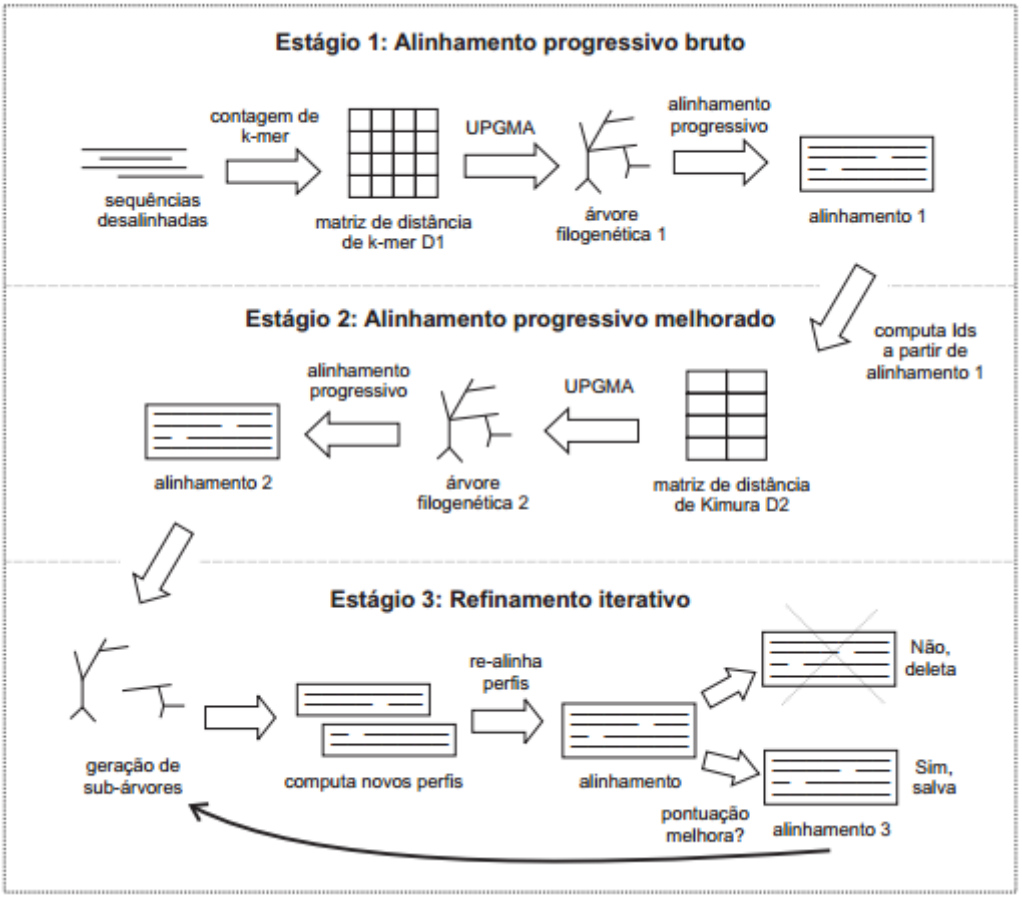
A MUSCLE é uma ferramenta para alinhamento múltiplo de sequências com base na heurística de Alinhamento Progressivo e que apresenta bom desempenho computacional e resultados biológicos significativamente precisos (EDGAR, 2004; MARUCCI et al., 2009).

Basicamente, a ferramenta é dividida em três estágios principais, conforme pode ser observado na Figura 6. Nos dois primeiros estágios, o alinhamento realizado consiste em calcular a distância entre cada par de sequências, construir uma árvore filogenética e, então, realizar o alinhamento entre dois nós irmãos (nós-filhos que possuem o mesmo nó-pai), de modo a armazenar essa informação em um nó pai. Essa etapa é repetida progressivamente até que se atinja o nó raiz, o que finaliza o alinhamento das sequências analisadas. O primeiro estágio é responsável pela obtenção de um alinhamento bruto com métodos de baixo custo computacional. No segundo estágio, esse resultado é aprimorado por meio de outros métodos, como mostra na Figura 6.

No terceiro estágio, é utilizada uma abordagem iterativa. Esse estágio é conhecido como estágio de refinamento, e recebe o resultado do segundo estágio e a árvore guia como entrada. Em cada iteração, o algoritmo elimina elementos que não serão importantes para obter o resultado final, como, por exemplo, exclusão de colunas que só possuem *gaps*. Assim, caso a pontuação da iteração aumente, o alinhamento é mantido, caso contrário, eliminado. Isso ocorre até que todas arestas da árvore sejam visitadas sem que nenhuma mudança seja mantida, ou até atingir o número máximo de iterações. As arestas da árvore são visitadas em ordem decrescente de distância até a raiz, e então realinha-se primeiro individualmente as sequências, que geram grupos fortemente relacionados.

No entanto, conforme observa-se na Figura 7, a MUSCLE é executada por meio de linhas de comando, o que pode prejudicar a usabilidade da ferramenta.

**Figura 6.** Divisão de estágio da Ferramenta MUSCLE




Fonte: (MARUCCI et al., 2009)

**Figura 7.** Execução da Ferramenta MUSCLE

Prompt de Comando

```
http://www.drive5.com/muscle
This software is donated to the public domain.
Please cite: Edgar, R.C. Nucleic Acids Res 32(5), 1792-97.


Basic usage

    muscle -in <inputfile> -out <outputfile>

Common options (for a complete list please see the User Guide):

    -in <inputfile>    Input file in FASTA format (default stdin)
    -out <outputfile>  Output alignment in FASTA format (default stdout)
    -diags             Find diagonals (faster for similar sequences)
    -maxiters <n>      Maximum number of iterations (integer, default 16)
    -maxhours <h>      Maximum time to iterate in hours (default no limit)
    -html              Write output in HTML format (default FASTA)
    -msf               Write output in GCG MSF format (default FASTA)
    -clw               Write output in CLUSTALW format (default FASTA)
    -clwstrict         As -clw, with 'CLUSTAL W (1.81)' header
    -log[a] <logfile>  Log to file (append if -loga, overwrite if -log)
    -quiet             Do not write progress messages to stderr
    -version           Display version information and exit

Without refinement (very fast, avg accuracy similar to T-Coffee): -maxiters 2
Fastest possible (amino acids): -maxiters 1 -diags -sv -distance1 kbit20_3
Fastest possible (nucleotides): -maxiters 1 -diags
```

Fonte: Elaborado pelo Autor.

### 2.7.2 Ferramenta MAFFT

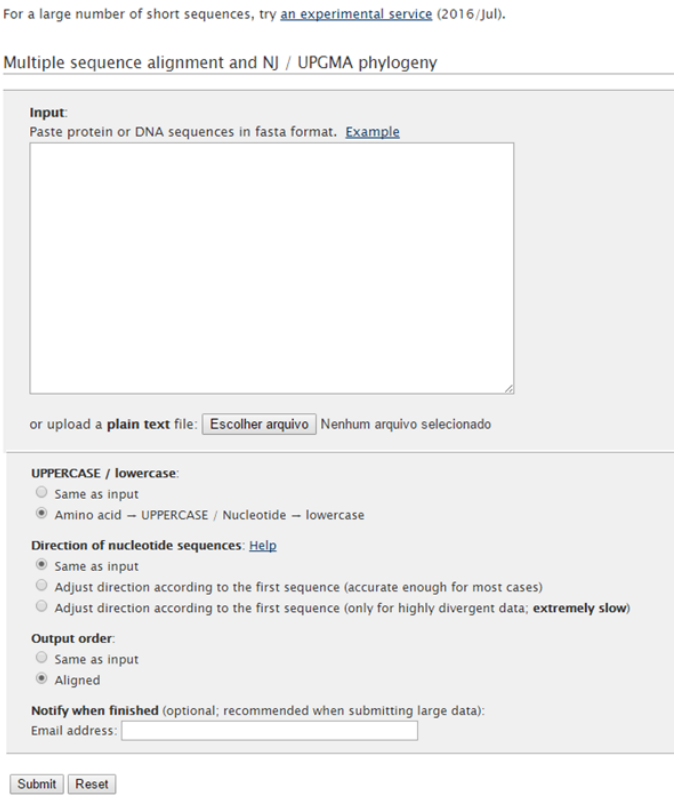
A ferramenta MAFFT (*Mutiple Alignment using Fast Fourier Transform*) utiliza a transformada rápida de Fourier (FFT) para executar alinhamentos múltiplos tanto de nucleotídeos como de aminoácidos.

A MAFFT tem como vantagem a execução de alinhamentos com significância biológica, aliada a um tempo de processamento relativamente menor quando comparada às outras ferramentas de bioinformática (KATOH ET AL., 2002).

O funcionamento da ferramenta divide-se em duas etapas principais, em que a primeira visa detectar regiões homólogas nas sequências de entradas por meio da Transformada Rápida de Fourier. Na segunda etapa, é utilizado um método eficiente que calcula a pontuação do alinhamento.

A MAFFT tem sido utilizada também em pesquisas que têm como foco a evolução dos processos de alinhamento, recebendo, assim, várias melhorias e incrementos em suas versões. Recentemente, recebeu novas alterações e funcionalidades que melhoram a experiência do usuário em termos de usabilidade (KATOH; STANDLEY, 2014), por meio um ambiente de interface *web*[1], conforme pode ser visto na Figura 8.

**Figura 8.** Interface web da ferramenta MAFFT.




Fonte: Elaborado pelo autor.

### 2.7.3 Ferramenta Clustal Omega

A família Clustal possui um conjunto de ferramentas para alinhamento múltiplo de sequências vastamente utilizadas em Bioinformática, de modo que a versão atual é a Clustal Omega (SIEVERS; HIGGINS, 2014).

A ferramenta Clustal Omega utiliza, assim como as versões anteriores, a heurística de Alinhamento Progressivo como base para realizar alinhamento múltiplo de sequências. Tem como vantagem principal, devido às constantes melhorias, a possibilidade de se executar alinhamentos que envolvem um volume muito grande de dados, o que, geralmente, degradadava o resultado obtido em versões anteriores.

Do ponto de vista biológico, as melhorias consistem na utilização do algoritmo *HHalign*, o qual se baseia no HMM. Nesse ponto, é importante observar também que é possível a reutilização de alinhamentos, de modo a se evitar recálculos e viabilizar a especificação de perfil do HMM, que pode ser derivado de sequências homólogas às de entradas (SIEVERS et al., 2011).

Do ponto de vista de eficiência computacional, a ferramenta Clustal Omega tem como vantagem a implementação do algoritmo *mBED* para construir a árvore filogenética, que é considerada a etapa mais custosa na heurística de Alinhamento Progressivo. Adequações nesse algoritmo por meio das novas versões possibilitaram a viabilidade de execução do alinhamento de um grande volume de sequências (SIEVERS et al., 2011).

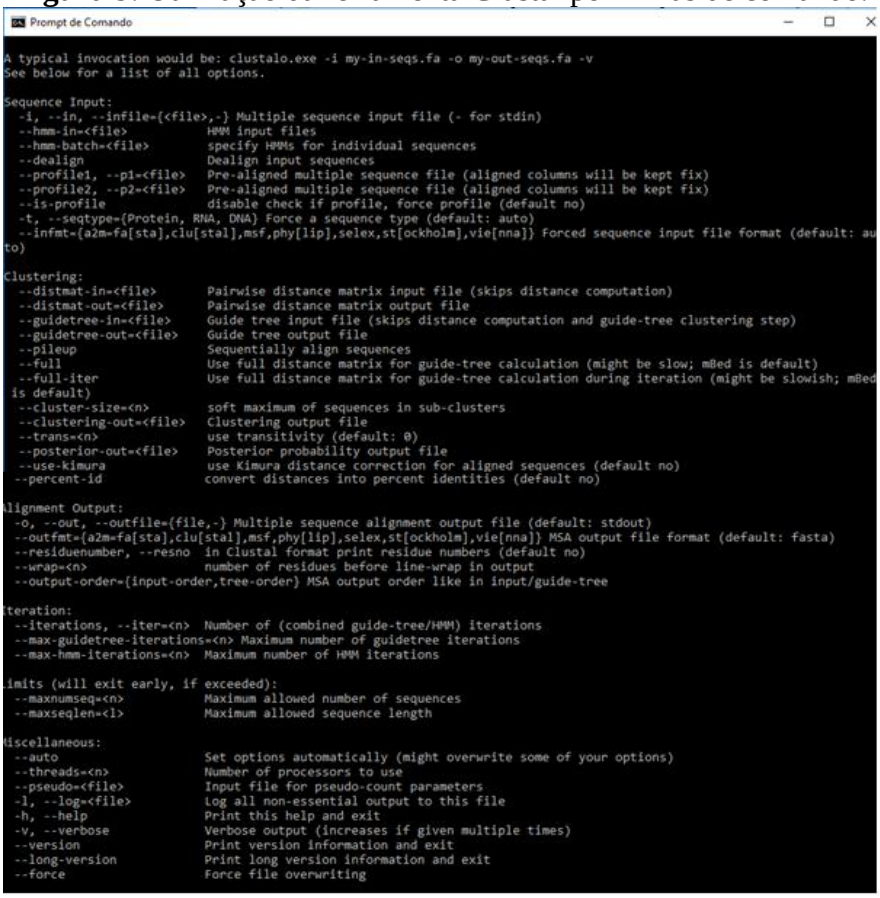
Conforme observa-se na Figura 9, inicialmente a execução da Clustal Omega se dava por meio de linhas de comando. No entanto, de modo a propiciar melhor usabilidade ao usuário, foi desenvolvida uma interface gráfica para a ferramenta, a qual pode ser gratuitamente acessada pela *web*[1].

### 2.7.4 A Ferramenta LOCQSPEC

A LOCQSPEC é uma ferramenta para reconhecimento de determinados padrões em biossequências, chamados de *quasispecies*, que tem como objetivo identificar semelhanças entre todas as sequências analisadas, para que se possam ser extraídas relações biológicas e funcionais (MARUCCI et al., 2008).

[1] http://www.ebi.ac.uk/Tools/msa/clustalo/

**Figura 9**. Utilização da ferramenta Clustal por linhas de comando.




Fonte: Elaborado pelo autor.

Inicialmente, a ferramenta recebe um arquivo de entrada que contém as sequências a serem analisadas. O arquivo de entrada deve estar no formato FASTA, e, assim como as ferramentas CLUSTAL e MUSCLE, a sua operabilidade é baseada em linhas de comando.

Feito isso, as sequencias semelhantes são separadas em grupos, para que posteriormente sejam extraídas similaridades ou diferenças entre as amostras.

A visualização das variabilidades é a última etapa do LOCQSPEC, que consiste em apresentar as posições conservadas, alteradas e as semelhanças e diferenças entre as sequências após a análise.

## 2.8 Interface gráfica e usabilidade

No desenvolvimento de software, não se deve apenas focar na entrega do produto funcional, mas além disso deve se pensar na interação do usuário final com o produto. A interface gráfica é a conexão entre o usuário e o sistema e, portanto, é de grande importância que seja de fácil utilização. A função da interface é apenas prover o contato do sistema com o

humano (RIGON, 2016 apud SCHULENBERG, 2013).

Em sistemas computacionais que necessitam da interação do usuário, a interface gráfica é crucial, pois, caso não seja de fácil uso e entendimento, o usuário pode se sentir frustrado ao usá-la, e até mesmo pode gerar erros na execução das tarefas, devido à confusão na hora de manusear o sistema.

A Interação Humano-Computador (IHC) é uma área de estudo atual e de grande e constante avanço (YAMAMOTO, 2015 apud HEWETT, 1992). A IHC tem como objetivo prover aos usuários de sistemas segurança e produtividade, supondo que, para que isso aconteça, é necessário, primeiramente, seu entendimento (YAMAMOTO, 2015 apud PREECE, 1994).

A usabilidade é considerada um fator que assegura essas questões, pois, a partir dela, pode-se analisar e comprovar a eficácia dos sistemas para com os usuários, a partir de metas como ser eficaz no uso, ser segura no uso, ser de boa utilidade e ser de fácil aprendizagem.

Além da usabilidade, a ergonomia e a comunicabilidade são os critérios mais utilizados para a avaliação de interfaces. Segundo apresentações, o conceito de usabilidade tem relação com a facilidade do usuário de aprender e utilizar o sistema de forma simples, bem como a satisfação na utilização da aplicação (NIELSEN, 1993).

Em bioinformática, geralmente os usuários são biólogos que, em sua maioria, não possuem familiaridade com certos tipos de *softwares*. Nota-se que a maioria das ferramentas de bioinformática são desenvolvidas para serem executadas por meio de linha de comando, o que pode se tornar confuso para usuários não familiarizados. Atualmente, a questão da interface gráfica está sendo melhor tratada, assim como elucidado nas seções anteriores.

A partir da necessidade de melhoria da interface gráfica e a fim de se aliar o conceito de usabilidade para propor uma ferramenta com interface amigável e funcional, o uso das Heurísticas de Usabilidade de Jakob Nielsen pode ser adequado para se obter as características desejadas.

### 2.8.1 Heurísticas de usabilidade de Jakob Nielsen

Jakob Nielsen, criador das Heurísticas de Usabilidade, é um cientista da computação com PhD em Interação Humano-Computador. Criador do movimento “engenharia de usabilidade com desconto”, que consiste na criação de interfaces rápidas e baratas.

Nielsen é considerado o mais conhecido analista de usabilidade do mundo, de modo a promover testes e pesquisas para comprovar a eficácia de suas análises.

As Heurísticas de Nielsen consistem em 10 parâmetros para avaliação de interfaces, com o intuito de evitar erros comuns que ele mesmo encontrava em suas análises (NIELSEN, 1993). No caso, as avaliações Heurísticas eram feitas por, no mínimo, 3 e, no máximo, 5 profissionais especializados.

Essas Heurísticas encontram-se descritas a seguir:

1. **Visibilidade e Status do Sistema**

   Consiste em sempre deixar o usuário ciente do que está acontecendo, para que, assim, ele possa se orientar no sistema.

2. **Relacionamento entre a interface e o mundo real**

   Essa heurística parte do pressuposto de que não se deve usar termos técnicos, que não são intuitivos para usuários considerados leigos, mantendo sempre a coerência, para que o usuário não se perca ou não se sinta frustrado.

3. **Liberdade e controle do usuário**

   Nesse ponto, Nielsen deixa claro que o usuário precisa conseguir desfazer ou refazer ações que foram feitas, para evitar que o usuário se perca ou caia em situações inesperadas. Porém, vale lembrar que o usuário pode, sim, fazer o que quiser no sistema, desde que respeite as regras de negócio da aplicação.

4. **Consistência e padronização**

   Manter sempre um padrão na interação com o usuário, ou seja, botões de enviar devem ter sempre o mesmo padrão, assim como botões de cancelar e ações que o usuário estará sempre tomando.

5. **Prevenção de erros**

   Assim como falado pelo próprio Nielsen, “Ainda melhor que uma boa mensagem de erro é um *design* cuidadoso que possa prevenir esses erros”. Nesse caso, Nielsen se refere a, por exemplo, sugestões de palavras, correção de ortografia e, principalmente, deleções e ações definitivas, que, segundo essa Heurística, devem vir sempre acompanhadas de *checkboxes* ou mensagens de confirmação, para que o usuário confirme que está realmente de acordo com a ação solicitada e suas consequências.

6. **Reconhecimento ao invés de lembrança**

   Para que isso funcione, a interface deve ser intuitiva e clara, assim, o usuário entende o que está fazendo e consegue sempre reproduzir suas ações sem precisar decorá-las. Uma técnica utilizada é *breadcrumb*, em que o sistema informa ao usuário os caminhos que ele tomou para chegar onde está, servindo como uma orientação para o usuário.

7. **Flexibilidade e eficiência de uso**

   A flexibilidade consiste em manter o sistema fácil de ser usado para leigos, mas também otimizado para usuário avançados, que já tenham experiência com o sistema em questão. Isso pode ser atingido ao implementarmos teclas de atalho, foco nos campos que estão sendo utilizados, máscaras e facilidade de navegação entre os campos de um formulário, por exemplo.

8. **Estética e *design* minimalista**

   Os detalhes do sistema não precisam ser exagerados, ou seja, o *design* não precisa se sobressair aos olhos do usuário mais do que a real funcionalidade do sistema. Manter mensagens simples e diretas ajudam muito nesse caso, assim como foi o criado o *Google Material Design*, com um design padrão e minimalista em todos seus sistemas.

9. **Ajude o usuário a reconhecer, diagnosticar e sanar seus erros**

   Mensagens de erro devem se dotadas de explicações e possíveis saídas para o erro que está sendo tratado. Deve-se, ao máximo, tentar evitar intimidar o usuário com mensagens de erro agressivas e com termos não convencionais, usando sempre mensagens que esclareçam o ocorrido, além de ajudar a sanar o problema encontrado.

10. **Ajuda e documentação**

    Ainda que um bom design deva evitar a necessidade de um conteúdo de ajuda para se utilizar o sistema, deve-se, ainda, criar um conjunto de documentação que possa orientar o usuário em caso de dúvida. Essa ajuda deve ser visível e facilmente acessada, de modo a deixar o usuário confortável em usá-la, por meio de métodos rápidos de acesso e de busca.

### 2.8.2 Interfaces Gráficas em Bioinformática

O uso de Interfaces Gráficas, principalmente em Bioinformática, é um fator crucial para garantir a interoperabilidade do sistema, que muitas vezes vem a ser um fator que limita o uso de certo programa ou ferramenta.

Como apresentado anteriormente, pesquisadores e usuários finais das ferramentas de bioinformática geralmente não possuem o conhecimento necessário para utilizarem sistemas que tenham a necessidade de manipulação por meio de linhas de comando. Em contraste às linhas de comando, a Interface Gráfica facilita e torna prática a utilização destas ferramentas. Além da qualidade da ferramenta em questão, um nível de interatividade adequado de sua interface é muito importante. Dados de qualidade com uma interface ruim, ou vice-versa, nunca

são suficientemente apreciados (BATEMEN, 2007).

Conforme pode ser observado no trabalho de OHTSUBO et al. (2016), ao desenvolver a ferramenta *GenomeMatcher* que tem como função identificar e apresentar similaridades genômicas por imagens bidimensionais e indicadas por cor, o uso da interface gráfica auxilia fortemente não só na operabilidade do sistema, mas também ao apresentar seus resultados para o usuário, sendo classificado como eficiente e de fácil uso (OHTSUBO et al, 2016).

Pode ser observado também no trabalho desenvolvido por WETTENHALL; SMYTH (2004), que a ferramenta *limmaGUI* baseou-se em uma interface gráfica para melhorar sua usabilidade. A ferramenta que tem como função a modelagem linear de dados de *microarray*, utiliza-se da interface gráfica para apresentar seus métodos estatísticos, auxiliando o usuário nessa manipulação através de cliques pelos componentes da interface. As funções disponíveis para manipulação do usuário através de clique são métodos de *background correction*, demonstrações de interface, normalização e dados de *microarray (*WETTENHALL; SMYTH, 2004).

No trabalho desenvolvido por JO et al. (2008), observa-se também mais um caso de utilização da interface gráfica para auxiliar na operabilidade de sistemas de Bioinformática. A ferramenta *CHARMM*, que tem como função análise e manipulação macromolecular, foi contemplada com uma interface gráfica para gerar uma grande quantidade de arquivos para facilitar e padronizar suas simulações. O ambiente gráfico em questão provê uma plataforma ideal para construir e validar sistemas de modelo molecular com interatividade suficiente para que, quando um problema é identificado pela inspeção visual, haja a possibilidade voltar para os outros passos e gerar novamente o sistema inteiro (JO et al., 2008).

A partir dos trabalhos e definições apresentadas anteriormente, percebe-se claramente que o uso de interfaces gráficas no âmbito da bioinformática é crucial. Desta forma, o presente trabalho destaca-se na proposta de uma interface gráfica intuitiva para assistir aos problemas de identificação de quasiespécies. Assim, haverá uma contribuição significativa no âmbito de operação desta ferramenta por parte de usuários não nativos da computação.

# CAPÍTULO 3 – DESENVOLVIMENTO DO TRABALHO

## 3.1 Considerações iniciais

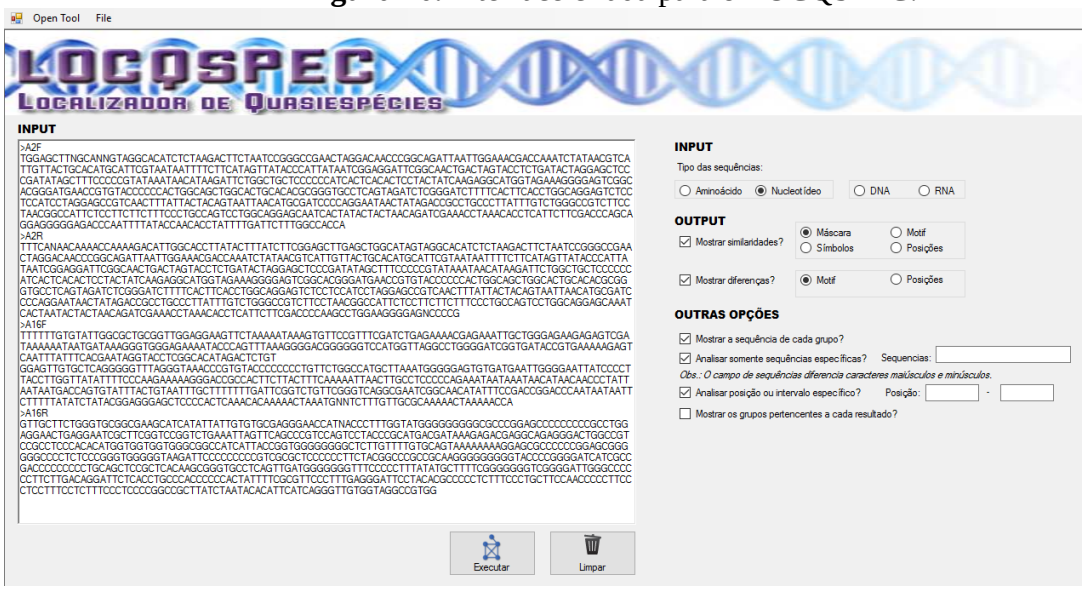
Neste capítulo, são mostrados os passos seguidos da implementação do projeto, que consiste na criação da interface, com base nas Heurísticas de Jakob Nielsen, e adaptação da chamada do núcleo da ferramenta LOCQSPEC para funcionamento na linguagem C#.

Além da melhoria da interface, é demonstrada também, a implementação do módulo de limpeza de sequências, que é acoplada como nova funcionalidade à LOCQSPEC.

## 3.2 A criação da interface

A interface foi criada com o intuito de oferecer maior usabilidade para o usuário o final, que, geralmente, é um biólogo que não é familiarizado com linhas de comando. Portanto, todos os parâmetros necessários para a execução, desde a etapa de limpeza das sequências até a execução do alinhamento em si, foram apresentados ao usuário por meio de *checkboxes* ou *radio buttons*, o que facilita a utilização da ferramenta. Na Figura 10 pode-se observar a interface completa do LOCQSPEC, implementada no presente trabalho, já com um arquivo de DNA inserido e algumas opções selecionadas.

**Figura 10.** Interface criada para o LOCQSPEC.

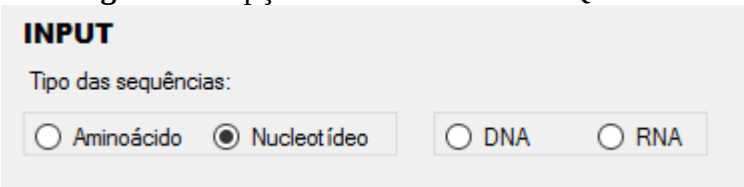



Fonte: Elaborado pelo autor.

Na Figura 11, pode-se observar as opções de entra, em que o usuário seleciona se a sequência de entrada é um aminoácido ou um nucleotídeo (selecionando também se é RNA ou DNA, nesse caso) de maneira intuitiva, para que o algoritmo de limpeza receba a informação

juntamente com o arquivo inicial.

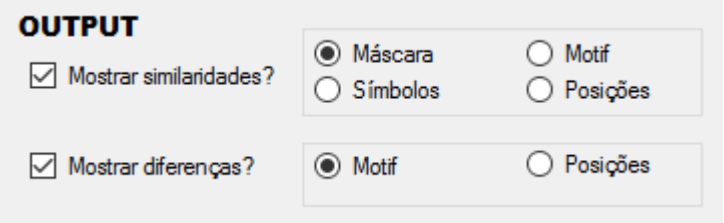
**Figura 11.** Opções de entrada da LOCQSPEC.




Fonte: Elaborado pelo autor.

Na Figura 12, são apresentadas as opções de saída, que serão apresentadas no arquivo final. O usuário escolhe se pretender ver as semelhanças e/ou as diferenças das sequências analisadas, além das opções com relação aos tipos de apresentações dessas características.

**Figura 12.** Opções de saída da LOCQSPEC.

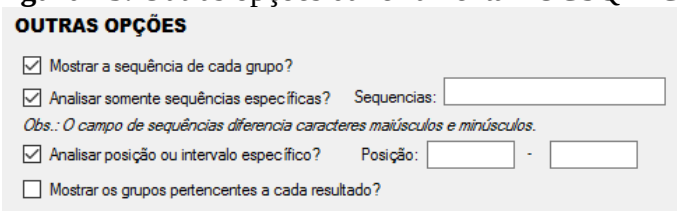



Fonte: Elaborado pelo autor.

Além das opções de entrada e saída, existem também as outras opções que o usuário pode selecionar, demonstradas na Figura 13. Uma delas é a opção de mostrar as sequências analisadas. Essa opção é interessante pois, ao selecioná-la, o usuário vê no arquivo de saída a limpeza que foi feita nas sequências.

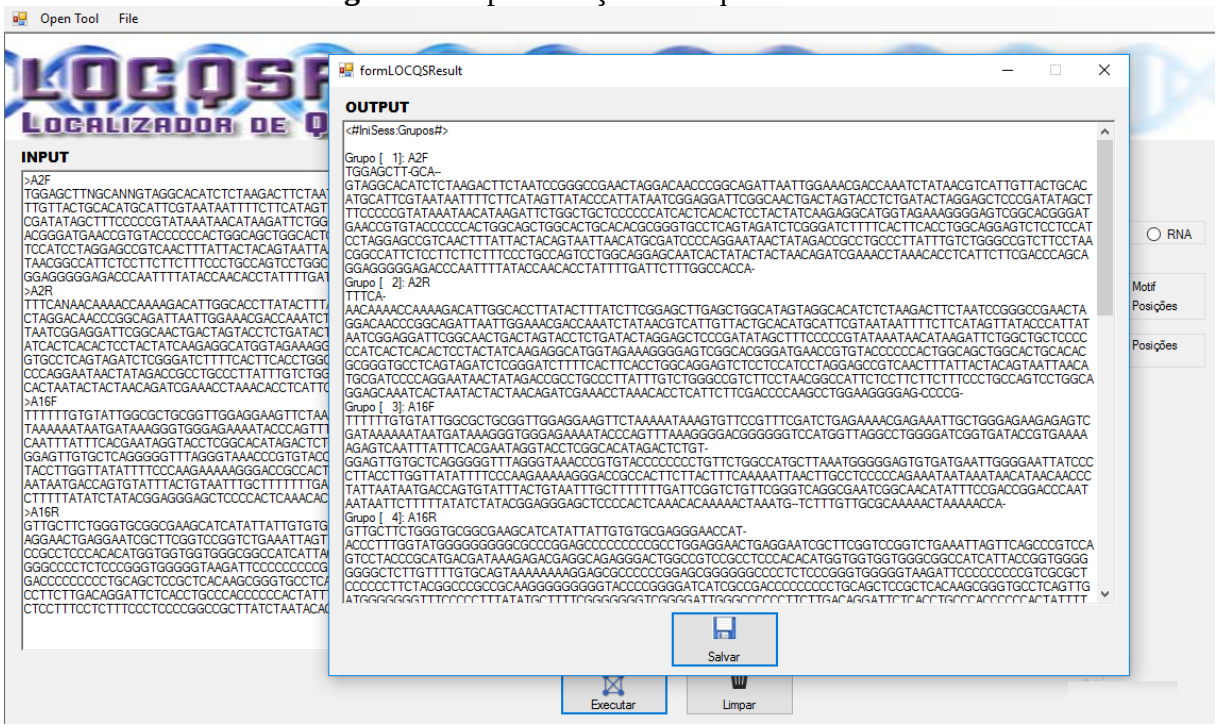
Ao executar o processo, o núcleo da ferramenta é chamado e então executa primeiramente a limpeza do arquivo de entrada, e posteriormente o alinhamento das sequências. Na Figura 14, é demonstrado como o arquivo de saída é apresentado ao usuário. A interface utiliza um objeto *richTextBox* que apresenta para o usuário tanto o arquivo de entrada quanto o de saída já em suas formatações específicas.

**Figura 13.** Outras opções da ferramenta LOCSQPEC.




Fonte: Elaborado pelo autor

**Figura 14.** Apresentação do arquivo de saída.



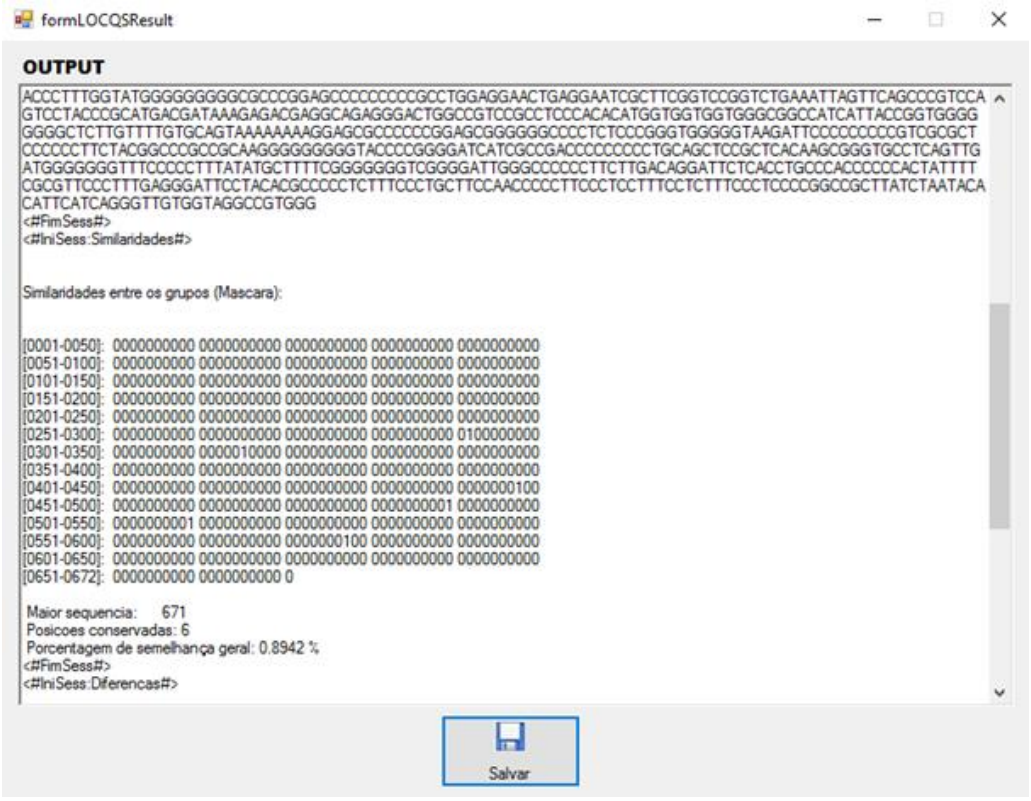

Fonte: Elaborado pelo autor.

Ao selecionar a opção de mostrar as sequências, pela apresentação do arquivo de saída o usuário é capaz de notar onde foram inseridos os *gaps*, que representam a substituição dos caracteres que não fazem parte do alfabeto da sequência entrada pelo usuário.

Na Figura 15, são apresentadas, também, as informações sobre a similaridade entre as sequências. É importante ressaltar que os dados são apresentados aos usuários de uma maneira visualmente simples e direta.

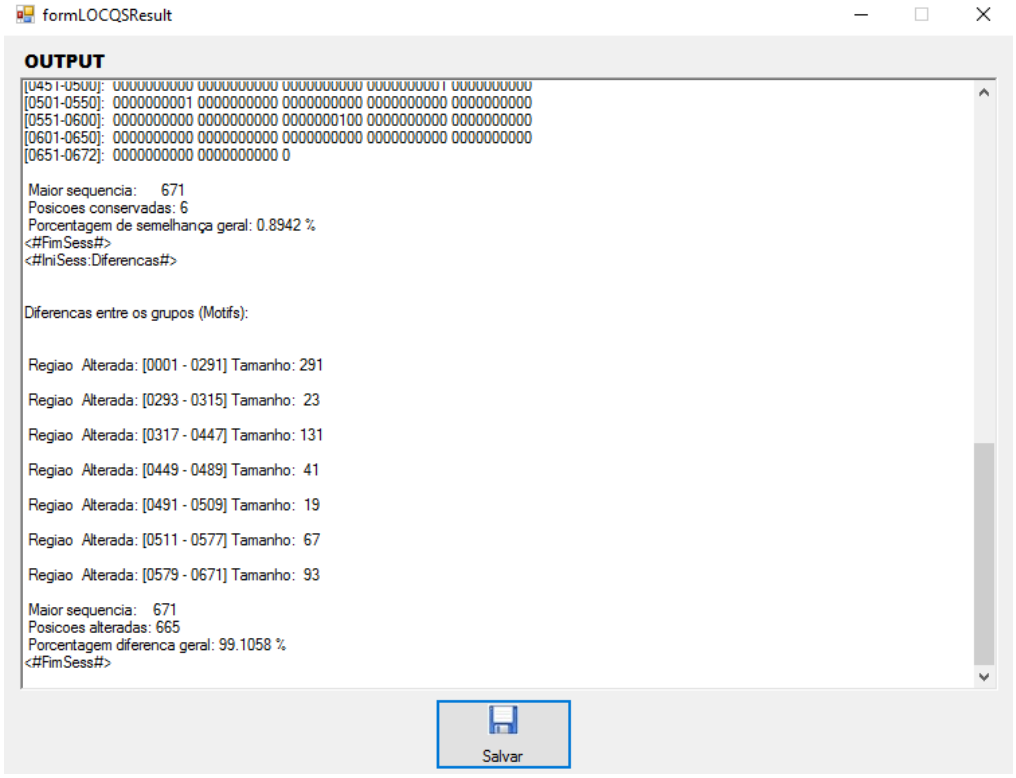
De maneira similar, na Figura 16 são apresentadas as informações sobre as diferenças nas sequências analisadas.

**Figura 15.** Similaridades no arquivo de saída.




Fonte: Elaborado pelo autor.

**Figura 16.** Diferenças no arquivo de saída.




Fonte: Elaborado pelo autor

Por fim, se necessário, o usuário pode salvar o resultado obtido em um arquivo junto à sua máquina para uso posterior. Esse processo é simples e intuitivo, executado quando o usuário pressiona o botão “Salvar”.

### 3.3 A chamada do núcleo

Com a nova implementação em C#, a classe utilizada é a *Process.StartInfo*, cuja função é abrir o *prompt* de comandos do *Windows* para que se possa executar programas por linha de comando. Como demonstrado na Figura 17, nesse caso, também foram armazenadas em uma variável do tipo *string* as informações necessárias para passar como parâmetro para os métodos. A partir dessa *string*, é formado um comando *shell* que é passado como parâmetro para a função encarregada de executar o comando.

Para o método *Process.Start*, é passada como parâmetro a variável *processInfo* que contém todos argumentos necessários para a execução da ferramenta, assim como todos os comandos de execução e realocação dos arquivos. Além disso, é utilizada a função *WaitForExit()*, cujo objetivo é deixar a execução do programa prosseguir caso o *shell* termine sua execução, de modo a controlar a parte de consumir os arquivos gerados pelos executáveis do sistema, com cuidado para que não ocorra o caso de consumir arquivos de execuções anteriores.

### 3.4 O algoritmo de limpeza de sequências

O algoritmo de limpeza a ser implementado faz o papel de um pré-mecanismo do sistema principal. O sistema, anteriormente, recebia como um de seus parâmetros o arquivo no formato FASTA para realizar o alinhamento da sequência. A partir de agora, o sistema, antes de executar o método principal, faz uma chamada à função de limpeza de sequências, que está contida no arquivo *limpaseqs.cpp*.

A função de limpeza, por sua vez, recebe a informação se a sequência a ser limpa é uma proteína ou um nucleotídeo. Em caso de nucleotídeo, é necessário informar se é um RNA ou DNA. No caso de um DNA, a função se encarrega de encontrar caracteres que não façam parte de um corpo de DNA, que é formado apenas por caracteres A, T, C e G e então substituí-los por um *gap* (‘-‘).

Assim, o próximo passo é a passagem do arquivo gerado para o método principal, que se encarrega de realizar o alinhamento das sequências e gerar o resultado final com os padrões

encontrados.

**Figura 17.** Gerenciamento da chamada do núcleo e passagem de parâmetros

```
//EXECUÇÃO DO LIMPASEQS
//Não cria janela do bash
var processInfo = new ProcessStartInfo();
processInfo.WindowStyle = ProcessWindowStyle.Hidden;
processInfo.CreateNoWindow = true;

//Executa o limpaseqs pelo bash
processInfo.FileName = "cmd.exe";
processInfo.WorkingDirectory = ".\\Ferramenta";
processInfo.Arguments = "/c limpaseqs.exe " + Path.GetFileName(openFileDialog1.FileName) + " " + opt_limpa;
processInfo.UseShellExecute = false;

//Executa o limpaseqs
Process.Start(processInfo).WaitForExit();

//EXECUÇÃO DO LOCQSPEC
//Não cria janela do bash
processInfo.WindowStyle = ProcessWindowStyle.Hidden;
processInfo.CreateNoWindow = true;

//Executa o locqspec pelo bash
processInfo.FileName = "cmd.exe";
processInfo.WorkingDirectory = ".\\Ferramenta";
processInfo.Arguments = "/c locqspec.exe -in limpa.fas -out novoArquivo.fas -seqtype " + seqtype + aux_command;
processInfo.UseShellExecute = false;

//Executa o locqspec
Process.Start(processInfo).WaitForExit();
//MOSTRAR ARQUIVO EM NOVA TELA
formLOCQSResult f = new formLOCQSResult();
f.MaximizeBox = false;
f.FormBorderStyle = FormBorderStyle.FixedSingle;
f.StartPosition = FormStartPosition.CenterScreen;
StreamReader SR = new StreamReader(".\\Ferramenta\\novoArquivo.fas", Encoding.Default);
string data = SR.ReadToEnd();
SR.DiscardBufferedData();
SR.Close();
f.richTextBox1.Text = data;
f.Show();
```

Fonte: Elaborado pelo autor.

### 3.5 Implementação do algoritmo de limpeza de sequências

A implementação do algoritmo começa com a adição da classe *limpaseqs.cpp* ao sistema do LOCQSPEC. Conforme elucidado anteriormente, DNA e RNA são processados no mesmo método, chamado *dnarna*, que filtra os caracteres inválidos das sequências, que são bastante parecidas em seus corpos, diferenciando-se apenas com relação à base U (uracila) do RNA para a base T (timina) do DNA.

O arquivo que contém as sequências deverá ser passado para a função, juntamente com a informação se é um nucleotídeo ou uma proteína, e informado, caso seja um nucleotídeo, se é DNA ou RNA. No caso de ser uma proteína, o método chamado será o *proteina*, que funciona analogamente ao método *dnarna*, buscando caracteres “lixo” e substituindo-os por *gap*.

Na Figura 18, observa-se uma demonstração da prototipação da classe *limpaseqs.cpp*.

**Figura 18.** Protótipo da classe *limpaseqs.cpp.*

```
class limpaseqs
{
	FILE *seq_limpa;

	public:

	void dnarna(const char*, int );
	void proteina(const char*);

};
```

Fonte: Elaborado pelo autor.

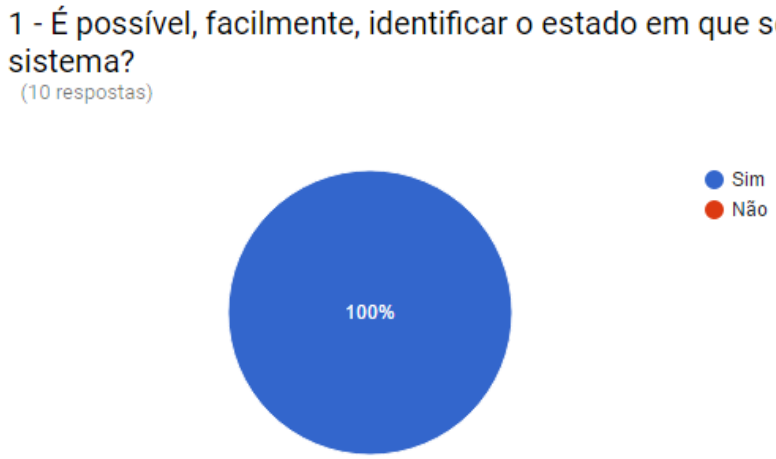
# CAPÍTULO 4 – RESULTADOS OBTIDOS

## 4.1 Melhoria na Usabilidade

Para comprovar os resultados obtidos em termos de melhoria na Usabilidade, foi desenvolvido um formulário com dez perguntas baseadas nas Heurísticas de Jakob Nielsen, apresentadas anteriormente. A ferramenta LOCQSPEC com a nova interface gráfica, exibida no presente trabalho, foi utilizada por dez pessoas, em que dois são leigos em relação ao assunto, quatro são desenvolvedores de *software* e quatro são biólogos. Para cada questão apresentada no formulário, os usuários escolheram entre as respostas “sim” ou “não”.

Na Figura 19 é possível observar os resultados obtidos com relação à primeira heurística, em que 100% dos usuários julgaram que é fácil identificar o estado em que se encontram junto à ferramenta. Isso indica que a interface ajuda o usuário a se situar em cada momento do processo executado.

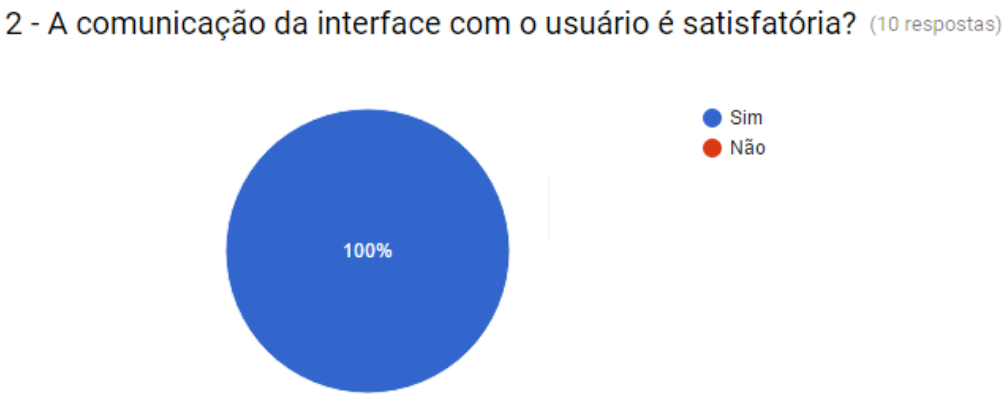
**Figura 19.** Avaliação da primeira heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 20 é possível observar os resultados obtidos com relação à segunda heurística, em que 100% dos usuários julgaram satisfatória a comunicação com a interface do sistema. Esse item indica que as melhorias implementadas no presente trabalho ofereceram ao usuário uma maneira mais facilitada de se utilizar o sistema.

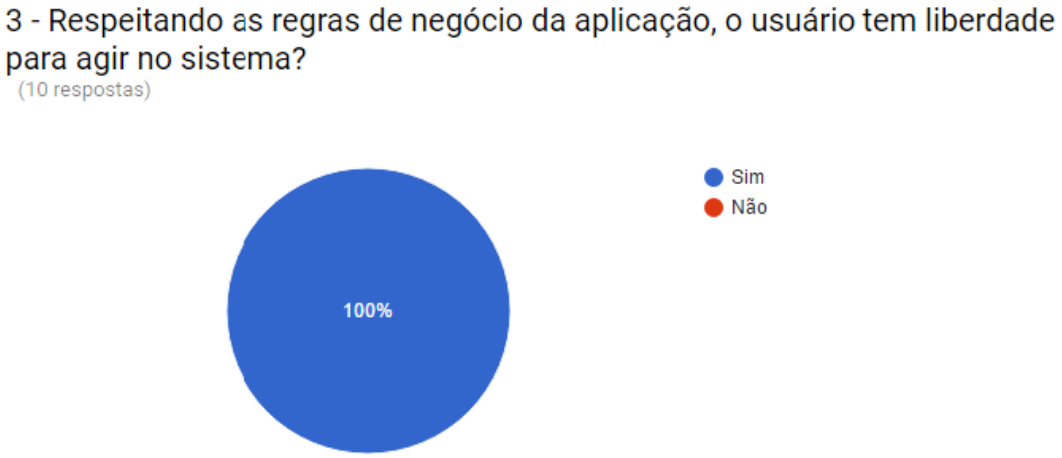
**Figura 20.** Avaliação da segunda heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 21 é possível observar os resultados obtidos com relação à terceira heurística, em que 100% dos usuários responderam “sim” com relação à liberdade para agir no sistema. Esse item indica que a interface propiciou ao usuário uma maneira mais facilitada para executar o sistema com base na liberdade para se escolher quais parâmetros de execução serão selecionados.

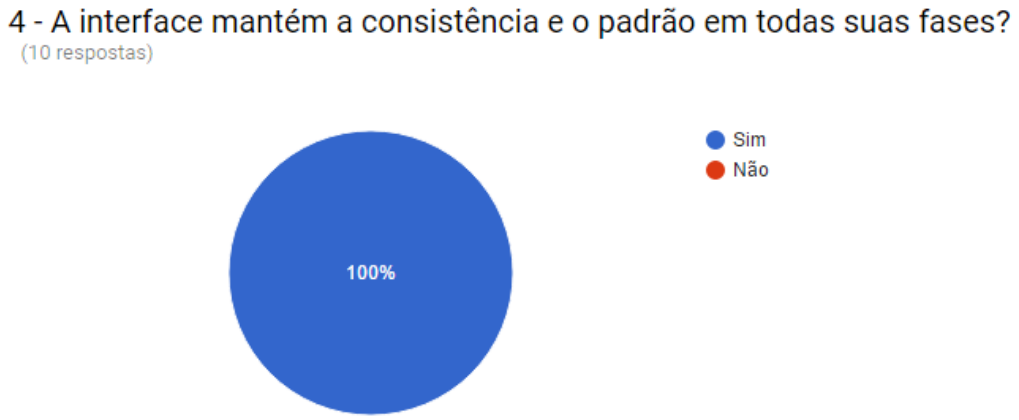
**Figura 21.** Avaliação da terceira heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 22 é possível observar os resultados obtidos com relação à quarta heurística, em que 100% dos usuários responderam “sim” quando perguntados se a interface mantém uma consistência e um padrão em todas as suas fases. Esse item aponta que a ferramenta é homogênea em todas as etapas de sua execução.

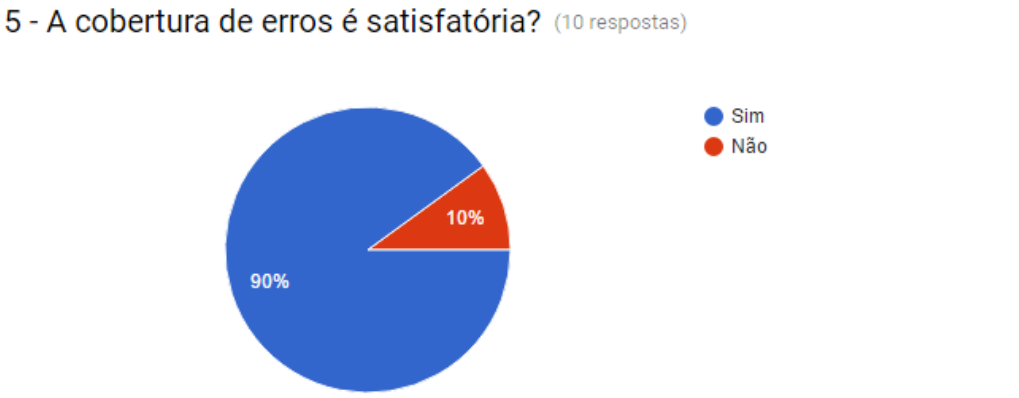
**Figura 22.** Avaliação da quarta heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 23 é possível observar os resultados obtidos com relação à quinta heurística, em que 90% das pessoas responderam que a cobertura de erros é satisfatória. Esse item indica que os erros de execução, em sua maioria, são tratados junto ao sistema, o que é importante para a usabilidade da ferramenta.

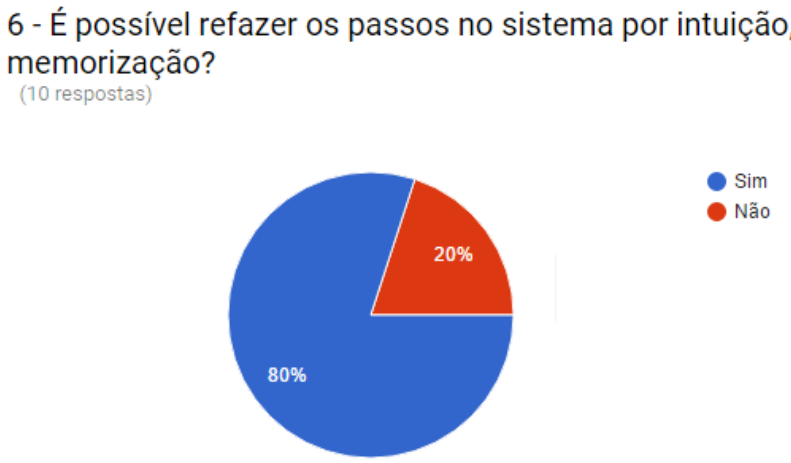
**Figura 23.** Avaliação da quinta heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 24 é possível observar os resultados obtidos com relação à sexta heurística, em que 80% das pessoas responderam que é possível refazer os passos de execução por intuição e não por memorização. Esse item indica que o uso da ferramenta é intuitivo por meio da interface implementada, de modo que não é necessária a memorização de um processo para execução.

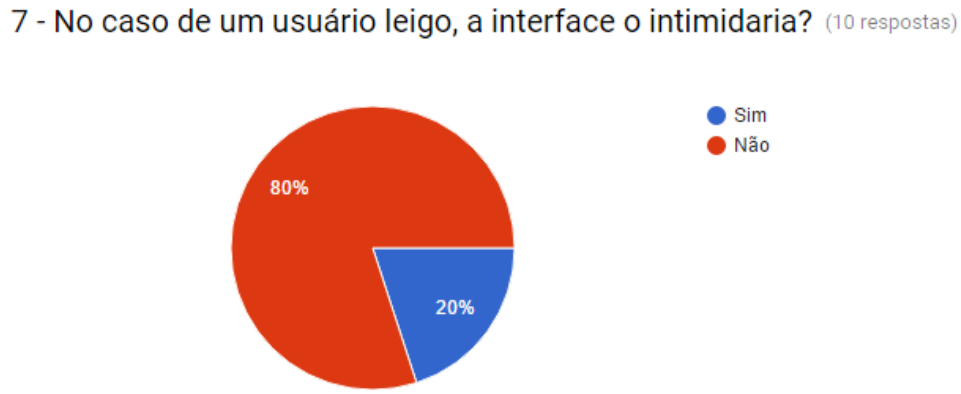
**Figura 24.** Avalição da sexta heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Na Figura 25 é possível observar os resultados obtidos com relação à sétima heurística, em que 80% dos usuários afirmam que a interface não intimidaria um usuário leigo. Esse item indica a facilidade de uso e clareza da ferramenta, fatores que são de suma importância.

**Figura 25.** Avaliação da sétima heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

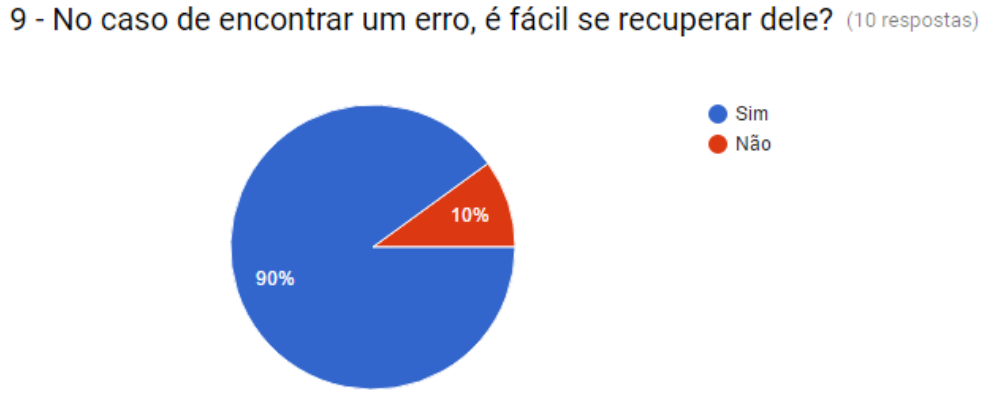
Na Figura 26 é possível observar os resultados obtidos com relação à oitava heurística, em que 100% dos avaliadores responderam que a interface não é confusa, o que confirma o sucesso da abordagem do presente trabalho com relação à facilidade de uso.

**Figura 26.** Avaliação da oitava heurística de Jakob Nielsen.


Fonte: Elaborado pelo autor.

Conforme pode ser observado na Figura 27, cerca de 90% dos usuários responderam que é fácil de se recuperar de um erro apresentado pela ferramenta. Esse item indica que a interface gráfica implementada no presente trabalho ofereceu maior resiliência à LOCQSPEC, o que é um fator importante em termos de usabilidade.

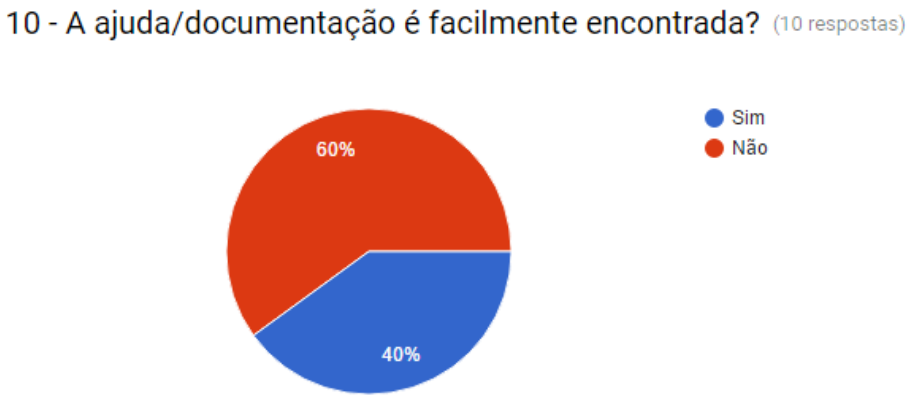
**Figura 27.** Avaliação da nona heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

Por fim, na Figura 28, observa-se que em 60% dos casos, o usuário respondeu que manuais de ajuda de utilização da ferramenta não são facilmente encontrados em sua interface. Essa avaliação é importante pois é possível observar um ponto de melhoria que ainda pode acrescentar à facilidade de uso da ferramenta.

**Figura 28.** Avaliação da décima heurística de Jakob Nielsen.




Fonte: Elaborado pelo autor.

## 4.2 Melhorias obtidas pelo algoritmo de limpeza

Com relação ao algoritmo de limpeza de sequências, foram obtidas melhorias com relação à análise de diferenças, semelhanças e também nas posições conservadas e alteradas das sequências, de modo a propiciar um resultado mais preciso após eliminar os caracteres indevidos das sequências de entrada.

Assim, na Figura 29 são mostrados os resultados obtidos pela versão original da LOCQSPEC a partir de um arquivo de entrada com sequências de nucleotídeos.

**Figura 29.** Resultado final sem o mecanismo de limpeza de sequências.

```
Maior sequencia:      1900
Posicoes conservadas: 1430
Porcentagem de semelhança geral: 75.2632 %
Maior sequencia:    1900
Posicoes alteradas: 470
Porcentagem diferenca geral: 24.7368 %
```

Fonte: Elaborado pelo autor.

Do mesmo modo, o teste foi realizado com o algoritmo de limpeza implementado no presente trabalho, em que os resultados obtidos podem ser observados na Figura 30.

**Figura 30.** Resultado final com o mecanismo de limpeza de sequências.

```
Maior sequencia:      1932
Posicoes conservadas: 1460
Porcentagem de semelhança geral: 75.5694 %
Maior sequencia:    1932
Posicoes alteradas: 472
Porcentagem diferenca geral: 24.4306 %
```

Fonte: Elaborado pelo autor.

Nota-se que o número de posições conservadas encontradas pela ferramenta passou de 1430 para 1460, o que implica que, com o algoritmo de limpeza implementado no presente trabalho, a LOCQSPEC foi capaz de fazer um reconhecimento de padrões melhor em termos de significância biológica. Além disso, observa-se que a porcentagem de diferença geral entre as sequências de entrada passou de 24,7% para 24,4%, o que indica que após o refinamento de limpeza das sequências, o percentual de semelhança aumentou, o que também implica em resultados com maior significância biológica e, consequentemente, em análises melhores por parte dos biólogos.

Além das melhorias obtidas nas porcentagens de diferença e semelhança e nas posições alteradas e conservadas, a LOCQSPEC anteriormente não conseguia realizar o alinhamento caso o arquivo de entrada tivesse algum caractere indevido, ou seja, fora do alfabeto de nucleotídeos ou aminoácidos. Esse era um fator agravante, visto que nesses casos a ferramenta não conseguia executar os processos, conforme pode ser visto na Figura 31.

**Figura 31.** Erro no alinhamento antes da implementação do mecanismo de limpeza.

```
Simbolo invalido para NUCLEIC_ACID
Sequencia: A2F
Simbolo: N
Posicao: 9
```

Fonte: Elaborado pelo autor.

Após a implementação do algoritmo de limpeza, a ferramenta é capaz de executar esse tipo de análise, visto que o algoritmo mostrado no presente trabalho faz uma busca pelos caracteres inválidos e os substitui por um caractere válido.

# CAPÍTULO 5 - CONCLUSÃO

## 5.1 Conclusões gerais

A partir dos resultados apresentados no Capítulo 4, observa-se que com a implementação da interface gráfica para a ferramenta LOCQSPEC foi possível propiciar maior usabilidade ao usuário, de modo a oferecer maior entendimento das tarefas realizadas, o que torna o usuário mais familiarizado com o sistema em comparação com a versão original da ferramenta, que era executada por meio de linhas de comando. Deste modo, pode-se afirmar que a principal contribuição com relação à implementação da interface gráfica foi facilitar o uso de uma ferramenta comumente utilizada por usuários que, em sua maioria, não são habituados a executar processos em linhas de comando.

Além disso, observa-se que com o mecanismo de pré-processamento para limpeza de sequências, foi possível obter resultados mais precisos, visto que grande parte da eficácia da ferramenta depende da qualidade do arquivo de entrada. Portanto, com os refinamentos apresentados no presente trabalho, foi possível obter resultados com maior significância biológica, o que contribui com análises mais precisas por parte dos biólogos.

## 5.2 Trabalhos futuros

Além das implementações e melhorias mostradas no presente trabalho, pretende-se realizar como trabalhos:

- A implementação de novas funcionalidades à LOCQSPEC;
- A adição de um Manual do Usuário junto à ferramenta, conforme necessidade observada nos resultados obtidos.

# REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTS, B., BRAY, D., HOPKIN, K., JOHNSON, A., LEWIS, J., RAFF, M. & WALTER, P. (2016). *Fundamentos da Biologia Celular-3ª edição*. Porto Alegre: Artmed Editora, 2016.

BARBOSA S.D.J., SILVA B.S.. Interação Humano-Computador. São Paulo: Elsevier Editora Ltda, 2010.

BATEMEN, A. Editorial: What makes a good database? Nucleic Acids Research, Database issue; v. 35, n. 10, p. 1.093-105, 2007.

BAWONO, P. et al. Multiple Sequence Alignment. Bioinformatics: Volume I: Data, Sequence Analysis, and Evolution, p. 167-189, 2017.

CHOU, H.; HOLMES, M. H. DNA sequence quality trimming and vector removal. Bioinformatics, v.17, n.12, p. 1093-1104, 2001.

COOPER, G. M.; HAUSMAN, R. E. A Célula-: Uma Abordagem Molecular. Artmed Editora, 2016.

COUTO, D.; ZIPFEL, C. Regulation of pattern recognition receptor signalling in plants. Nature Reviews Immunology, 2016.

EDGAR, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research, v. 32, n. 5, p. 1792-1797, 2004.

GIBAS, C.; JAMBECK, P. Desenvolvendo Bioinformática. 2.ed. Rio de Janeiro: Campus, 2001.

HEWETT, T. T. et al. ACM SIGCHI curricula for human-computer interaction. ACM, 1992.

HUSSEIN, A. M.; RASHID, N. A.; ABDULAH, R.. Parallelisation of maximal patterns finding algorithm in biological sequences. In: Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on. IEEE, 2016. p. 227-232.

JO, S., KIM, T., IYER, V. G., IM, W. "CHARMM-GUI: a web-based graphical user interface for CHARMM." *Journal of computational chemistry* 29.11 (2008): 1859-1865.

KATOH, K.; STANDLEY, D. M. A simple method to control over-alignment in the mafft multiple sequence alignment program. Bioinformatics, Oxford Univ Press, p. btw108, 2016.

KATOH, K. et al. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. Nucleic acids research, Oxford Univ Press, v.30, n.14, p. 3059–3066, 2002.

LEMOS, M.; ARAGÃO, M. V. S. P.; CASANOVA, M. A. Padrões em Biossequências. In: PUC-RioInf.MCC17/03. Rio de Janeiro, 2003.

MARUCCI, E. A.. Paralelização da ferramenta de alinhamento de sequências MUSCLE para um ambiente distribuído. 2009.

MARUCCI, E. A.; ZAFALON, G. F. D.; JARDIM, A. C. G.; YAMASAKI, L. H. T.; BITTAR, C.; RAHAL, P.; MACHADO, J. M. Routine libraries for pattern recognition in quasispecies. In: Genetics and Molecular Research 7 (3): 970-981, 2008.

MENG, F. et al. Analysis of Quasispecies of Avain Leukosis Virus Subgroup J Using Sanger and High-throughput Sequencing. Virology Journal, v. 13, n. 1, p. 112, 2016.
NIELSEN, J. Usability Engineering. New York, NY, Academic Press, 1993.

NIU, N., STROULIA, E., & EL-RAMLY, M. Understanding web usage for dynamic web-site adaptation: A case study. In *Web Site Evolution, 2002. Proceedings. Fourth International Workshop on* (pp. 53-62). IEEE.

O'DONOGHUE, S. I. et al. Visualizing biological data—now and in the future. Nature methods, v.7, p. S2-S4, 2010.

OHTSUBO, Y., et al. "*GenomeMatcher*: a graphical user interface for DNA sequence comparison." *BMC bioinformatics* 9.1 (2008): 376.

PREECE J, ROGERS Y, SHARP H, BENYON D, HOLLAND S, CAREY T. Human-Computer Interaction. Michigan: Addison-Wesley Publishing Company; 1994.

PROSDOCIMI, F. et al. Bioinformática: Manual do Usuário. In: Biotecnologia Ciência e Desenvolvimento, v. 29, p. 12-25, 2003.

RIBEIRO, D. "Ferramentas de bioinformática: manipulação de sequências e recuperação de regiões flanqueadoras de um alvo."

RIGON, L. G.; SILVA, E. F.; ALVES, V. M.. Avaliação da Interface de um Sistema de Informação: Uma análise baseada no checklist, considerando a avaliação preditiva e prospectiva entre o desenvolvedor e os usuários. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, v.1, n.4, 2016.

SIEVERS, F. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. Molecular systems biology, EMBO Press, v.7, n.1, p. 539, 2011.

SIEVERS, F., HIGGINS, D. G. Clustal Omega, accurate alignment of very large numbers of sequences. *Multiple sequence alignment methods*, 105-116. 2014.

SCHULENBURG, R.; PEZZINI, M. R.; Sistematização de conceitos ergonômicos e semióticos para projetos de interfaces gráficas do usuário, 2013.

WÅNGGREN, M.; BILLETER, M.; KEMP, G. J. L. "Computational protein modelling based on limited sets of constraints." *Workshop on Constraint-Based Methods for Bioinformatics (WCB'16)*. 2016.

WETTENHALL, J. M.; SMYTH, G. K. "*limmaGUI*: a graphical user interface for linear modeling of microarray data." *Bioinformatics* 20.18 (2004): 3705-3706.

YAMAMOTO, T. T. I.; BANDIERA-PAIVA, P.; ITO, M.. Avaliação da usabilidade de interface gráfica de dois sistemas de gestão hospitalar. Journal of Health Informatics, v.7, n.2, 2015.

ZAFALON, G. et al. A parallel approach of coffee objective function to multiple sequence alignment. In: IOP PUBLISHING. Journal of Physics: Conference Series. [S.l.], 2015. v. 633, n. 1, p. 012084.