PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS MÉDICAS E DA VIDA
CURSO DE CIÊNCIAS BIOLÓGICAS

David Daniel Ferreira dos Santos

Desenvolvimento de um pipeline de bioinformática para análise de redes de co-expressão gênica com dados públicos de sequenciamento: estudo de caso com *Spodoptera frugiperda* (J. E. Smith, 1797)

Goiânia, Goiás

2024

David Daniel Ferreira dos Santos

Desenvolvimento de um pipeline de bioinformática para análise de redes de co-expressão gênica com dados públicos de sequenciamento: estudo de caso com *Spodoptera frugiperda* (J. E. Smith, 1797)

Monografia apresentada ao Departamento de Biologia da Pontifícia Universidade Católica de Goiás como requisito parcial para obtenção do Título de Bacharel em Biologia.

Orientadora: Profª. Dra. Mariana Pires de Campos Telles
Co-Orientadora: Profª. Dra. Renata de Oliveira Dias (UFG)

Goiânia, Goiás

2024

Dedico esta conquista, a minha namorada e futura esposa que me apoiou, incentivou e acolheu, nos momentos que mais precisei. E aos meus pais que sempre me apoiaram e incentivaram.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me permitiu chegar até aqui, me deu oportunidades e me agraciou com todas as minhas conquistas e desempenho, por meio de suas lições de amor, sabedoria, respeito, compaixão e empatia.

Agradeço aos meus familiares, que me apoiaram e contribuíram, diretamente ou indiretamente, na minha vida acadêmica e pessoal.

Aos professores do curso de Ciências Biológicas, que me ensinaram, apoiaram, prestaram ajuda e esclareceram dúvidas quando se mostravam necessárias.

Agradeço imensamente à Prof. Dra. Renata de Oliveira Dias pelas orientações, assistência, motivação, paciência e conselhos.

À minha orientadora, Prof. Dra. Mariana Pires de Campos Telles, por ter me aceitado como orientando. Também agradeço pelos seus ensinamentos e indicações.

Aos amigos que fiz na faculdade, Matheus, Kairo, Caio, Daniel e Gabriel. E também às minhas amigas Cimara e Giovanna.

Ao meu melhor amigo, Caio César, que me acompanhou durante toda minha jornada acadêmica do ensino fundamental à faculdade.

Aos membros da banca, pelas contribuições que certamente enriqueceram este trabalho.

Ademais, agradeço a todos que participaram desta minha longa jornada, que estiveram presentes e acompanharam meu desenvolvimento pessoal, acadêmico e profissional.

RESUMO

Com a crescente redução nos custos de sequenciamento de nucleotídeos, a disponibilidade de dados oriundos de trabalhos de RNA-seq em bancos de dados públicos vem crescendo exponencialmente. No entanto, ainda são poucos os trabalhos que exploram esses recursos, principalmente, pensando em abordagens integrativas, tais como a predição de redes de co-expressão gênica. Diante disso, este trabalho propôs a elaboração de um pipeline de Bioinformática para a obtenção, filtragem e análise de dados de RNA-seq publicamente disponíveis no banco de dados SRA do NCBI para a predição de redes de co-expressão gênica, utilizando a espécie *Spodoptera frugiperda* como modelo, dada sua relevância como praga agrícola de grande impacto para a agricultura brasileira. Por meio desse pipeline, 839 conjuntos de dados foram obtidos do banco de dados do SRA e submetidos a uma série de filtros, para a remoção de sequências e conjuntos de dados de baixa qualidade ou apresentando riscos de contaminação, após os quais foram selecionados 641 conjuntos de dados. Além desses filtros, precisaram ser removidos das análises 339 conjuntos de dados sem metadados informando o tecido ou a fase de vida do inseto amostrado. Por fim, 10 condições foram selecionadas por conterem mais do que 12 amostras cada, resultando em um total de 276 amostras sendo de fato incorporadas nas análises de co-expressão. Nas análises de WGCNA, sete módulos de co-expressão foram preditos, incluindo um módulo composto por genes mais expressos em média no intestino e corpo inteiro de larvas de *S. frugiperda.* Nossos resultados demonstram que é possível utilizar dados públicos de sequenciamento de RNA para predizer redes de co-expressão e propõe um pipeline para a filtragem das sequências, seleção das amostras e análise desses dados. Por fim, ressalta-se a importância da disponibilização e a completa descrição desses conjuntos de dados em bancos de dados públicos, tais como SRA, para o seu uso em abordagens integrativas tais como a utilizada neste trabalho.

Palavras-chave: pragas agrícolas; inseto; biologia de sistemas; genômica funcional; RNA-seq

ABSTRACT

As the costs associated with nucleotide sequencing have decreased, the quantity of data available from RNA-seq work in public databases has increased exponentially. However, there are still few studies that exploit these resources, especially when considering integrative approaches such as predicting gene co-expression networks. In light of the aforementioned considerations, this work proposed the development of a Bioinformatics pipeline for obtaining, filtering and analyzing RNA-seq data publicly available in the NCBI SRA database for the prediction of gene co-expression networks, using the *Spodoptera frugiperd*a species as a model, given its relevance as an agricultural pest with a major impact on Brazilian agriculture. Through this pipeline, 839 datasets were obtained from the SRA database and subjected to a series of filters to remove sequences and datasets of low quality or presenting risks of contamination, after which 641 datasets were selected. In addition to these filters, 339 datasets without metadata informing the tissue or life stage of the insect sampled had to be removed from the analysis. Finally, 10 conditions were selected because they contained more than 12 samples each, resulting in a total of 276 samples actually being incorporated into the co-expression analyses. In the WGCNA analyses, seven co-expression modules were predicted, including a module composed of genes most expressed on average in the gut and whole body of *S. frugiperda* larvae. Our results demonstrate that it is possible to use public RNA sequencing data to predict co-expression networks and propose a pipeline for filtering sequences, selecting samples and analyzing these data. Finally, we emphasize the importance of making these datasets available and fully describing them in public databases, such as SRA, for their use in integrative approaches such as the one used in this work.

Key words: agricultural pests; insect; systems biology; functional genomics; RNA-seq

SUMÁRIO

1 INTRODUÇÃO

A lagarta-do-cartucho do milho, *Spodoptera frugiperda* (SMITH, 1797) (Lepidoptera: Noctuidae), é uma das principais pragas do milho e de outras culturas (VALICENTE, 2015). Durante o seu ciclo de vida, essa mariposa passa por 4 estágios, sendo eles ovo, larva (popularmente conhecida como lagarta), pupa e adulto (ASHOK *et al*., 2020). Ademais, em sua fase larval, pode-se haver de 5 a 10 instares (fases) (LEIDERMAN; SAUER, 1953; CAMPOS, 1970; ESCALANTE, 1974; ALI *et al*., 1990, *apud* KENIS, 2023), dependendo de diferentes condições como temperatura, fotoperíodo, quantidade e qualidade dos alimentos, umidade, densidade populacional, condição física, herança e sexo (ESPEK, *et al.*, 2007).

Considerando seus diferentes estágios e fases da vida, esta praga acaba sendo uma grande ameaça para o cultivo do milho, visto que ela se alimenta da planta em todo seu ciclo de desenvolvimento, podendo causar uma perda de 17% a 38,7% da produção (CRESPO *et al*, 2021).

O controle biológico da espécie geralmente é feito por meio de plantações geneticamente modificadas que expressam toxinas da bactéria *Bacillus thuringiensis* (milho Bt), ou por meio do uso de pesticidas químicos (CRESPO *et al*, 2021). Adicionalmente, também são utilizados métodos de controle biológico por meio da aplicação do parasita de ovos do gênero *Trichogrammma* e bioinseticidas à base de baculovírus (VALICENTE, 2015). No entanto, vários estudos têm demonstrado uma crescente resistência desses insetos a diferentes métodos de controle (NASCIMENTO *et al*., 2016; NASCIMENTO *et al*., 2022; NIZ *et al*., 2020; BOLZAN *et al*., 2019; CARVALHO *et al*., 2013; DIEZ-RODRÍGUEZ; OMOTO, 2001; GARLET *et al*., 2021; LIRA *et al*., 2020; MURARO *et al*., 2021).

Para lidar com esses desafios, novos compostos estão sendo pesquisados, tais como biodefensivos à base de óleos essenciais e extratos vegetais, entomopatógenos, entre outros (JARA *et al.*, 2019). Diante da resistência aos métodos de controle tradicionais e a necessidade de explorar novos métodos, diferentes pesquisas têm sido conduzidas para tentar melhor compreender a biologia desses insetos, incluindo as suas respostas moleculares a esses diferentes métodos de controle (KANNO *et al*., 2024; SHI *et al*., 2023).

Dentre os métodos disponíveis para o estudo das respostas moleculares dos organismos mediante diferentes condições biológicas, o sequenciamento de RNA (RNA-seq) emergiu como um método diferencial, por permitir a análise do transcriptoma das espécies em diferentes condições experimentais (RAPAPORT *et al*., 2013), oferecendo, desta forma, uma perspectiva mais aprofundada e completa da expressão gênica do que as tecnologias precedentes (XU *et al*., 2014; MARTIN *et al*., 2014). Esta técnica apresenta diversos benefícios, tais como a identificação de novos componentes estruturais de genes, maior sensibilidade em relação à tecnologia de microarranjos e a oportunidade de estudar fenômenos como fusões gênicas e expressão alélica específica (WANG *et al*., 2008).

Dados de RNA-seq também têm sido frequentemente aplicados em análises de Rede de Co-expressão Gênica Ponderada (*Weighted Gene Co-expression Network Analysis* - WGCNA), um método computacional frequentemente utilizado em vários contextos biológicos para identificar módulos de genes co-expressos e suas conexões com características fenotípicas (LANGFELDER *et al.*, 2008). Nos últimos anos, esta técnica tem recebido uma atenção considerável, principalmente na área de identificação de biomarcadores e identificação de alvos terapêuticos (JIANG *et al*., 2022).

O WGCNA é capaz de identificar padrões de correlação entre genes, com base em dados de expressão gênica em grande escala (FULLER *et al*., 2011). Além da identificação de alvos terapêuticos (ZHAO *et al*., 2021), ao associar os módulos genéticos a características fenotípicas ou tratamentos, esse procedimento também torna possível identificar genes centrais de processos biológicos específicos (DAI *et al.*, 2022; ZHAO *et al.*, 2016). Essas análises de correlação podem, portanto, auxiliar na descoberta de genes centrais para explicar a crescente resistência desses insetos aos métodos de controle tradicionais e a prospectar novas moléculas chave para a atuação de métodos alternativos de controle, tais como os biodefensivos, aumentando a sua eficácia. No entanto, para realizar análises de WGCNA, uma grande quantidade de dados de sequenciamento de RNA precisa estar disponível, com no mínimo 6 replicatas por condição e pelo menos 12 amostras quando necessário a identificação de genes diferencialmente expressos sendo recomendadas para a realização desses estudos (POPLOWSKI *et al.*, 2018; SCHURCH *et al.*, 2016). Diante disso, bancos de dados públicos de sequências tornam-se essenciais por aumentar o número de amostras disponíveis e reduzir os custos dessas análises.

O Sequence Read Archive (SRA), inserido no NCBI, é o principal banco de dados de leituras de sequenciamento, tanto de DNA quanto RNA. A disponibilidade desses dados no SRA está crescendo exponencialmente à medida que mais conjuntos de dados estão sendo gerados por pesquisadores ao redor do mundo e tornando-se acessíveis. Segundo Marche (2023), o SRA já atingiu 45 petabytes de sequências brutas, com a quantidade de nucleotídeos depositados duplicando a cada dois anos. Para facilitar o acesso e o processamento desses dados, o SRA Toolkit foi desenvolvido, permitindo o download, conversão e manipulação das sequências de forma eficiente. Portanto, é possível a utilização dos dados que estão sendo gerados para análises mais robustas e novas estratégias podem ser criadas para explorar melhor esses recursos.

Diversas estratégias têm sido criadas para otimizar os pipelines de Bioinformática, incluindo as análises de RNA-seq, tais como o nf-core. Ao utilizar o nf-core, uma estrutura projetada para a execução e desenvolvimento de análises colaborativas, revisadas por pares, é possível garantir uma resolução de problemas de padronização, portabilidade e reprodutividade (EWELS *et al*., 2020). Ele utiliza do Nextflow, uma ferramenta de gerenciamento de workflows que permite a criação de pipelines compatíveis com diferentes infraestruturas, como computadores locais, sistemas de alta performance e provedores de nuvem. Além disso, possui integração com gestores de pacotes, como Conda, e de conteinerização, como Docker e Singularity (EWELS *et al*., 2020).

Especificamente dentro do nf-core, são disponibilizados pipelines para diferentes finalidades, como o nf-core/rnaseq, o qual possui uma abordagem metodologicamente robusta para a análise de dados de sequenciamento de RNA (RNA-seq) obtidos de organismos com um genoma de referência e anotação gênica, integrando diversas ferramentas bioinformáticas que garantem a precisão, eficiência e reprodutibilidade das análises. Além disso, através da ampla quantidade de ferramentas, é possível a identificação de vieses experimentais, como os efeitos de lote, que podem comprometer os dados devido a variações técnicas, incluindo tempos diferentes de experimento, reagentes, instrumentos e outros fatores, causando conclusões imprecisas (YANG, 2021; GOH, 2017).

Para garantir uma precisão e qualidade dos dados, a aplicação de filtros é essencial, como a escolha do layout de biblioteca pareada, que está diretamente relacionado à tecnologia de sequenciamento e à forma como as leituras (*reads*) são geradas durante a análise. Nesse tipo

de sequenciamento, ocorre a leitura de ambas as extremidades de um fragmento, resultando em dois conjuntos de leituras, um para cada extremidade, o que proporciona o dobro de informações para as análises.

Ademais, durante a etapa de sequenciamento, são utilizados adaptadores, sequências curtas de DNA sintético projetadas para se ligarem às extremidades dos fragmentos de DNA. Esses adaptadores facilitam a amplificação e a identificação das sequências (STRÖHER, 2018), porém devem ser removidos durante a análise para garantir maior qualidade dos dados.

A ampla disponibilidade de dados genômicos possibilita novas descobertas apenas utilizando de informações públicas, reduzindo drasticamente os custos envolvidos nessas análises e ampliando a capacidade amostral. Considerando a grande quantidade de dados de sequenciamento de RNA disponíveis para *S. frugiperda* e outras espécies pragas da agricultura brasileira e a subutilização de tais dados, o presente trabalho objetivou elaborar e aplicar um pipeline com foco no uso de banco de dados públicos para realizar análises integrativas de redes de co-expressão.

2 OBJETIVOS

2.1 Objetivo Geral

Elaborar um pipeline de Bioinformática para a exploração de dados públicos de sequenciamento de RNA em análises de redes de co-expressão gênica, utilizando *S. frugiperda* como modelo.

2.2 Objetivos Específicos:

Obter dados brutos de sequenciamento de RNA publicamente disponíveis no banco de dados SRA do NCBI;

Elaborar uma estratégia para fazer o download dos dados diretamente em um servidor local;

Desenvolver um pipeline para realizar análises de expressão gênica de forma a otimizar o espaço de armazenamento dos dados;

Realizar as análises de expressão gênica utilizando os dados obtidos;

Criar filtros para selecionar conjuntos de dados com boa qualidade de sequenciamento e alinhamento ao genoma de referência da espécie;

Realizar as análises de rede de co-expressão gênica.

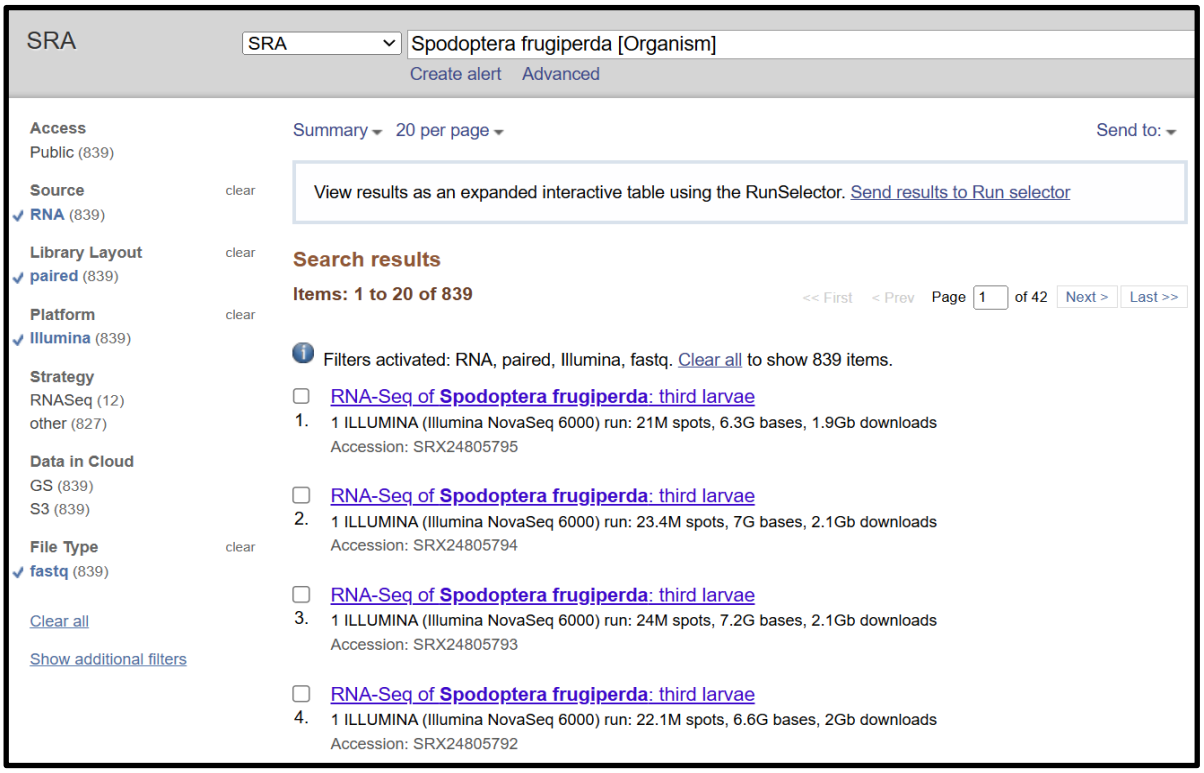
3 METODOLOGIA

3.1 Obtenção dos conjuntos de dados

A obtenção do genoma de referência da espécie *Spodoptera frugiperda* foi realizada em 01/07/2024, utilizando o banco de dados *Genome* do *National Center for Biotechnology Information* (NCBI). Para isso, foi selecionada a versão do genoma depositada no banco de dados do RefSeq, a qual possui o número de acesso GCF_023101765.2. As sequências genômicas foram obtidas no formato Fasta, enquanto as anotações foram obtidas nos formatos GTF e GFF. Este procedimento garantiu a existência de dados de referência para a execução das análises de RNA-seq.

Para a obtenção do conjunto de dados de sequenciamento, foi utilizado o banco de dados *Sequence Read Archive* (SRA) do NCBI, adotando o critério de busca "*Spodoptera frugiperda* [Organism]", o que assegurou a inclusão exclusiva de dados pertinentes ao referido organismo. A busca foi realizada em 24/07/2024. Filtros adicionais foram aplicados (Figura 1), abrangendo o tipo de sequenciamento como RNA, o layout da biblioteca definido como pareado (*paired*), a plataforma de sequenciamento, que foi Illumina, e o tipo de arquivo no formato fastq. Após a aplicação desses critérios, foram obtidos um total de 839 conjuntos de dados.

Figura 1 - Obtenção do conjunto de dados brutos de sequenciamento de RNA do NCBI

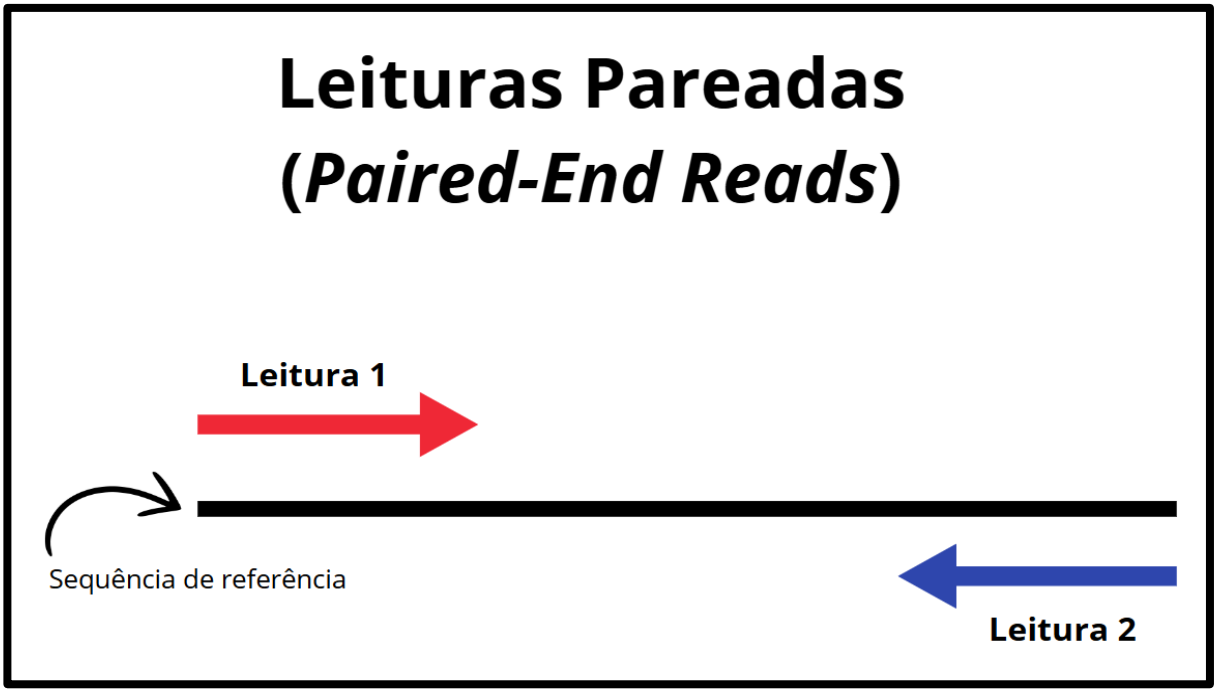


Fonte: Elaborado pelo autor.

Posteriormente, estes 839 resultados foram exportados em formato de Lista de Acessos (*Accession List*) (Apêndice A). Ela foi utilizada como arquivo de entrada para um script *Bash* que foi criado para automatizar o download dos dados do SRA e sua conversão para o formato fastq em um *loop* contínuo (Apêndice B). Desta maneira, o script criava uma lista de arquivos baixados e uma lista de arquivos novos para evitar duplicação, baixando os dados com o comando prefetch e convertendo-os com fasterq-dump, ambos parte do SRA Toolkit. Em caso de falhas no download, o script tentava novamente até que fosse bem-sucedido. Ao final de cada ciclo, a lista de arquivos baixados era atualizada, garantindo que apenas os itens restantes fossem processados, e o ciclo continuava até que todos os dados tivessem sido baixados com sucesso.

Após o download, os arquivos foram verificados manualmente, para observar se houve a criação dos dados brutos de sequenciamento (.fastq) com os dois sentidos de leitura, com os formatos *_1.fastq e *_2.fastq. Essa verificação foi realizada para confirmar se as sequências estavam no formato de sequenciamento emparelhado (*paired-end)* (Figura 2).

Figura 2 - Sequenciamento paired-end



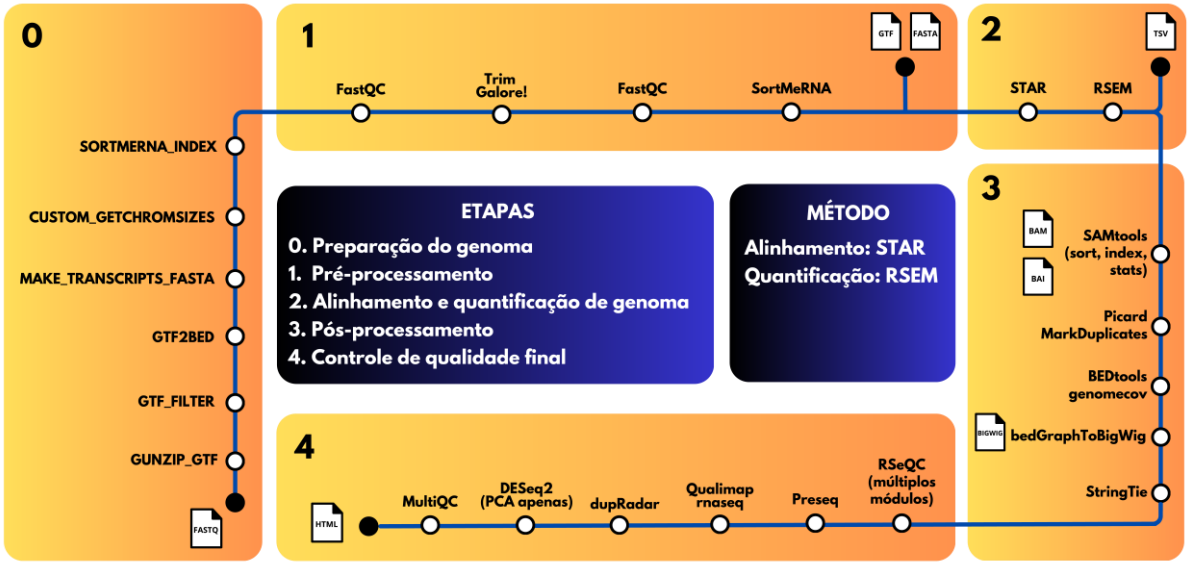
Fonte: Elaborado pelo autor.

3.2 Análises de expressão gênica

Devido ao grande número de conjuntos de dados de sequenciamento, foi elaborada uma estratégia por meio de um script em *Bash* (Apêndice C) para dividir os conjuntos de dados em grupos (*Runs*) com base em um tamanho total máximo definido pelo usuário, e uma lista contendo os arquivos *_1.fastq e *_2.fastq e seus respectivos tamanhos. Desta maneira, à medida que o script que foi elaborado lê o arquivo de entrada linha por linha e verifica o tamanho, há a criação de arquivos de agrupamento contendo os nomes dos conjuntos de dados que pertencem a este grupo, levando em conta os IDs semelhantes e os sufixos _1 e _2. Caso o tamanho acumulado dos arquivos no agrupamento exceda o tamanho máximo especificado, ele cria um novo grupo.

A realização da análise de expressão gênica foi feita utilizando o pipeline de bioinformática nf-core/rnaseq, que possui diferentes etapas e subetapas (Figura 3). Para facilitar e automatizar a execução do pipeline para os diferentes grupos de conjuntos de dados (*Runs*) gerados na etapa anterior, foi elaborado um script em *Bash* (Apêndice D). Desta maneira, o pipeline utiliza-se do genoma de referência, de uma planilha de amostras e dos arquivos fastq como entrada, executa controles de qualidade (QC), corte, pseudo-alinhamento e produz uma matriz de expressão gênica e um extenso relatório QC.

Figura 3 - Principais etapas e subetapas realizadas no pipeline nf-core/rnaseq



Fonte: Elaborado pelo autor.

Anterior à execução do pipeline nf-core/rnaseq, foi gerado o índice do genoma através do STAR (DOBIN *et al*., 2013), que serve como estrutura de dados para o mapeamento das leituras de RNA-seq contra o genoma de referência, utilizando o arquivo do genoma de referência em fasta e a anotação gênica em gtf.

Inicialmente, o pipeline nf-core/rnaseq realiza a preparação (PREPARE_GENOME) do genoma de referência para a etapa de alinhamento das leituras de sequenciamento. A subetapa GUNZIP_GTF utiliza a ferramenta gunzip para descompactar o arquivo GTF (neste caso: Sfru.gtf.gz), resultando no arquivo descompactado (Sfru.gtf). Este arquivo contém as anotações gênicas necessárias para as análises subsequentes. Em seguida, na subetapa GTF_FILTER, o arquivo GTF descompactado é filtrado para garantir sua consistência com o genoma de referência (Sfru.fna). Esta filtragem, realizada por scripts internos do pipeline, remove entradas

inválidas ou inconsistentes, produzindo um arquivo corrigido (Sfru.filtered.gtf), que é posteriormente convertido para o formato BED (GTF2BED) utilizando ferramentas internas de conversão de formatos. O formato BED facilita a manipulação e visualização de intervalos genômicos em etapas futuras.

Prosseguindo, a subetapa MAKE_TRANSCRIPTS_FASTA utiliza o genoma de referência (Sfru.fna) para gerar um arquivo FASTA de transcritos (transcripts.fasta). Este arquivo é essencial para a quantificação da expressão gênica, servindo como referência para ferramentas como o RSEM (RNA-Seq *by Expectation-Maximization*) (LI *et al*., 2011). Paralelamente, a subetapa CUSTOM_GETCHROMSIZES extrai os tamanhos dos cromossomos a partir do genoma de referência, produzindo um arquivo denominado de chrom_sizes.txt. Este arquivo é fundamental para a criação de arquivos BigWig, que são utilizados para visualizar a cobertura de sequências em navegadores genômicos.

Uma etapa crucial no posterior pré-processamento das leituras de RNA-seq é a construção do índice para a remoção de RNA ribossomal, realizada por PREPARE_GENOME:SORTMERNA_INDEX. Utilizando a ferramenta SortMeRNA (KOPYLOVA *et al*., 2012), o pipeline constrói um índice a partir das sequências de RNA ribossomal especificadas no arquivo de manifesto (ribo_database_manifest). Este índice permite a filtragem eficiente de leituras derivadas de RNA ribossomal nas etapas subsequentes, reduzindo a contaminação e focando a análise em mRNA de alta qualidade.

Com o genoma de referência preparado, o pipeline avança para o pré-processamento das leituras fastq. A primeira subetapa envolve a verificação de qualidade das leituras, utilizando o FastQC (ANDREWS, s.d.). O FastQC gera relatórios detalhados (*.fastqc.html e *.fastqc.zip) que avaliam métricas como a qualidade das bases, a distribuição de sequências de adaptadores e a uniformidade da cobertura ao longo das leituras. Identificar e corrigir potenciais problemas nesta fase é essencial para garantir a integridade dos dados antes do alinhamento.

Posteriormente, a etapa de trimming de adaptadores e filtragem de qualidade é realizada pelo Trim Galore! (KRUEGER *et al*., 2021). O Trim Galore! remove sequências de adaptadores e bases de baixa qualidade das leituras fastq, aprimorando a qualidade geral das sequências para um alinhamento mais preciso. Após esse processo, um novo passo de avaliação da qualidade com o FastQC é realizado sobre as leituras já trimadas, assegurando que as sequências

resultantes atendam aos padrões de qualidade necessários para as etapas subsequentes. Em seguida, essas leituras são filtradas para remover RNA ribossomal utilizando o SortMeRNA. O SortMeRNA, com o índice previamente construído, filtra eficientemente as leituras, gerando arquivos fastq não ribossomais (*.fastq.gz), que serão então utilizados nas etapas de alinhamento e quantificação.

A quantificação da expressão gênica pode ser realizada por diferentes ferramentas, incluindo o RSEM, utilizado neste trabalho. Nesta etapa, as leituras são primeiramente alinhadas ao genoma de referência por meio do STAR, invocado internamente pelo RSEM, para então estimar a abundância da expressão dos genes e de suas isoformas. O RSEM emprega um modelo probabilístico para distribuir as leituras entre os diferentes transcritos, produzindo arquivos de resultados (*.genes.results e *.isoforms.results) que contêm contagens de expressão e valores normalizados de TPM (*Transcripts Per Million*).

Após a quantificação, os arquivos BAM resultantes dos alinhamentos são ordenados (sort) utilizando o SAMtools (LI *et al*., 2009), que organiza as leituras por coordenadas genômicas, facilitando o acesso e a análise subsequente. A indexação dos arquivos BAM ordenados é realizada pela subetapa de index, que gera arquivos de índice (*.sorted.bam.bai ou *.sorted.bam.csi). Estes índices permitem o acesso rápido a regiões específicas do genoma durante a visualização e análise dos dados.

Além disso, estatísticas (stats) detalhadas do alinhamento são geradas pelo SAMtools através das subetapas SAMTOOLS_STATS, SAMTOOLS_FLAGSTAT e SAMTOOLS_IDXSTATS. Estas estatísticas fornecem métricas abrangentes sobre a qualidade do alinhamento, incluindo o número de leituras mapeadas, não mapeadas, duplicadas e distribuídas por cromossomos. Tais informações são fundamentais para avaliar a eficiência do alinhamento e identificar possíveis problemas como baixa taxa de mapeamento ou contaminação cromossômica.

Para mitigar vieses introduzidos pela amplificação por PCR, o pipeline utiliza o Picard MarkDuplicates (BROARD INSTITUTE, 2019). Esta ferramenta identifica e marca leituras duplicadas no arquivo BAM, evitando a contaminação dos dados de expressão gênica por redundâncias técnicas. Após a marcação de duplicatas, o arquivo BAM é novamente indexado

e submetido a novas estatísticas pelo SAMtools, assegurando que a limpeza de duplicatas não comprometa a qualidade do alinhamento final.

Para a criação de arquivos de cobertura BigWig, o pipeline emprega o BEDTools (genomecov) (QUINLAN *et al*., 2010) para calcular a cobertura do genoma nas fitas direta e reversa do DNA, resultando em arquivos BEDGraph (*.forward.bedGraph e *.reverse.bedGraph). Posteriormente, esses arquivos BEDGraph são convertidos para o formato BigWig por meio do bedGraphToBigWig. Os arquivos BigWig são gerados como parte do pipeline do nf-core/rnaseq para possibilitar a visualização da cobertura genômica em navegadores genômicos, sendo otimizados para representar a distribuição das leituras ao longo do genoma.

Em uma etapa subsequente do pipeline, a montagem e quantificação de transcritos podem ser conduzidas pela ferramenta StringTie (KOVAKA *et al*., 2019). Diferentemente do RSEM, que partiu de um conjunto pré-definido de transcritos para quantificação, o StringTie pode montar transcritos de novo ou ajustá-los guiado pela anotação de referência a partir das leituras alinhadas. Como resultado, são gerados arquivos como *.transcripts.gtf e *.gene_abundance.txt, permitindo a identificação de novas variantes de splicing e a quantificação de isoformas específicas.

Para assegurar a qualidade dos dados, o pipeline incorpora várias ferramentas de controle de qualidade extensivas. O RSeQC (WANG *et al*., 2012) executa múltiplas análises específicas para dados de RNA-seq, incluindo a geração de estatísticas detalhadas do arquivo BAM (RSEQC_BAMSTAT), avaliação da distância interna entre leituras emparelhadas (RSEQC_INNERDISTANCE), inferência da orientação das leituras (strandedness) (RSEQC_INFEREXPERIMENT), análise de junções de splicing (RSEQC_JUNCTIONANNOTATION e RSEQC_JUNCTIONSATURATION), distribuição das leituras em relação às características genômicas (RSEQC_READDISTRIBUTION) e avaliação da duplicação de leituras (RSEQC_READDUPLICATION). Essas análises fornecem uma visão abrangente da integridade e qualidade dos dados de RNA-seq, identificando possíveis problemas como amostras degradadas ou contaminação por RNA não ribossomal.

Adicionalmente, o Qualimap (OKONECHNIKOV *et al*., 2016) avalia a qualidade do alinhamento e a cobertura das leituras em relação às características genômicas, identificando possíveis vieses ou problemas no sequenciamento. O dupRadar (SAYOLS *et al*., 2016) relaciona a taxa de duplicação de leituras com os níveis de expressão gênica, permitindo a identificação de duplicações técnicas excessivas que podem enviesar a quantificação de expressão.

A análise de complexidade da biblioteca é conduzida pelo DESeq2 (LOVE *et al*., 2014), que realiza análises de componentes principais (PCA) e gera mapas de calor de similaridade entre amostras. Estas análises auxiliam na identificação de agrupamentos de amostras, efeitos de lote ou outros vieses experimentais, assegurando a reprodutibilidade e a robustez das análises de expressão gênica.

Finalmente, a ferramenta MultiQC (EWELS *et al*., 2016) agrega todos os resultados de controle de qualidade gerados pelas diversas etapas do pipeline em um único relatório consolidado (multiqc_report.html). Este relatório interativo contém visualizações abrangentes e resumos das diferentes análises de qualidade, facilitando a interpretação dos dados e permitindo aos pesquisadores identificar rapidamente quaisquer problemas ou padrões nos dados de RNA-seq.

Em conclusão, o pipeline nf-core/rnaseq oferece uma abordagem integrada e rigorosa para a análise de dados de RNA-seq, combinando ferramentas especializadas para preparação do genoma de referência, pré-processamento das leituras, alinhamento, quantificação de expressão gênica e controle de qualidade. Cada software empregado desempenha um papel específico, assegurando que os dados processados e análises subsequentes, como estudos de expressão diferencial, sejam baseadas em dados robustos e confiáveis. A integração dessas etapas, juntamente com a geração de relatórios detalhados, permite um controle de qualidade, promovendo resultados coerentes e reprodutíveis em pesquisas acadêmicas.

3.3 Filtragem dos conjuntos de dados de sequenciamento para as análises de rede de co-expressão

Para garantir melhores resultados, diferentes critérios foram utilizados para filtrar os conjuntos de dados de sequenciamento do SRA que seriam utilizados nas análises de WGCNA, de acordo com o descrito a seguir.

3.3.1 Remoção de conjuntos de dados sem os dois arquivos de leitura pareados

Inicialmente, foram removidos 2 conjuntos de dados, SRR24746394 e SRR24746395, por não apresentarem ambos os sentidos de leitura *_1.fastq e *_2.fastq esperados para dados de *paired-end*. Esse filtro foi necessário, porque mesmo que tenha sido realizado o pré-selecionamento das amostras do SRA, utilizando como critério apenas buscar conjuntos que possuíssem leituras pareadas, após o download observou-se esse problema nos arquivos descritos.

3.3.2 Remoção de leituras de sequenciamento com tamanho muito reduzido

O tamanho das leituras de sequenciamento pode influenciar na qualidade do alinhamento das leituras de RNA ao genoma de referência, aumentando as chances de erro. Por isso, optou-se por selecionar amostras que obtiveram uma média de tamanho mínimo da soma dos dois pares de leituras de pelo menos 190 pb após a filtragem dos dados pelo Trimmomatic.

3.3.3 Remoção de conjuntos de dados com baixa cobertura do genoma

Para aumentar a eficácia das análises de RNA-seq, apenas amostras que tiveram pelo menos 5 milhões de leituras de sequenciamento alinhadas contra o genoma foram mantidas.

3.3.4 Remoção de conjuntos de dados com baixa taxa de alinhamento

Com o objetivo de manter apenas conjuntos de dados com boa qualidade de sequenciamento e baixas taxas de contaminação, apenas aqueles que obtiveram um mínimo de 70% de leituras de sequenciamento alinhadas contra o genoma de referência foram mantidas.

3.3.5 Remoção de conjuntos de dados com percentual de GC divergente do esperado para a espécie

Com o intuito de remover conjuntos de dados com grandes quantidades de sequências contaminantes ou erroneamente depositadas no SRA como pertencentes a indivíduos de *S. frugiperda*, o valor médio de GC (guanina-citosina) (47,2%) foi utilizado como filtro. Amostras que desviaram em pelo menos 2 vezes o desvio padrão (2,6) encontrado para o valor médio de GC, ou seja, as amostras com valores percentuais de GC menores que 41,9% e superiores a 52,4%, foram removidas.

3.3.6 Seleção das amostras com base nos tecidos a serem utilizados nas análises de WGCNA

Após a filtragem de qualidade do sequenciamento, as amostras foram selecionadas com base nas condições experimentais, adotando-se um mínimo de 10 amostras por tecido. Esse critério foi definido considerando as características do conjunto de dados e as recomendações da literatura, que indicam um mínimo de 6 replicatas por condição e pelo menos 12 amostras para a identificação de genes diferencialmente expressos (POPLOWSKI *et al*., 2018; SCHURCH *et al*., 2016).

3.4 Identificação dos genes presentes em redes co-expressão

Após a execução do pipeline nf-core/rnaseq e a realização das filtragens, aplicou-se um script em linguagem R que foi criado (Apêndice E) para identificar os módulos de co-expressão gênica relacionados aos tecidos e estágio de desenvolvimento de *S. frugiperda.*

Inicialmente, o script carrega os dados de TPM gerados pelo nf-core/rnaseq e metadados de tecido e estágio de vida associados às amostras específicas. Ele inicia a verificação da integridade das informações, removendo as amostras que estão ausentes nos metadados, assim como os genes que tiveram valores nulos de expressão nas amostras. Em seguida, alinha os metadados às amostras de expressão gênica, assegurando correlação entre ambos os dados e metadados.

Após a limpeza, os dados de expressão são normalizados pelo pacote DESeq2, que aplica uma transformação de variância para melhorar a comparação entre as amostras. Desta maneira, os genes são filtrados com base na variância, selecionando apenas aqueles com alta variabilidade de percentil 95, a fim de identificar padrões biológicos significativos.

A partir do WGCNA, é criado uma rede de co-expressão gênica, calculando o melhor *soft-thresholding power*, um parâmetro que ajuda a determinar a conectividade entre os genes, garantindo que a rede siga uma topologia livre de escala. Através deste parâmetro, o código gera uma matriz de sobreposição topológica (TOM) e *clusters* de genes em módulos, cada um identificado por uma cor específica.

Sendo identificado os módulos, o script calcula os *eigengenes* (representantes médios de cada módulo) e analisa suas relações com os tecidos e estágios de vida. Isso é demonstrado em gráficos de calor (*heatmaps*) e dendrogramas, propiciando a observação das associações entre módulos e condições experimentais. Também são gerados gráficos de violino para ilustrar a distribuição da expressão gênica por tecido, e gráficos de linha que ressaltam padrões de expressão por módulo ao longo dos tecidos e estágios de vida.

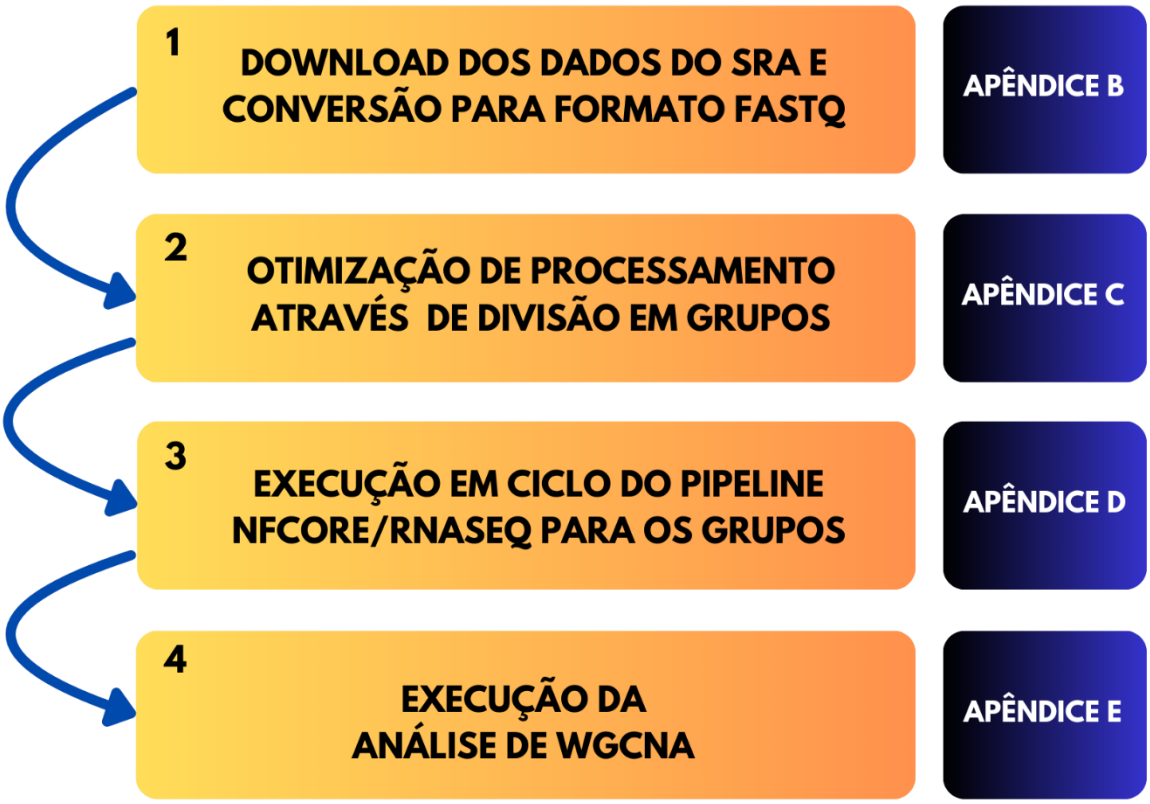
Posteriormente, os dados relevantes são exportados, tais como tabelas que relacionam genes aos módulos, e redes de co-expressão gênica que podem ser utilizadas em softwares como Cytoscape para análises mais detalhadas. Esses resultados podem identificar genes centrais e fornecer informações sobre funções biológicas que podem ser compartilhadas entre genes co-expressos.

4 RESULTADOS E DISCUSSÃO

4.1 Descrição do pipeline desenvolvido

O pipeline desenvolvido e implementado neste estudo representa uma abordagem integrativa para a análise de dados públicos de RNA-seq, com foco na construção de redes de co-expressão gênica para *Spodoptera frugiperda*. Ele foi estruturado em um fluxo de quatro etapas principais (Figura 4), permitindo a organização e o processamento automatizado de grandes volumes de dados. Ao longo das etapas, foram aplicadas filtragens essenciais, como a remoção de leituras de baixa qualidade, a exclusão de genes com valores nulos ou significativamente baixos de expressão, além da eliminação de amostras ausentes nos metadados (tecido e estágio de vida), garantindo a integridade e a qualidade dos resultados obtidos.

Figura 4 – Fluxo geral do pipeline desenvolvido



Fonte: Elaborado pelo autor.

Na primeira etapa, foi implementado um script *Bash* (Apêndice B) para automatizar o download dos dados brutos do *Sequence Read Archive* (SRA) e sua conversão para o formato FASTQ. O script utilizou os comandos prefetch e fasterq-dump, ambos do SRA Toolkit, e gerenciou os casos de duplicação, garantindo a conclusão bem-sucedida do download mesmo em situações de falha.

A segunda etapa consistiu na otimização do processamento por meio de um script Bash (Apêndice C), que agrupou os conjuntos de dados de acordo com um tamanho total máximo definido pelo usuário.

Na terceira etapa, os grupos de dados foram processados em ciclo pelo pipeline nf-core/rnaseq, amplamente utilizado para o processamento de dados de RNA-seq. Um script específico (Apêndice D) foi implementado para automatizar a execução sequencial do pipeline para cada grupo.

Por fim, na quarta etapa, realizou-se a análise de WGCNA (*Weighted Gene Co-expression Network Analysis*) utilizando um script em linguagem R (Apêndice E). O script processou os dados de expressão gênica, alinhando-os aos metadados de tecido e estágio de desenvolvimento.

4.2 Descrição do conjunto de dados utilizado

Um total de 837 conjuntos de dados de sequenciamento foram obtidos do banco de dados do SRA, utilizando os critérios de busca descritos no item 3.1. Estes conjuntos de dados incluíam amostras de diferentes tecidos e fases de vida do inseto (Tabela 1) e provenientes de diferentes países (Figura 5).

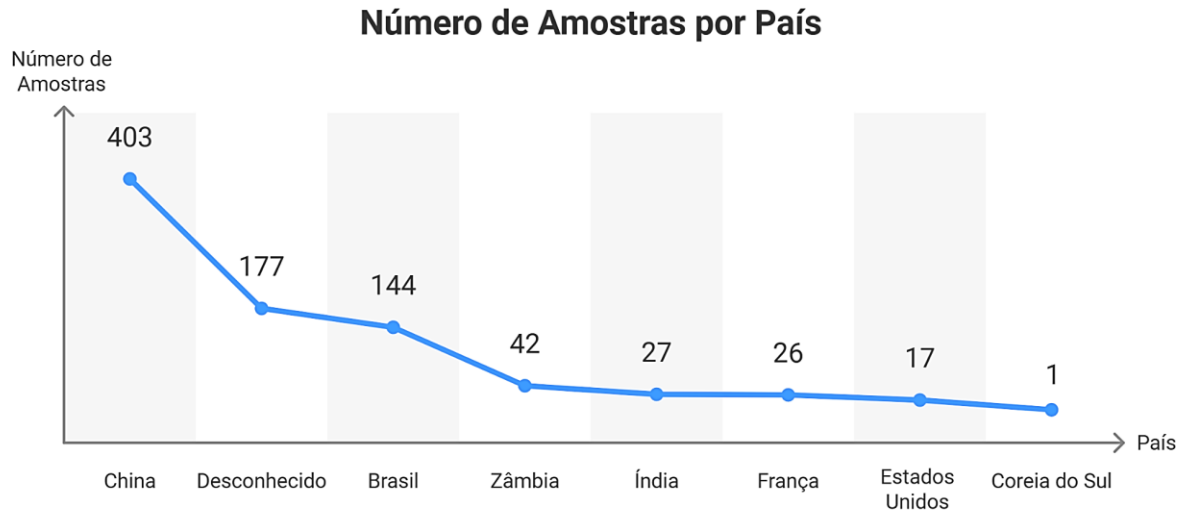
O gráfico da Figura 5 destaca a China como o país com maior quantidade de amostras (403), seguido por 'Desconhecido' (177) e Brasil (144), enquanto outros países apresentam números significativamente menores, como Zâmbia (42), Índia (27), França (26), Estados Unidos (17) e Coreia do Sul (1).

Tabela 1 - Distribuição das amostras conforme o tipo de tecido, a fase da vida e o sexo dos insetos coletados, incluindo o número inicial de amostras e o número final após as filtragens aplicadas aos conjuntos de dados

Tecido	Fase da vida	Sexo	Número de amostras (Inicial)	Número de amostras (Filtrado)	Observação
Corpo inteiro	Larva	-	195	64	
Corpo inteiro	Pupa	-	39	39	
Corpo inteiro	Adulto	Macho	14	14	
Corpo inteiro	Adulto	Femêa	14	14	Selecionadas
Instestino médio	Larva	-	58	55	para as
Ovário	Pupa	-	25	18	análises de
Corpo gorduroso	Larva	-	22	22	WGCNA
Hemolinfa	Larva	-	18	18	
Integumento	Larva	-	19	17	
Tórax	Adulto	-	15	15	
Desconhecido	Larva	-	136	-	
Corpo inteiro	Desconhecido	-	92	-	
Desconhecido	Desconhecido	-	45	-	Excluídas
Linhagem celular	-	-	25	-	por falta de
Instestino médio	Desconhecido	-	20	-	informação
Desconhecido	Adulto	-	12	-	
Ovário	Desconhecido	-	10	-	
-	Embrião	-	12	6	
Hemócito	Larva	-	9	-	
Gândula de feromônio	Desconhecido	-	9	-	
Cerébro	Adulto	-	8	-	
Glândulas salivares	Larva	-	8	-	
Intestino	Desconhecido	-	6	-	
Abdômen e glândula de feromônio	Desconhecido	-	6	-	
Antena	Adulto	-	3	-	Excluídas
Antena	Desconhecido	-	2	-	pelo número
Asas membranosas	Adulto	-	5	-	de amostras
Ovário	Adulto	Fêmea	3	-	
Testículo	Adulto	Macho	2	-	
Cutícula	Larva	-	1	-	
Cabeça	Larva	-	1	-	
Túbulo de Malpighi	Larva	-	1	-	
Corpo Inteiro	Larva, pupa, adulto	-	1	-	
Hemolinfa	Desconhecido		1	-	

Fonte: Elaborado pelo autor.

Figura 5 - Distribuição do número de amostras por país



Fonte: Elaborado pelo autor.

Em relação às diferentes fases de vida, a maior parte dos dados obtidos são referentes à larva (470), seguidos de adulto (54), pupa (65) e embrião (12), enquanto 236 conjuntos de dados não possuem essa informação no SRA. Há uma maior incidência de estudos com larva, o que pode ser atribuído ao fato de ser a fase da vida da lagarta-do-cartucho que causa maior dano às plantas cultivadas (BAKRY *et al.*, 2023; WANG *et al.*, *2020*). Em estágios iniciais, as larvas alimentam-se de folhas na parte superior da copa da planta, já as larvas mais velhas geralmente se alimentam em áreas protegidas da planta, como o verticilo, a base da folha ou o fruto (BUNTIN *et al.*, 1986). Desta maneira, os estudos com larvas podem ser mais interessantes para a identificação de genes alvo para o desenvolvimento de estratégias de controle.

Em relação aos tecidos, o maior número de conjunto de dados foi referente a amostras de corpo inteiro (341), seguido de intestino médio (78) e ovário (38). Estudos com o corpo inteiro podem auxiliar na montagem de um transcriptoma de referência para esses insetos e auxiliar em análises voltadas à comparação de grupos controles e tratados. Estudos com dados de RNA-seq de corpo inteiro já foram utilizados, por exemplo, para comparar o transcriptoma de larvas de *S. frugiperda* em um grupo controle com aquele de larvas de grupos tratados com três tipos de inseticidas diferentes, permitindo a identificação de genes diferencialmente expressos em todos os tratamentos e aqueles que foram específicos de cada um deles (SHU *et al.*, 2023). Além do corpo inteiro, é interessante observar a grande quantidade de dados de intestino médio, o que pode refletir a busca por alvos nessa região do corpo dos insetos que está

diretamente em contato com os alimentos ingeridos, incluindo as moléculas de defesa das plantas. O intestino médio, por exemplo, é o alvo das toxinas Cry isoladas de *Bacillus thuringiensis* (Bt) e inseridas nas plantas transgênicas contendo Bt (KARUPPAIYAN *et al.*, 2022; WANG *et al*., 2019; MAHALAKSHMI *et al*., 2021).

Ao extrair as informações dos metadados obtidos, observou-se que a China foi responsável pela grande maioria dos conjuntos de dados de sequenciamento (Figura 5), isto podendo ser devido a grande presença da praga no país, resultante da grande produção de milho (GLOBO RURAL, 2024), concomitantemente com uma maior disponibilidade de infraestrutura e qualificação para a realização de sequenciamentos. No país há, por exemplo, o The National Genomics Data Center (NGDC) fundamental para a gestão de dados genômicos na China, com infraestrutura avançada de 1,6 Gbps de largura de banda, 11.200 núcleos de processamento, 437 TFlops e 46 petabytes de armazenamento (NATIONAL GENOMICS DATA CENTER, s.d.).

A mariposa Spodoptera frugiperda dispersou-se pela primeira vez no sul da China em dezembro de 2018, marcando o início de sua invasão nessa região e, ao longo do tempo, espalhou-se pelo país, representando uma ameaça severa à agricultura (SUN *et al*., 2021; ZHOU *et al*., 2021). Em 2019 e 2020, ela se espalhou de 26 para 27 províncias, respectivamente, danificando por volta de 1 milhão de hectares de plantio por ano, sendo o milho a cultura mais significativamente impactada (ZHOU *et al*., 2021). Tendo em conta estas informações, fica claro a grande presença de dados de sequenciamento visando estudar o organismo para desenvolvimento de métodos de controle mais eficazes.

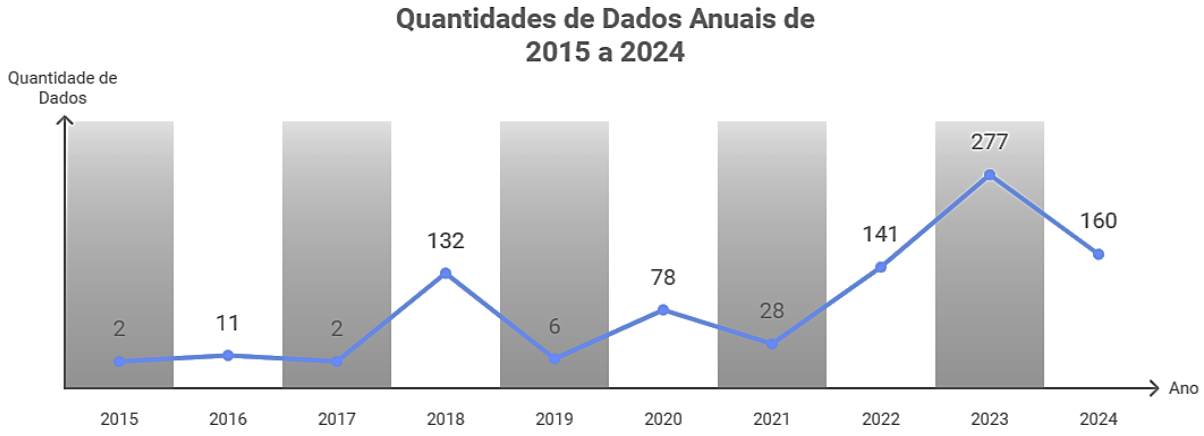
Além da China, observou-se que o Brasil foi o segundo país que mais contribuiu com dados de sequenciamento, produzindo um total de 144 conjuntos de dados. Essa alta contribuição do Brasil, também pode ter ocorrido visto que o país também é um grande produtor de milho (GLOBO RURAL, 2024) e de outras plantas cultivadas e que são atacadas por essa espécie. *S. frugiperda* tem sido reportada, como a principal praga do milho no Brasil em diferentes anos (CRUZ, 1995; VALICENTE, 2015). No entanto, é provável que haja menos

dados provenientes do Brasil do que da China, devido à falta de maiores investimentos em infraestrutura e capacitação de pessoal para a realização dessas análises.

Além desses dados, ressalta-se novamente a falta de uma completa descrição dos dados depositados no SRA, refletida no fato de que 177 das amostras disponíveis no banco de dados não apresentaram em seus metadados o local de proveniência do sequenciamento, levando a uma perda de informações e uma possível análise.

Outra observação interessante acerca dos dados de RNA-seq obtidos refere-se à data de liberação dos conjuntos de dados no banco de dados do SRA (Figura 6), a qual demonstra uma crescente quantidade de dados, principalmente a partir de 2018, o que pode ser devido ao início da incidência dessa praga na China, conforme descrito anteriormente.

Figura 6 - Evolução da Quantidade de Dados Anuais de 2015 a 2024



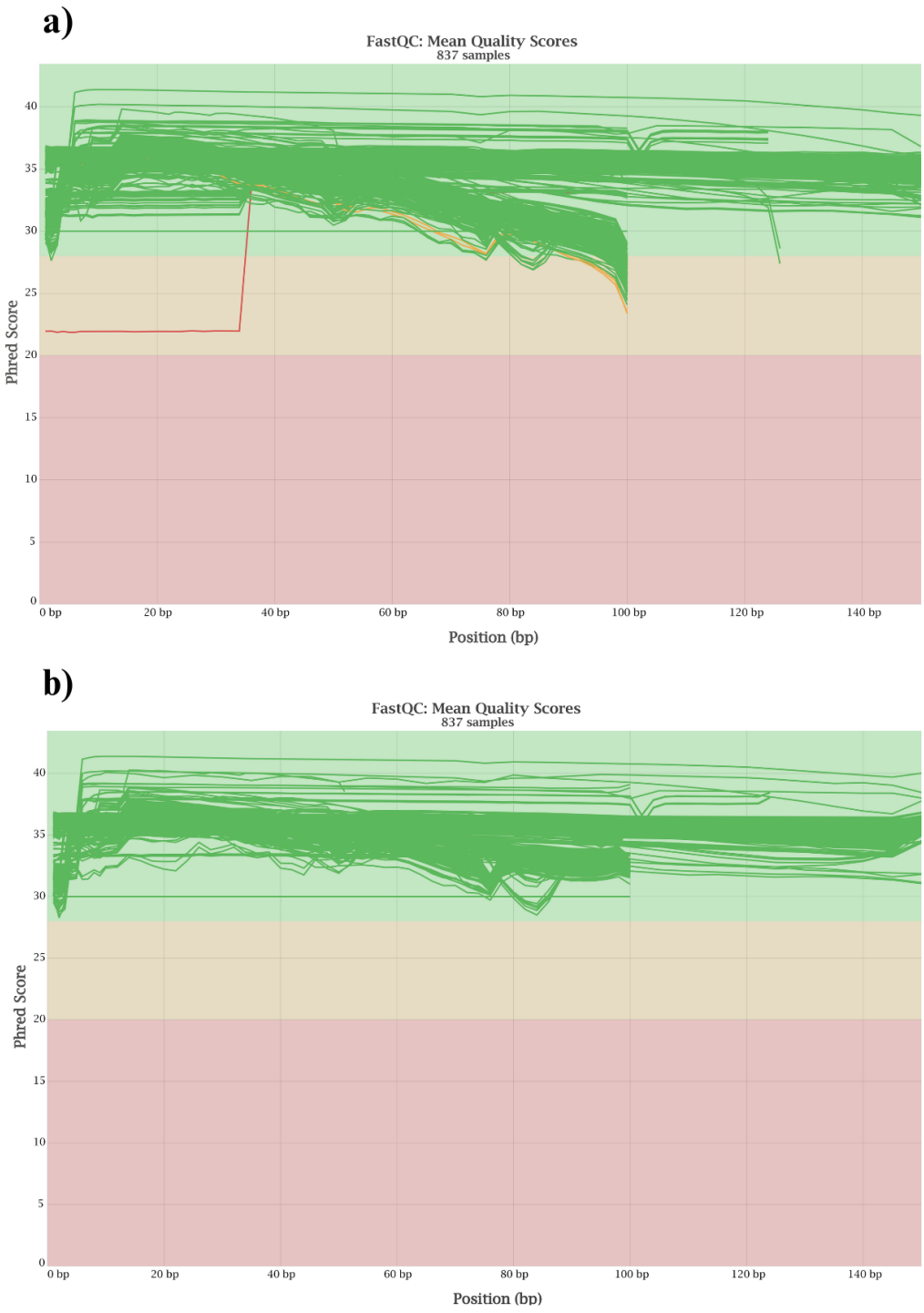
Fonte: Elaborado pelo autor.

4.3 Filtragem das sequências por qualidade de sequenciamento

Os dados de sequenciamento foram filtrados por qualidade pelo pipeline do nf-core, utilizando os softwares CutAdapt e Trimmomatic. A Figura 7 apresenta os gráficos de qualidade das sequências por posição antes e após a filtragem por estes programas, demonstrando a sua eficácia na limpeza desses dados. Além das posições com baixa qualidade de sequenciamento, também foram removidas sequências de adaptadores de diferentes tamanhos (Figura 8), resultando em uma significante redução do número de conjuntos de dados com presença de adaptadores. Com este filtro, os conjuntos que apresentavam adaptadores foram reduzidos de

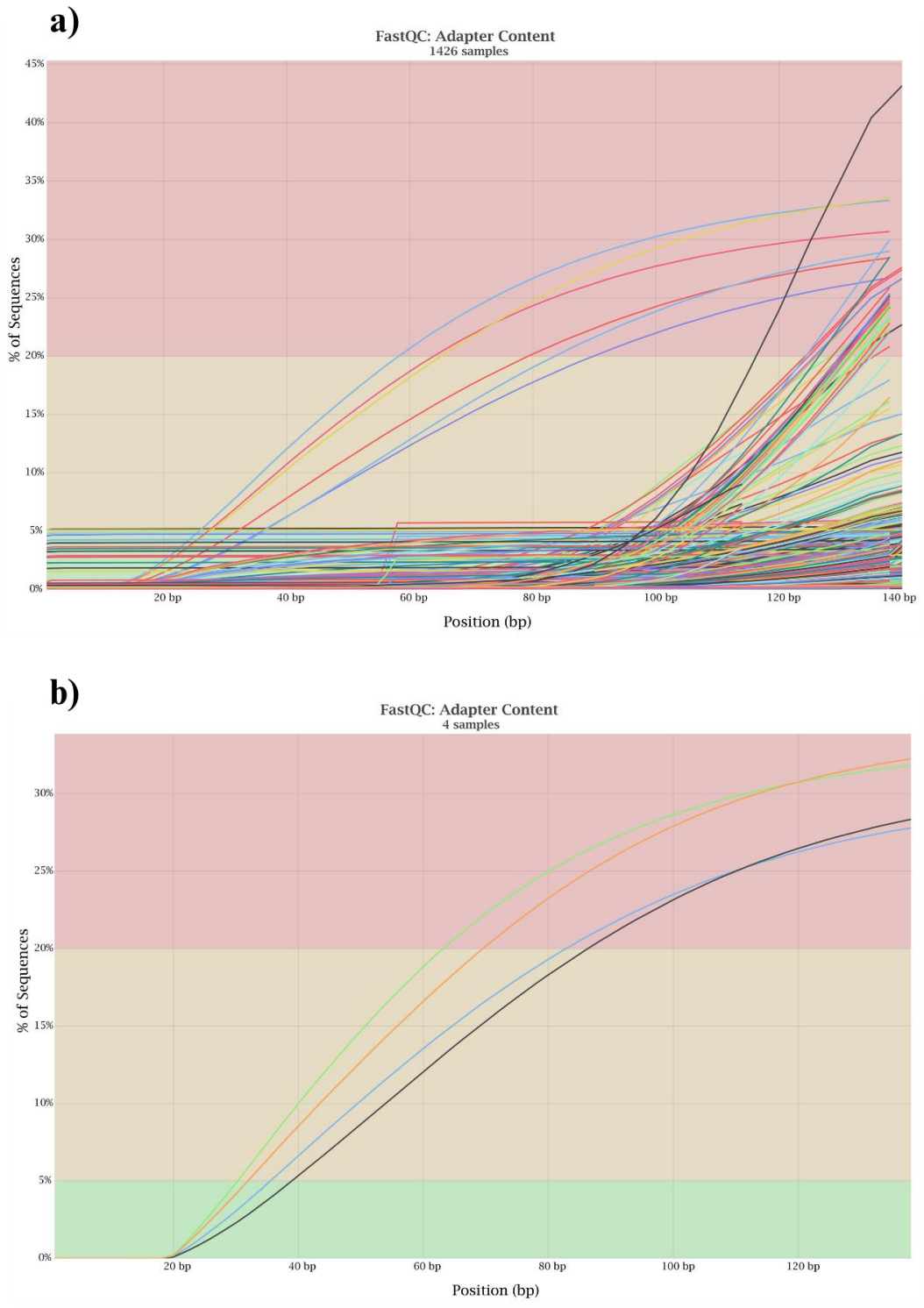
1426 para 4 (Figura 8). Desta forma, é possível observar que os conjuntos de dados utilizados nas análises posteriores foram satisfatoriamente filtrados por essas etapas.

Figura 7 - Avaliação dos escores de qualidade média das sequências ao longo das posições de bases em 837 amostras, agregados pelo MultiQC, utilizando dados do FastQC.



(a) Representa a qualidade inicial das sequências antes da trimagem, com uma queda acentuada em determinadas regiões. (b) Mostra a qualidade das sequências após a trimagem, evidenciando uma melhora geral nos escores e uma maior estabilidade ao longo das posições.
Fonte: Elaborado pelo autor.

Figura 8 - Conteúdo de adaptadores nas sequências ao longo das posições de bases, analisado pelo FastQC.



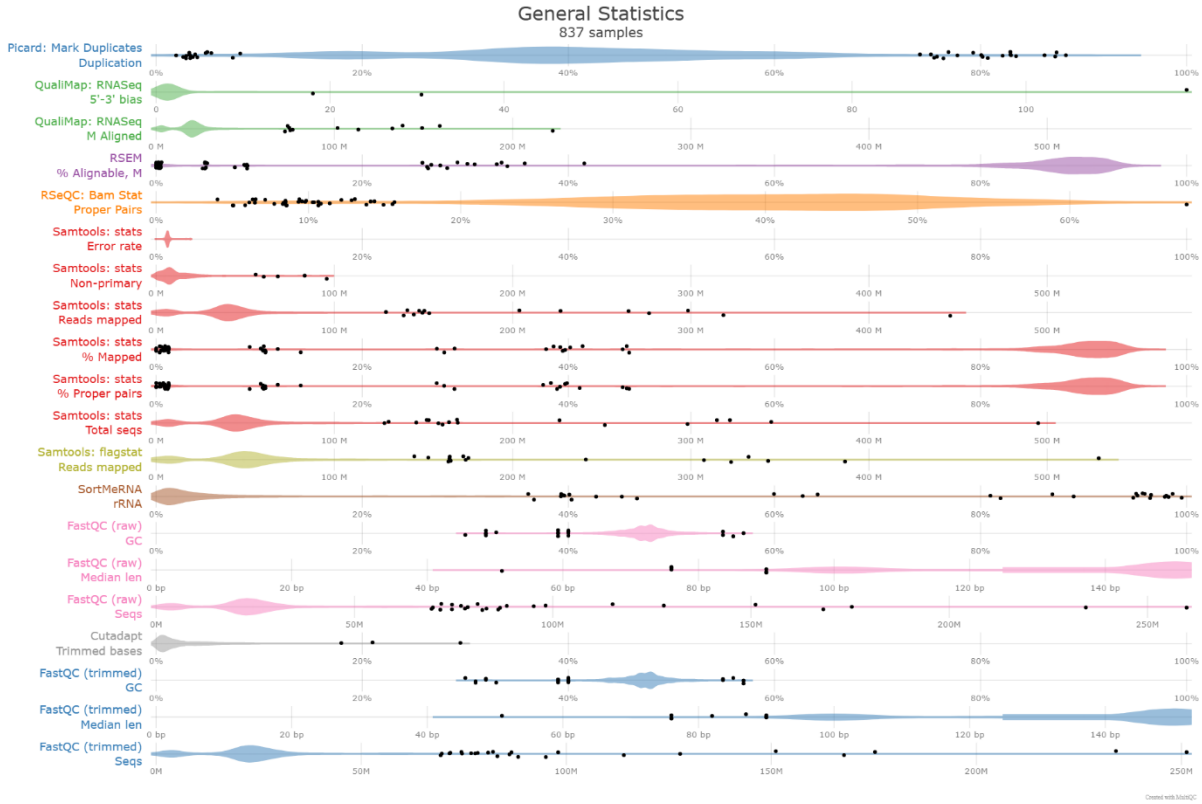
(a) Exibe a presença de adaptadores em 1.426 amostras antes da trimagem, com aumento significativo nas posições finais das sequências. (b) Mostra a redução do conteúdo de adaptadores após a trimagem em 4 amostras, evidenciando uma melhoria na remoção de adaptadores e maior qualidade das sequências.

Fonte: Elaborado pelo autor.

4.3 Filtragem dos dados para remoção de sequências contaminantes e profundidade de sequenciamento

Após as filtragens das sequências por qualidade e presença de adaptadores descritas anteriormente, muitas sequências ainda apresentavam valores de percentual de GC desviantes do padrão encontrado na maioria das amostras (Figura 9). Para evitar ruídos oriundos de conjuntos de dados contendo muitas sequências contaminantes, foram removidas as amostras com valores de percentual médio de GC inferiores ou superiores à média (47,2%), mais ou menos duas vezes o valor do desvio padrão (2,6) observado para todas as amostras obtidas do banco de dados do SRA. Este filtro removeu 32 conjuntos de dados que podem, dentre outras coisas, representar conjuntos de dados anotados de forma errônea no SRA como provenientes de *S. frugiperda*.

Figura 9 - Estatísticas gerais de qualidade e alinhamento das sequências em 837 amostras



A figura mostra métricas detalhadas sobre duplicação, alinhamento, taxa de erro, mapeamento e conteúdo de bases, obtidas por diversas ferramentas, incluindo Picard, QualiMap, RSeQC, Samtools e FastQC. Cada linha representa uma métrica específica, com distribuição de valores para cada amostra, evidenciando a variabilidade e a qualidade geral do conjunto de dados após o processamento.
Fonte: Elaborado pelo autor.

Além dos contaminantes, foram removidas sequências com tamanho total das leituras de sequenciamento (somadas as duas leituras pareadas) inferiores a 190 pb, com o intuito de diminuir o percentual de alinhamentos inespecíficos contra o genoma de referência. Esse filtro removeu 10 sequências apresentadas como *outliers* nos valores de "*FastQC (trimmed) - Median len*" na Figura 8, para as quais os tamanhos médios de leituras de sequenciamento de cada amostra foram inferiores a 95pb.

Ademais, com o intuito de selecionar conjuntos de dados com uma maior cobertura do genoma, aumentando as chances de genes pouco expressos serem contemplados pela amostragem, apenas conjuntos de dados que apresentaram um mínimo de 5 milhões de leituras de sequenciamento alinhadas contra o genoma foram mantidos. Nesse filtro, foram removidos 162 conjuntos de dados. Além do número de leituras de sequenciamento mapeadas, também foram removidas amostras para as quais menos de 70% das leituras de sequenciamento foram alinhadas ao genoma de referência. Nesta etapa, foram removidas as 62 amostras destacadas como *outliers* na Figura 8 (*"Samtools: stats - % mapped"*).

Desta forma, foram selecionados 641 conjuntos de dados que passaram em todos os filtros descritos anteriormente.

4.4 Seleção das amostras para as análises de WCGNA

Muitos conjuntos de dados obtidos do banco de dados do SRA não apresentavam uma descrição completa dos tecidos e fases da vida amostrados (Tabela 1). Diante disso, 339 conjuntos de dados precisaram ser removidos das análises de WGCNA por falta de informação sobre as amostras obtidas. Além desses, 25 conjuntos de dados de sequenciamento provenientes de análises com linhagens celulares não identificados também foram excluídos das análises. Além disso, dentre as possíveis condições, para os quais seria possível classificar as amostras por tecido e fase da vida, 12 foram excluídos por possuírem menos de 10 amostras cada, antes ou após a filtragem dos dados (Tabela 1), levando à exclusão de um total de 54 amostras nesta etapa. Desta maneira, 365 conjuntos foram eliminados tendo em conta a falta de informação e baixa quantidade de amostras presentes na condição.

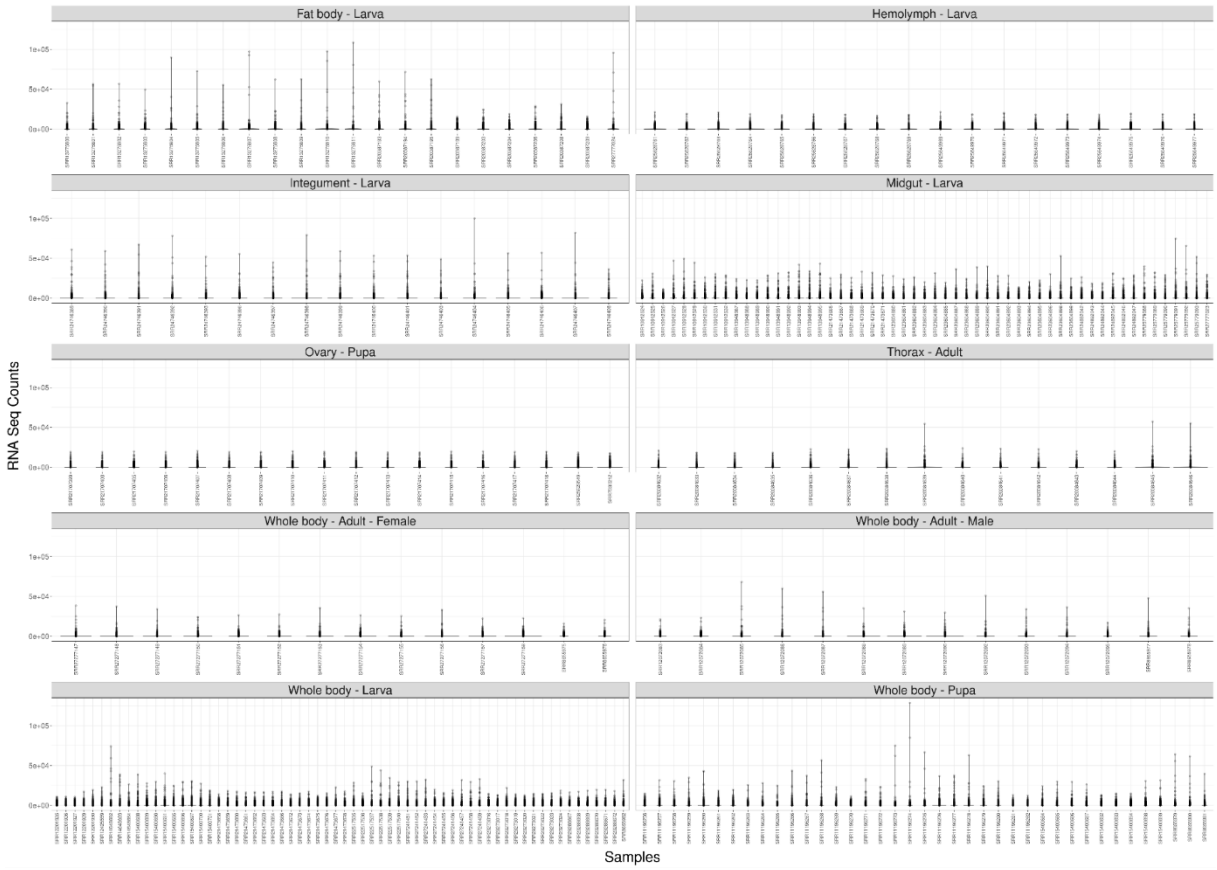
Após as filtragens, foram selecionadas 10 condições para as análises de WGCNA, incluindo amostras de corpo inteiro de larvas (64), pupa (39), machos adultos (14) e fêmeas

adultas (14); intestino médio (55), corpo gorduroso (22), hemolinfa (18) e integumento (17) de larvas; ovário de pupas (18); e tórax de adultos (15) (Tabela 1), totalizando 276.

4.5 Rede de co-expressão gênica

As amostras selecionadas na etapa anterior foram submetidas às análises de predição de redes de co-expressão gênica utilizando o método WGCNA. Uma análise exploratória do conjunto de dados nos permitiu analisar a presença de *outliers* entre as amostras selecionadas para cada condição. Na Figura 10 são apresentados gráficos comparando a contagem de leituras de RNA-seq por amostra para cada uma das 10 condições selecionadas, demonstrando uma uniformidade nos dados obtidos para cada um deles, sendo que cada painel representa um tecido específico em diferentes estágios, como larva, pupa e adulto (feminino e masculino). O eixo vertical exibe a contagem de RNA Seq em uma escala logarítmica, enquanto o eixo horizontal mostra as diferentes amostras analisadas.

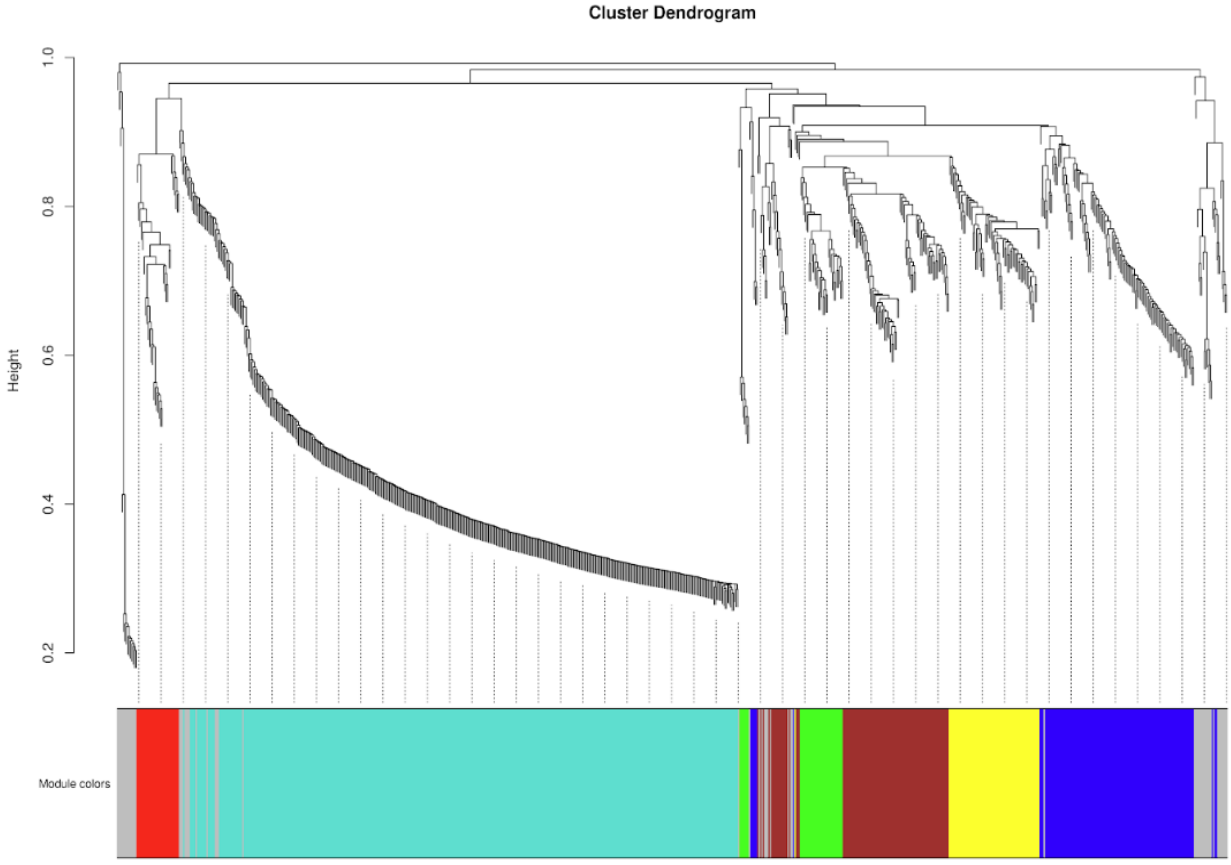
Figura 10 - Contagem de leituras de RNA Seq para diferentes amostras e tecidos em vários estágios de desenvolvimento.



Fonte: Elaborado pelo autor.

Um total de sete módulos de co-expressão foram preditos pelo WGCNA (Figura 11): turquesa (*turquoise)* (393 genes), azul (*blue)* (120 genes), marrom (*brown)* (97 genes), amarelo (*yellow)* (67 genes), cinza (*grey)* (60 genes), verde (green) (38 genes) e vermelho (*red)* (31 genes). Cada ramo representa a similaridade entre genes com base na expressão. A altura dos nós indica o nível de similaridade: nós mais baixos representam genes mais similares. Na parte inferior, as cores dos módulos indicam agrupamentos de genes com padrões de co-expressão, usados para identificar grupos de genes co-regulados e funcionalmente relacionados.

Figura 11 - Dendrograma de agrupamento (*clustering)*

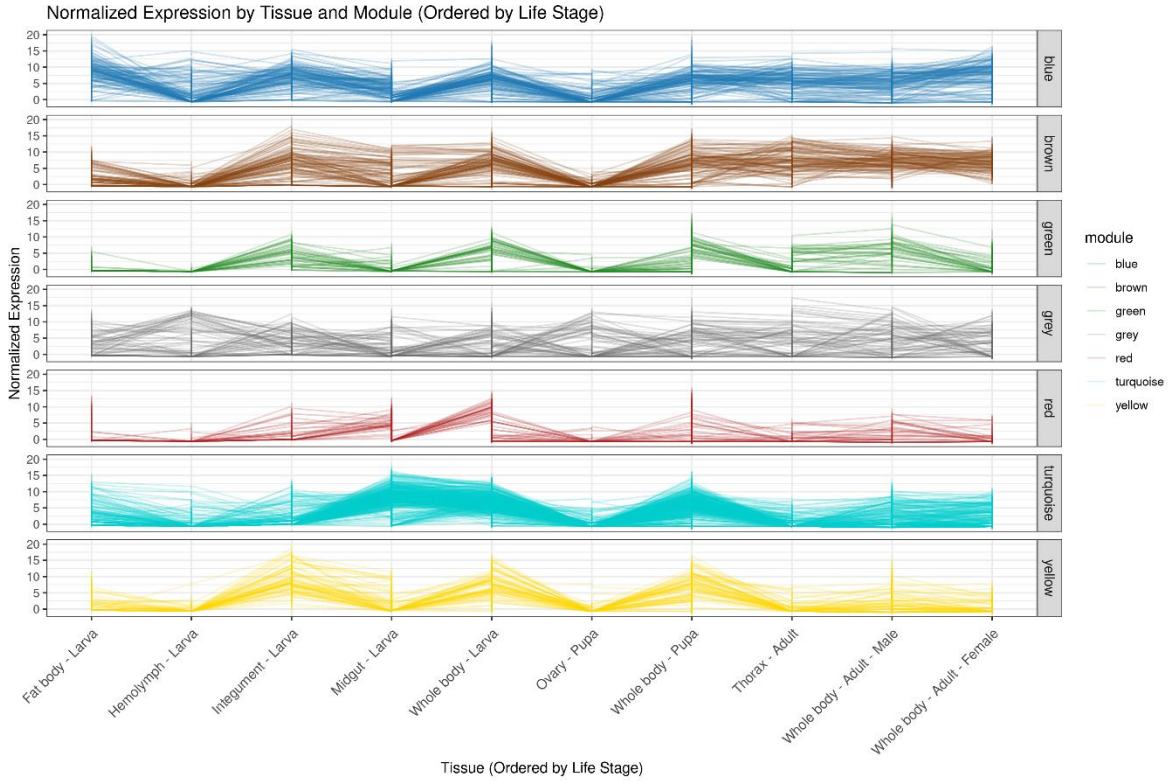


Fonte: Elaborado pelo autor.

Comparações entre os valores de expressão normalizados e a sua média por tecido dentro de cada módulo são apresentadas nas Figuras 12 e 13, respectivamente. Esses dados demonstram, por exemplo, que o maior módulo (turquesa) é composto por genes mais expressos no intestino médio e no corpo inteiro das larvas, demonstrando que esse módulo de genes pode ser o mais interessante para prospecção de genes alvos nessa fase de desenvolvimento.

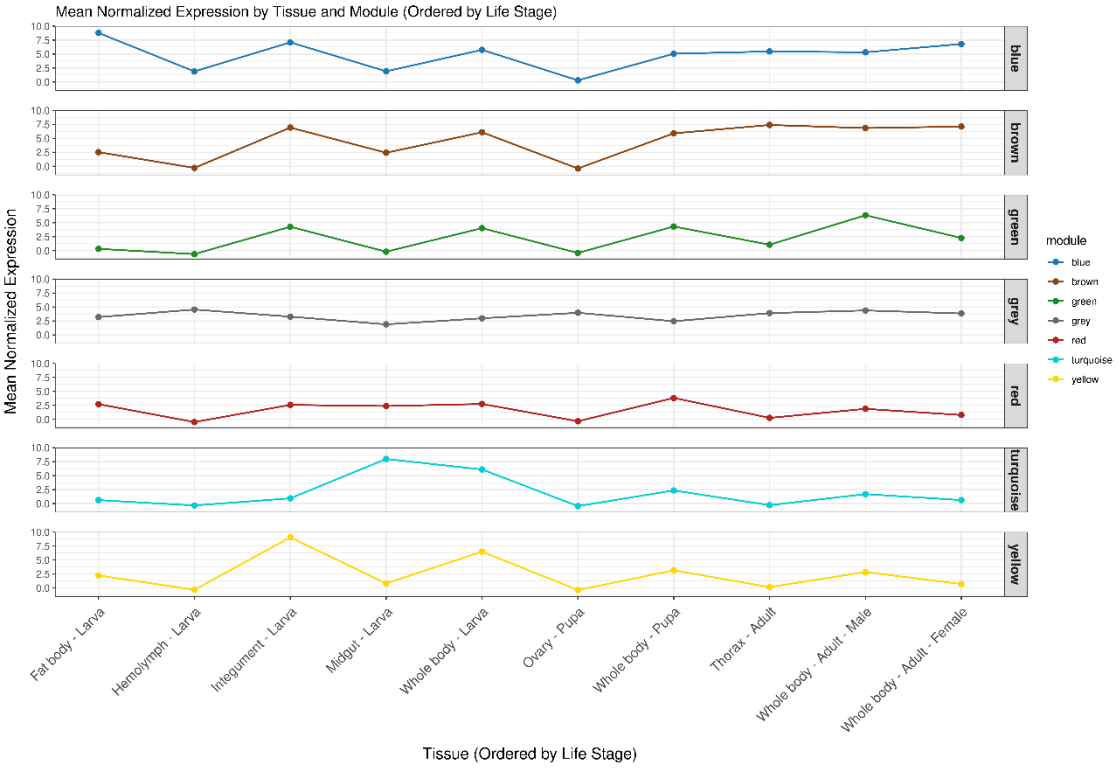
Na figura 13, cada linha representa a média de expressão de um módulo de co-expressão específico, indicado pela cor (azul, marrom, verde, cinza, vermelho, turquesa e amarelo). Os pontos em cada linha destacam os valores médios de expressão para facilitar a comparação entre módulos e tecidos.

Figura 12 - Expressão normalizada de genes em diferentes tecidos e fases da vida



Fonte: Elaborado pelo autor.

Figura 13 - Expressão média normalizada de genes em diferentes tecidos e tecidos.



Fonte: Elaborado pelo autor.

5 CONCLUSÃO

Através deste estudo, foi observada uma presença significativa de conjuntos de dados de sequenciamento de S. frugiperda provenientes da China, seguida pelo Brasil, o que reflete o impacto desta praga nesses países e a disponibilidade de infraestrutura para pesquisas ômicas.

A fase larval do organismo foi a mais representativa nos dados coletados, sendo justificável devido a que este estágio de desenvolvimento causa os maiores danos às plantas cultivadas. Concomitantemente, a grande presença do tecido de intestino médio e corpo inteiro para essa fase destaca a busca por alvos moleculares que possam ser pesquisados para o desenvolvimento de estratégias de controle mais eficazes, especialmente a interação do intestino médio com compostos, toxinas e defensivos agrícolas.

Os dados obtidos ao longo do desenvolvimento deste trabalho demonstraram que existe uma crescente quantidade de dados sendo disponibilizados no banco de dados do SRA do NCBI, demonstrando que análises comparando esses dados são cada dia mais possíveis de serem realizadas. No entanto, ainda pode ser observada uma limitação no uso e na realização de análises aprofundadas desses dados, devido à falta de metadados. Essa lacuna resulta da ausência de informações básicas sobre a procedência dos dados, tais como local de origem, tecido e fase da vida amostrados, no momento de submissão ao NCBI por parte dos pesquisadores.

Entretanto, mesmo com um número considerável de dados sendo descartados por falta de metadados, conseguimos, por meio deste trabalho, predizer redes de co-expressão pelo método WGCNA para 10 diferentes condições e 7 módulos de co-expressão gênicas, agrupados por fase de vida e tecido amostrado para *S. frugiperda*, demonstrando o potencial de uso desses dados.

Especialmente, ressalta-se a possibilidade de uma investigação aprofundada do módulo turquesa por apresentar genes com maior expressão no intestino médio da larva. A partir de análises mais detalhadas, poderia-se considerar a desativação de genes ou a aplicação da técnica de CRISPR (*Clustered Regularly Interspaced Short Palindromic Repeats*), com a finalidade de desenvolver novos métodos de controle da praga.

Por fim, esse trabalho gerou um pipeline de Bioinformática que pode ser aplicado em futuras análises que visem identificar redes de genes co-expressos, utilizando dados brutos de sequenciamento disponíveis no SRA. Bem como gerou dados que podem ser futuramente analisados para a identificação de genes de interesse em *S. frugiperda*.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS, S. FastQC: A Quality Control tool for High Throughput Sequence Data. *Babraham Bioinformatics*. Disponível em: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Acesso em: 14 nov. 2024.

ASHOK, K.; KENNEDY, J. S.; GEETHALAKSHMI, V.; JEYAKUMAR, P.; SATHIAH, N.; BALASUBRAMANI, V. Lifetable study of fall armyworm *Spodoptera frugiperda* (J. E. Smith) on maize. *Indian Journal of Entomology*, v. 82, n. 3, p. 574-579, 2020. DOI: 10.5958/0974-8172.2020.00143.1.

BAKRY, M. M. S.; ABDEL-BAKY, N. F. Population density of the fall armyworm, *Spodoptera frugiperda* (Smith) (Lepidoptera: Noctuidae) and its response to some ecological phenomena in maize crop. *Brazilian Journal of Biology*, v. 83, e271354, 2023. DOI: 10.1590/1519-6984.271354.

BOLZAN, A.; PADOVEZ, F. E.; NASCIMENTO, A. R.; KAISER, I. S.; LIRA, E. C.; AMARAL, F. S.; KANNO, R. H.; MALAQUIAS, J. B.; OMOTO, C. Selection and characterization of the inheritance of resistance of *Spodoptera frugiperda* (Lepidoptera: Noctuidae) to chlorantraniliprole and cross-resistance to other diamide insecticides. *Pest Management Science*, v. 75, p. 2682-2689, 2019. DOI: 10.1002/ps.5376.

BUNTIN, G. D. A review of plant response to fall armyworm, *Spodoptera frugiperda* (J. E. Smith), injury in selected field and forage crops. *Florida Entomologist*, v. 69, n. 3, p. 549-559, 1986.

BROARD INSTITUTE**.** Picard Toolkit. Broad Institute, GitHub Repository, 2019. Disponível em: https://broadinstitute.github.io/picard/. Acesso em: 10 nov. 2024.

CARVALHO, R. A.; OMOTO, C.; FIELD, L. M.; WILLIAMSON, M. S.; BASS, C. Investigating the molecular mechanisms of organophosphate and pyrethroid resistance in the fall armyworm *Spodoptera frugiperda*. *PLOS ONE*, v. 8, n. 4, e62268, 2013. DOI: 10.1371/journal.pone.0062268.

CRUZ, I. A lagarta-do-cartucho na cultura do milho. Sete Lagoas: EMBRAPA-CNPMS, 1995. 45 p. (*EMBRAPA-CNPMS. Circular técnica, 21*).

CRESPO, A. M.; GONÇALVES, D. da C.; SOUZA, M. N.; ZANÚNCIO JUNIOR, J. S.; COSTA, H.; FAVARATO, L. F.; RANGEL, O. J. P.; ARAÚJO, J. B. S. Manejo da lagarta-do-cartucho do milho (*Spodoptera frugiperda*): panorama geral das atualizações no controle alternativo. *Boletim Técnico*, n. 6. Alegre: Instituto Federal do Espírito Santo, 2021. Disponível em: http://biblioteca.incaper.es.gov.br/digital/handle/123456789/4192.

DAI, Y.; CHEN, W.; HUANG, J.; XIE, L.; LIN, J.; CHEN, Q.; JIANG, G.; HUANG, C. Identification of key pathways and genes in nasopharyngeal carcinoma based on WGCNA. *Auris Nasus Larynx*, v. 50, n. 1, p. 126-133, 2023. DOI: 10.1016/j.anl.2022.05.013.

DIEZ-RODRÍGUEZ, G. I.; OMOTO, C. Herança da resistência de *Spodoptera frugiperda* (J.E. Smith) (Lepidoptera: Noctuidae) a lambda-cialotrina. *Neotropical Entomology*, v. 30, n. 2, p. 311–316, 2001. DOI: 10.1590/S1519-566X2001000200016.

DOBIN, A.; DAVIS, C. A.; SCHLESINGER, F.; DRENKOW, J.; ZALESKI, C.; JHA, S.; BATUT, P.; CHAISSON, M.; GINGERAS, T. R. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, v. 29, n. 1, p. 15-21, 2013. DOI: 10.1093/bioinformatics/bts635.

ESPEK, T.; TAMMARU, T.; NYLIN, S. Intraspecific variability in number of larval instars in insects. *Journal of Economic Entomology*, v. 100, n. 3, p. 627-645, 2007. DOI: 10.1603/0022-0493(2007)100[627]2.0.CO;2.

EWELS, P.; MAGNUSSON, M.; LUNDIN, S.; KÄLLER, M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, v. 32, n. 19, p. 3047-3048, 2016. DOI: 10.1093/bioinformatics/btw354.

EWELS, P. A.; PELTZER, A.; FILLINGER, S. *et al*. The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, v. 38, p. 276–278, 2020. DOI: 10.1038/s41587-020-0439-x.

FULLER, T.; LANGFELDER, P.; PRESSON, A.; HORVATH, S. Review of weighted gene coexpression network analysis. In: *Springer Handbooks Comp. Stat.*, 2011. p. 369-388.

GARLET, C. G.; GUBIANI, P. S.; PALHARINI, R. B.; MOREIRA, R. P.; GODOY, D. N.; FARIAS, J. R.; BERNARDI, O. Field-evolved resistance to chlorpyrifos by *Spodoptera frugiperda* (Lepidoptera: Noctuidae): Inheritance mode, cross-resistance patterns, and synergism. *Pest Management Science*, v. 77, p. 5367-5374, 2021. DOI: 10.1002/ps.6576.

GLOBO RURAL. Quais são os maiores produtores de milho do mundo? *Globo Rural*, 04 jan. 2024. Disponível em: https://globorural.globo.com/agricultura/noticia/2024/01/quais-sao-os-maiores-produtores-de-milho-do-mundo.ghtml. Acesso em: 13 nov. 2024.

GOH, W. W. B.; WANG, W.; WONG, L. Why Batch Effects Matter in Omics Data, and How to Avoid Them. *Trends in Biotechnology*, v. 35, n. 6, p. 498-507, 2017. DOI: https://doi.org/10.1016/j.tibtech.2017.02.012.

JARA, J. S. *et al*. Control biológico de *Spodoptera frugiperda* en cultivo de *Zea mays*: uso de nematodos entomopatógenos. *Scientia Agropecuaria*, v. 4, n. 10, p. 551-557, 2019.

JIANG, F.; ZHOU, H.; SHEN, H. Identification of critical biomarkers and immune infiltration in rheumatoid arthritis based on WGCNA and LASSO algorithm. *Frontiers in Immunology*, v. 13, 2022. DOI: 10.3389/fimmu.2022.925695.

KANNO, R. H.; NASCIMENTO, A. R. B.; MONTEIRO, C. P.; AMARAL, F. S. A.; SINGH, K. S.; TROCZKA, B. J.; BASS, C.; CÔNSOLI, F. L.; OMOTO, C. Bulk segregant mapping and transcriptome analyses reveal the molecular mechanisms of spinetoram resistance in *Spodoptera frugiperda*. *Pesticide Biochemistry and Physiology*, v. 202, 2024. DOI: 10.1016/j.pestbp.2024.105921.

KARUPPAIYAN, T.; BALASUBRAMANI, V.; MURUGAN, M.; RAVEENDRAN, M.; RAJADURAI, G.; KOKILADEVI, E. Characterization and evaluation of indigenous *Bacillus thuringiensis* isolate T352 against fall armyworm, *Spodoptera frugiperda* (J.E. Smith). *International Journal of Plant & Soil Science*, v. 34, n. 21, p. 729-736, 2022. DOI: 10.9734/IJPSS/2022/v34i2131325.

KOPYLOVA, E.; NOÉ, L.; TOUZET, H. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. *Bioinformatics*, v. 28, n. 24, p. 3211-3217, 2012. DOI: 10.1093/bioinformatics/bts611.

KOVAKA, S.; ZIMIN, A. V.; PERTEA, G. M.; RAZAGHI, R.; SALZBERG, S. L.; PERTEA, M. Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology*, v. 20, n. 1, p. 278, 2019. DOI: 10.1186/s13059-019-1910-1.

KRUEGER, Felix; JAMES, Frankie; EWELS, Phil; AFYOUNIAN, Ebrahim; SCHUSTER-BOECKLER, Benjamin. FelixKrueger/TrimGalore: v0.6.7 - DOI via Zenodo (0.6.7). *Zenodo*, 2021. DOI: 10.5281/zenodo.5127899.

LANGFELDER, P.; HORVATH, S. WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*, v. 9, p. 559, 2008. DOI: 10.1186/1471-2105-9-559.

LI, B.; DEWEY, C. N. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, v. 12, p. 323, 2011. DOI: 10.1186/1471-2105-12-323.

LI, H. *et al*. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, v. 25, n. 16, p. 2078-2079, 2009. DOI: 10.1093/bioinformatics/btp352.

LIRA, E. C.; BOLZAN, A.; NASCIMENTO, A. R.; AMARAL, F. S.; KANNO, R. H.; KAISER, I. S.; OMOTO, C. Resistance of *Spodoptera frugiperda* (Lepidoptera: Noctuidae) to spinetoram: inheritance and cross-resistance to spinosad. *Pest Management Science*, v. 76, p. 2674-2680, 2020. DOI: 10.1002/ps.5812.

LOVE, M. I.; HUBER, W.; ANDERS, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, v. 15, n. 12, p. 550, 2014. DOI: 10.1186/s13059-014-0550-8.

MARCHE, C.; LIMASSET, A. Scalable sequence database search using partitioned aggregated Bloom comb trees. *Bioinformatics*, v. 39, suplemento 1, p. i252-i259, jun. 2023.

MARTIN, J. A.; JOHNSON, N. V.; GROSS, S. M.; SCHNABLE, J.; MENG, X.; WANG, M.; COLEMAN-DERR, D.; LINDSQUIST, E.; WEI, C.; KAEPPLER, S.; CHEN, F.; WANG, Z. A near complete snapshot of the *Zea mays* seedling transcriptome revealed from ultra-deep sequencing. *Scientific Reports*, v. 4, p. 4519, 2014. DOI: 10.1038/srep04519.

MAHALAKSHMI, M.; MAHEESHA, M.; VENKATASAMY, B.; MURUGAN, M.; MUTHURAJAN, R.; GOTHANDARAMAN, R.; TAMILNAYAGAN, T.; KOKILADEVI,

E.; SATHIAH, N. Characterisation of native *Bacillus thuringiensis* isolates toxicity to fall armyworm, *Spodoptera frugiperda* (J.E. Smith). *Journal of Biological Control*, v. 35, p. 171-180, 2021. DOI: 10.18311/jbc/2021/28812.

MURARO, D. S.; OLIVEIRA ABBADE NETO, D.; KANNO, R. H.; KAISER, I. S.; BERNARDI, O.; OMOTO, C. Inheritance patterns, cross-resistance and synergism in *Spodoptera frugiperda* (Lepidoptera: Noctuidae) resistant to emamectin benzoate. *Pest Management Science*, v. 77, p. 5049-5057, 2021. DOI: 10.1002/ps.6545.

NASCIMENTO, A. R. B.; FARIAS, J. R.; BERNARDI, D.; HORIKOSHI, R. J.; OMOTO, C. Genetic basis of *Spodoptera frugiperda* (Lepidoptera: Noctuidae) resistance to the chitin synthesis inhibitor lufenuron. *Pest Management Science*, v. 72, n. 4, p. 810–815, 2016. DOI: 10.1002/ps.4057. Acesso em: 15 nov. 2024.

NASCIMENTO, J.; GONÇALVES, K. C.; DIAS, N. P.; OLIVEIRA, J. L.; BRAVO, A.; POLANCZYK, R. A. Adoption of *Bacillus thuringiensis*-based biopesticides in agricultural systems and new approaches to improve their use in Brazil. *Biological Control*, v. 165, 2022. DOI: 10.1016/j.biocontrol.2021.104792.

NATIONAL GENOMICS DATA CENTER. National Genomics Data Center, China National Center for Bioinformation / Beijing Institute of Genomics, Chinese Academy of Sciences. Disponível em: https://ngdc.cncb.ac.cn/. Acesso em: 13 nov. 2024.

NIZ, J. M.; SALVADOR, R.; FERRELLI, M. L.; SCIOCCO DE CAP, A.; ROMANOWSKI, V.; BERRETTA, M. F. Genetic variants in Argentinean isolates of *Spodoptera frugiperda* Multiple Nucleopolyhedrovirus. *Virus Genes*, 2020. DOI: 10.1007/s11262-020-01741-9.

OKONECHNIKOV, K.; CONESA, A.; GARCÍA-ALCALDE, F. Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data. *Bioinformatics*, v. 32, n. 2, p. 292-294, 2016. DOI: 10.1093/bioinformatics/btv566.

POPLOWSKI, A.; BINDER, H. Feasibility of sample size calculation for RNA-seq studies. *Briefings in Bioinformatics*, v. 19, n. 4, p. 713-720, 2018. DOI: 10.1093/bib/bbw144.

QUINLAN, A. R.; HALL, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, v. 26, n. 6, p. 841-842, 2010. DOI: 10.1093/bioinformatics/btq033.

RAPAPORT, F.; KHANIN, R.; LIANG, Y.; PIRUN, M.; KREK, A.; ZUMBO, P.; MASON, C. E.; SOCCI, N. D.; BETEL, D. Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biology*, v. 14, n. 9, p. R95, 2013. DOI: 10.1186/gb-2013-14-9-r95.

SAYOLS, S.; SCHERZINGER, D.; KLEIN, H. dupRadar: a Bioconductor package for the assessment of PCR artifacts in RNA-Seq data. *BMC Bioinformatics*, v. 17, n. 1, p. 428, 2016. DOI: 10.1186/s12859-016-1276-2.

SCHURCH, N. J.; SCHOFIELD, P.; GIERLIŃSKI, M.; COLE, C.; SHERSTNEV, A.; SINGH, V.; WROBEL, N.; GHARBI, K.; SIMPSON, G. G.; OWEN-HUGHES, T.;

BLAXTER, M.; BARTON, G. J. How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? *RNA*, v. 22, n. 6, p. 839-851, 2016. DOI: 10.1261/rna.053959.115.

SHI, Y.; HE, L.; DING, W.; HUANG, H.; HE, H.; XUE, J.; GAO, Q.; ZHANG, Z.; LI, Y.; QIU, L. Function analysis of *CYP321A9* from *Spodoptera frugiperda* (Lepidoptera: Noctuidae) associated with emamectin benzoate, and a novel insecticide, cyproflanilide detoxification. *Journal of Economic Entomology*, v. 116, n. 5, p. 1812-1819, 2023. DOI: 10.1093/jee/toad168.

SHU, H.; LIN, Y.; ZHANG, Z.; QIU, L.; DING, W.; GAO, Q.; XUE, J.; LI, Y.; HE, H. The transcriptomic profile of *Spodoptera frugiperda* differs in response to a novel insecticide, cyproflanilide, compared to chlorantraniliprole and avermectin. *BMC Genomics*, v. 24, n. 1, p. 3, 2023. DOI: 10.1186/s12864-022-09095-2.

STRÖHER, P. Sequenciamento de nova geração e entomologia: Novas perspectivas para antigos questionamentos. *Revista da Biologia*, v. 18, p. 27-36, 2018. DOI: https://doi.org/10.7594/revbio.18.01.04.

SUN, X. X.; HU, C. X.; JIA, H. R.; WU, Q. L.; SHEN, X. J.; ZHAO, S. Y.; JIANG, Y. Y.; WU, K. M. Case study on the first immigration of fall armyworm, *Spodoptera frugiperda* invading into China. *Journal of Integrative Agriculture*, v. 20, n. 3, p. 664-672, 2021. DOI: 10.1016/S2095-3119(19)62839-X.

VALICENTE, F. H. Manejo integrado de pragas na cultura do milho. Sete Lagoas: Embrapa Milho e Sorgo, 2015. Circular Técnica *Embrapa Milho e Sorgo*, n. 208. DOI: 10.13140/RG.2.1.3438.8885.

WANG, L.; WANG, S.; LI, W. RSeQC: quality control of RNA-seq experiments. *Bioinformatics*, v. 28, n. 16, p. 2184-2185, 2012. DOI: 10.1093/bioinformatics/bts356.

WANG, W.; HE, P.; ZHANG, Y.; LIU, T.; JING, X.; ZHANG, S. The population growth of *Spodoptera frugiperda* on six cash crop species and implications for its occurrence and damage potential in China. *Insects*, v. 11, n. 9, 639, 2020. DOI: 10.3390/insects11090639.

WANG, Y.; WANG, J.; FU, X.; NAGEOTTE, J. R.; SILVERMAN, J.; BRETSNYDER, E. C.; CHEN, D.; RYDEL, T. J.; BEAN, G. J.; LI, K. S.; KRAFT, E.; GOWDA, A.; NANCE, A.; MOORE, R. G.; PLEAU, M. J.; MILLIGAN, J. S.; ANDERSON, H. M.; ASIIMWE, P.; EVANS, A.; MOAR, W. J.; MARTINELLI, S.; HEAD, G. P.; HAAS, J. A.; BAUM, J. A.; YANG, F.; KERNS, D. L.; JERGA, A. *Bacillus thuringiensis* Cry1Da_7 and Cry1B.868 protein interactions with novel receptors allow control of resistant fall armyworms, *Spodoptera frugiperda* (J.E. Smith). *Applied and Environmental Microbiology*, v. 85, e00579-19, 2019. DOI: 10.1128/AEM.00579-19.

WANG, Z.; GERSTEIN, M.; SNYDER, M. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, v. 10, n. 1, p. 57-63, 2008. DOI: 10.1038/nrg2484.

YANG, Y.; LI, G.; QIAN, H.; WILHELMSEN, K. C.; SHEN, Y.; LI, Y. SMNN: batch effect correction for single-cell RNA-seq data via supervised mutual nearest neighbor detection. *Briefings in Bioinformatics*, v. 22, n. 3, 2021. DOI: https://doi.org/10.1093/bib/bbaa097.

XU, J.; SU, Z.; HONG, H.; THIERRY-MIEG, J.; THIERRY-MIEG, D.; KREIL, D. P.; MASON, C. E.; TONG, W.; SHI, L. Cross-platform ultradeep transcriptomic profiling of human reference RNA samples by RNA-Seq. *Scientific Data*, v. 1, p. 20, 2014. DOI: 10.1038/sdata.2014.20.

ZHAO, Q.; ZHANG, Y.; SHAO, S.; SUN, Y.; LIN, Z. Identification of hub genes and biological pathways in hepatocellular carcinoma by integrated bioinformatics analysis. *PeerJ*, v. 9, e10594, 2021. DOI: 10.7717/peerj.10594.

ZHAO, X.; YU, H.; KONG, L.; LI, Q. Gene co-expression network analysis reveals the correlation patterns among genes in Euryhaline adaptation of *Crassostrea gigas*. *Marine Biotechnology*, v. 18, p. 535-544, 2016.

ZHOU, Y.; WU, Q. L.; ZHANG, H. W.; WU, K. M. Spread of invasive migratory pest *Spodoptera frugiperda* and management practices throughout China. *Journal of Integrative Agriculture*, v. 20, n. 3, p. 637-645, 2021. DOI: 10.1016/S2095-3119(21)63621-3.

APÊNDICE A – LISTA DE ACESSO

A lista de acesso contendo os identificadores dos conjuntos de dados utilizados neste trabalho está disponível para consulta no repositório público do GitHub, no seguinte endereço: https://github.com/SpatiumRimor/CoExGenePipeline/blob/8943bb53c591e2151fb4b0525852c89e052a98a6/SraAccList.txt

APÊNDICE B - SCRIPT PARA DOWNLOAD DE DADOS DO SRA E CONVERSÃO PARA FORMATO FASTQ

```
#!/bin/bash

# Script: NCBI_fastq.sh
# Descrição: Este script realiza o download de arquivos SRA e os converte para o formato FASTQ, verificando se os arquivos já foram baixados e atualizando as listas correspondentes.
# Uso: ./NCBI_fastq.sh


# Verifica se os arquivos 'baixados.txt' e 'SraAccList_novos.txt' existem
if [[ ! -f baixados.txt || ! -f SraAccList_novos.txt ]]; then
    echo "Criando arquivos iniciais 'baixados.txt' e 'SraAccList_novos.txt'."

    # Cria o arquivo 'baixados.txt' com a lista de diretórios existentes
    for dir in */ ; do
        echo "${dir%/}"
    done | sort | uniq > baixados.txt

    # Cria o arquivo 'SraAccList_novos.txt' com os novos itens que ainda não foram baixados
    grep -vxFf baixados.txt SraAccList.txt > SraAccList_novos.txt
else
    echo "Arquivos 'baixados.txt' e 'SraAccList_novos.txt' já existem. Pulando criação inicial."
fi

# Função para verificar se um arquivo está vazio
check_empty_file() {
    if [[ ! -s $1 ]]; then
        echo "Nenhum arquivo para processar. Todos os arquivos foram baixados com sucesso."
        exit 0
    fi
}

# Função para processar cada linha do arquivo (cada ID SRA)
process_line() {
    local sra_id=$1
    local retry=0

    while true; do
        echo
        echo "Baixando $sra_id"
        echo
```

```
        # Baixando arquivo com prefetch e capturando toda a saída
        echo "Iniciando prefetch para $sra_id..."
        prefetch $sra_id --max-size 1000000000000000000000000000000000000
        local prefetch_exit_code=$?

        # Verifica se o comando prefetch foi bem-sucedido
        if [[ $prefetch_exit_code -ne 0 ]]; then
            # Se houver erro de serviço externo, tenta novamente
            if prefetch $sra_id 2>&1 | grep -q "Failed to call external
services."; then
                echo "Falha ao chamar serviços externos durante prefetch.
Tentativa: $((retry+1)) para $sra_id."
                echo
                sleep 10
                ((retry++))
                continue
            else
                # Se houver outro erro inesperado, retorna falha
                echo "Erro inesperado durante prefetch"
                echo
                return 1
            fi
        fi

        # Convertendo arquivo SRA para fastq com fasterq-dump
        fasterq-dump $sra_id
        if [[ $? -eq 3 ]]; then
            # Se o arquivo já existir, realiza comandos adicionais e atualiza a
lista
            echo "Erro de arquivo já existente. Executando comandos adicionais."
            echo # Adiciona uma linha em branco

            # Atualiza a lista de arquivos baixados
            update_downloaded_list

            return 0
        fi

        # Se tudo deu certo, sai do loop
        break
    done
}

# Função para atualizar a lista de arquivos baixados
update_downloaded_list() {
    # Atualiza 'baixados.txt' com a lista de diretórios baixados
    for dir in */ ; do
        echo "${dir%/}"
```

```
    done | sort | uniq > baixados.txt

    # Atualiza 'SraAccList_novos.txt' com os IDs que ainda precisam ser baixados
    grep -vxFf baixados.txt SraAccList.txt > SraAccList_novos.txt
}

# Loop externo que continuará até que todos os arquivos sejam baixados
while true; do
    # Verifica se há arquivos a serem processados
    check_empty_file "SraAccList_novos.txt"

    # Loop para ler o arquivo de entrada 'SraAccList_novos.txt' linha por linha
    while IFS=$'\t' read -r linha1 linha2 linha3 linha4 linha5; do
        need_retry=false
        while true; do
            # Processa cada linha, que corresponde a um ID SRA
            process_line "$linha1"

            # Se o process_line retornar 1, significa que houve falha e precisa
tentar novamente
            if [[ $? -eq 1 ]]; then
                echo "Reiniciando tentativa para o arquivo $linha1."
                echo
                need_retry=true
                sleep 10
                continue
            fi

            # Se o process_line retornar 0, significa que o download foi bem-
sucedido
            break
        done
    done < SraAccList_novos.txt

    # Atualiza a lista de arquivos baixados após tentar todos os IDs
    update_downloaded_list

    # Verifica novamente se há arquivos a serem processados
    if [[ ! -s SraAccList_novos.txt ]]; then
        echo "Todos os arquivos foram baixados com sucesso."
        break
    fi

    # Se houver falhas, tenta baixar novamente
    if $need_retry; then
        echo "Tentando novamente baixar arquivos que falharam..."
        echo
    fi
done
```

APÊNDICE C - SCRIPT PARA DIVISÃO DE RUNS

```
#!/bin/bash

# Script: split_run_size.sh
# Descrição: Divide uma lista de arquivos fastq.gz em múltiplos arquivos de
execução (Run) com base no tamanho total especificado.
# Uso: ./split_run_size.sh fastqSizesRuns.txt XXXGB X

# Verifica se os argumentos necessários foram fornecidos
if [ "$#" -ne 3 ]; then
    echo "Uso: $0 fastqSizesRuns.txt TAMANHO_MAXIMO NUMERO_INICIAL (exemplo: 200GB
5)"
    exit 1
fi

INPUT_FILE="$1"
MAX_SIZE="$2"
START_RUN_NUMBER="$3"

# Verifica se o arquivo de entrada existe
echo "Verificando se o arquivo de entrada existe: $INPUT_FILE"
if [ ! -f "$INPUT_FILE" ]; then
    echo "Erro: Arquivo de entrada $INPUT_FILE não encontrado."
    exit 1
fi

# Função para converter tamanho para bytes
# Esta função recebe uma string representando um tamanho (e.g., "200GB") e converte
para bytes
size_to_bytes() {
    local size_str="$1"
    size_str=$(echo "$size_str" | tr -d "'\"")  # Remove aspas
    size_str=$(echo "$size_str" | sed 's/,/./g')  # Substitui vírgulas por pontos
    number=$(echo "$size_str" | sed -E 's/^([0-9.]+)[KMGT]?B?$/\1/')  # Extrai a
parte numérica
    unit=$(echo "$size_str" | sed -E 's/^[0-9.]+([KMGT]?B?)$/\1/')  # Extrai a
unidade
    case "$unit" in
        K|KB|k|kb) bytes=$(awk "BEGIN {print $number * 1000}") ;;
        M|MB|m|mb) bytes=$(awk "BEGIN {print $number * 1000 * 1000}") ;;
        G|GB|g|gb) bytes=$(awk "BEGIN {print $number * 1000 * 1000 * 1000}") ;;
        T|TB|t|tb) bytes=$(awk "BEGIN {print $number * 1000 * 1000 * 1000 * 1000}")
;;
        B|b) bytes=$(awk "BEGIN {print $number}") ;;
        *) bytes=0 ;;
    esac
```

```
    echo "$bytes"
}

# Função para converter bytes para formato legível
# Esta função converte um valor em bytes para um formato mais legível (e.g.,
"200GB")
bytes_to_human() {
    local bytes=$1
    local unit="B"
    local size=$bytes
    if (( bytes >= 1000**4 )); then
        size=$(awk "BEGIN {printf \"%.2f\", $bytes/1000000000000}")
        unit="TB"
    elif (( bytes >= 1000**3 )); then
        size=$(awk "BEGIN {printf \"%.2f\", $bytes/1000000000}")
        unit="GB"
    elif (( bytes >= 1000**2 )); then
        size=$(awk "BEGIN {printf \"%.2f\", $bytes/1000000}")
        unit="MB"
    elif (( bytes >= 1000 )); then
        size=$(awk "BEGIN {printf \"%.2f\", $bytes/1000}")
        unit="KB"
    fi
    echo "${size}${unit}"
}

# Converte o TAMANHO_MAXIMO para bytes
echo "Convertendo TAMANHO_MAXIMO para bytes: $MAX_SIZE"
MAX_SIZE_BYTES=$(size_to_bytes "$MAX_SIZE")
if [ "$MAX_SIZE_BYTES" -eq 0 ]; then
    echo "Erro: Formato de TAMANHO_MAXIMO inválido. Use formatos como 200GB, 100MB,
etc."
    exit 1
fi

# Inicializa variáveis
echo "Inicializando variáveis"
run_number=$START_RUN_NUMBER  # Começa do número especificado
current_size=0
output_file="Run${run_number}.txt"
> "$output_file"  # Cria ou esvazia o arquivo de saída inicial

declare -A run_sizes  # Declaração de um array associativo para armazenar os
tamanhos dos arquivos de execução

prev_id=""
group_files=()
group_size=0
```

```
# Função para adicionar um grupo de arquivos ao run atual
# Verifica se o tamanho do grupo ultrapassa o limite e cria um novo run, se
necessário
add_group_to_run() {
    local group_size=$1
    local -n group_files_arr=$2  # Referência para o array de arquivos

    # Verifica se o grupo atual excede o tamanho máximo permitido
    if (( current_size + group_size > MAX_SIZE_BYTES )); then
        if (( current_size == 0 )); then
            # Caso em que o tamanho do grupo é maior que o permitido e não há um
run atual
            echo "O tamanho do grupo é maior que o tamanho máximo permitido e
nenhum run atual existente."
        else
            # Finaliza o run atual e cria um novo
            run_sizes["Run${run_number}.txt"]=$current_size
            size_human=$(bytes_to_human "$current_size")
            echo "Run${run_number}.txt criado com tamanho: $size_human"
            run_number=$((run_number + 1))
            output_file="Run${run_number}.txt"
            > "$output_file"
            current_size=0
        fi
    fi

    # Adiciona cada arquivo do grupo ao run atual
    for f in "${group_files_arr[@]}"; do
        echo "Adicionando arquivo ao run atual: $f"
        echo "$f" >> "$output_file"
    done
    current_size=$((current_size + group_size))
}

line_count=0

echo "Iniciando processamento do arquivo de entrada"
# Loop para processar cada linha do arquivo de entrada
while IFS= read -r line || [ -n "$line" ]; do
    line_count=$((line_count + 1))
    echo "Processando linha $line_count: $line"
    if [ -z "$line" ]; then
        echo "Linha vazia encontrada, ignorando"
        continue
    fi

    # Extrai o nome do arquivo e o tamanho da linha atual
    filename=$(echo "$line" | awk '{print $1}')
    size_str=$(echo "$line" | awk '{print $2}' | tr -d "'\"" | sed 's/,/./g')
```

```
    # Converte o tamanho para bytes
    size_bytes=$(size_to_bytes "$size_str")

    if [ "$size_bytes" -eq 0 ]; then
        echo "Aviso: Formato de tamanho '$size_str' para o arquivo '$filename'
inválido. Ignorando."
        continue
    fi

    # Extrai o ID do arquivo, removendo sufixos como "_1" ou "_2" e a extensão
.fastq.gz
    id=$(echo "$filename" | sed -E 's/(_[12])?\.fastq\.gz$//')

    echo "ID do arquivo: $id"
    # Verifica se o ID é igual ao do arquivo anterior, indicando que faz parte do
mesmo grupo
    if [ "$id" == "$prev_id" ]; then
        echo "ID igual ao anterior ($prev_id), adicionando ao grupo atual"
        group_files+=("$filename")
        group_size=$((group_size + size_bytes))
        add_group_to_run "$group_size" group_files
        group_files=()
        group_size=0
        prev_id=""
    else
        # Caso o ID seja diferente, finaliza o grupo anterior (se houver) e inicia
um novo
        if [ "${#group_files[@]}" -gt 0 ]; then
            echo "Novo ID encontrado, adicionando grupo atual ao run e iniciando
novo grupo"
            add_group_to_run "$group_size" group_files
            group_files=()
            group_size=0
        fi
        group_files=("$filename")
        group_size="$size_bytes"
        prev_id="$id"
    fi
done < "$INPUT_FILE"

# Adiciona o último grupo ao run, se houver
if [ "${#group_files[@]}" -gt 0 ]; then
    echo "Adicionando grupo final ao run"
    add_group_to_run "$group_size" group_files
fi

# Finaliza o último run
run_sizes["Run${run_number}.txt"]=$current_size
```

```
size_human=$(bytes_to_human "$current_size")
echo "Run${run_number}.txt criado com tamanho: $size_human"
echo "Total de linhas processadas: $line_count"
echo "Divisão concluída em Run${START_RUN_NUMBER}.txt até Run${run_number}.txt"
```

APÊNDICE D - SCRIPT PARA EXECUÇÃO DO NFCORE-RNA/SEQ

```
#!/bin/bash
set -e  # Faz o script parar em caso de erro

# Autor: David Daniel
# Descrição: Este script automatiza o processamento e empacotamento de runs de
dados de RNA-Seq usando o pipeline nf-core/rnaseq.
# Ele realiza as seguintes etapas principais:
# 1. Verifica se cada run já foi processada e registrada em um log.
# 2. Exclui arquivos temporários ('work') para liberar espaço.
# 3. Cria um pacote tar.gz dos dados processados para armazenamento eficiente.
# 4. Processa os dados de RNA-Seq com o Nextflow, gerando relatórios de qualidade
com o MultiQC.
# 5. Registra as runs processadas em um log.

# Definição de Cores para Melhorar a Estética da Saída
GREEN='[0;32m'  # Cor verde para mensagens de sucesso
YELLOW='[1;33m'  # Cor amarela para avisos
RED='[0;31m'     # Cor vermelha para erros
BLUE='[0;34m'    # Cor azul para informações
NC='[0m'         # Sem cor

# Funções para Exibir Mensagens com Cores
function echo_info() {
    echo -e "${BLUE}$1${NC}"
}

function echo_success() {
    echo -e "${GREEN}$1${NC}"
}

function echo_warning() {
    echo -e "${YELLOW}$1${NC}"
}

function echo_error() {
    echo -e "${RED}$1${NC}"
}

# Função para Verificar Integridade do tar.gz usando tar -tzf
function verify_tar_integrity() {
    local tar_file="$1"

    # Verifica se o arquivo não está vazio
    if [ ! -s "$tar_file" ]; then
        echo_error "Arquivo $tar_file está vazio."
```

```
        return 1
    fi

    # Tenta listar o conteúdo do tar.gz
    if ! tar -tzf "$tar_file" > /dev/null 2>&1; then
        echo_error "Arquivo $tar_file está corrompido ou inválido."
        return 1
    fi

    echo_success "Arquivo $tar_file está íntegro."
    return 0
}

# Definições de Diretórios e Arquivos
BASE_DIR=$(pwd)  # Define o diretório base como o diretório atual
SAMPLE_SHEET_ORIGINAL="/media/hd9/star_usage/david-
sfru/Requisitos/sample_sheet.csv"  # Caminho do arquivo de sample sheet original
LOG_FILE="$BASE_DIR/processed_runs.log"  # Arquivo de log para registrar runs
processadas
DEST_DIR="/media/lgbio-nas1/davidsantos/TCC/Star-Nfcore/Analises"  # Diretório de
destino para os arquivos empacotados

# Verificar se o diretório base existe
if [ ! -d "$BASE_DIR" ]; then
    echo_error "Erro: O diretório base $BASE_DIR não existe."
    exit 1
fi

# Verificar se o diretório de destino existe
if [ ! -d "$DEST_DIR" ]; then
    echo_error "Erro: O diretório de destino $DEST_DIR não existe."
    exit 1
fi

# Criar o arquivo de log se não existir
touch "$LOG_FILE"

# Navegar para o diretório base
cd "$BASE_DIR" || { echo_error "Erro: Não foi possível acessar $BASE_DIR."; exit 1;
}

# Gerar a lista de Runs
Runs=$(ls Run*.txt 2>/dev/null | grep -oP '\d+' | sort -n)  # Lista todos os
arquivos Run*.txt, extrai os números e os ordena

# Verificar se existem Runs para processar
if [ -z "$Runs" ]; then
    echo_error "Erro: Nenhum arquivo Run*.txt encontrado em $BASE_DIR."
    exit 1
```

```
fi

# Função para Verificar se uma Run já foi Processada
function is_run_processed() {
    local run_number="$1"
    grep -qw "Run$run_number" "$LOG_FILE"  # Verifica se a run já está registrada
no log
}

# Loop para cada Run
for Run in $Runs; do
    echo "----------------------------------------"
    echo_info "Processando Run$Run"
    echo "----------------------------------------"

    # Verificar se a Run já foi processada
    if is_run_processed "$Run"; then
        echo_success "Run$Run já foi processada anteriormente. Pulando para a
próxima Run..."
        continue
    fi

    # Definir variáveis
    RunDir="$BASE_DIR/Run$Run"  # Diretório da run atual
    OutputDir="$RunDir/star-rsemN$Run"  # Diretório de saída para a run atual
    MultiqcReport="$OutputDir/multiqc/star_rsem/multiqc_report.html"  # Caminho do
relatório MultiQC
    RunTxt="$BASE_DIR/Run$Run.txt"  # Caminho do arquivo de entrada da run
    TarFileDest="$DEST_DIR/Run$Run.tar.gz"  # Caminho final do arquivo tar.gz no
destino
    TempTarFileDest="$BASE_DIR/Run$Run.tar.gz.tmp"  # Caminho temporário do arquivo
tar.gz no diretório base

    # Verificar se o arquivo Run$Run.txt existe
    if [ ! -f "$RunTxt" ]; then
        echo_error "Erro: O arquivo $RunTxt não foi encontrado."
        continue
    fi

    # Verificar se o arquivo tar.gz já existe e está íntegro no destino
    if [ -f "$TarFileDest" ]; then
        echo_info "Verificando a integridade do arquivo tar existente no destino:
$TarFileDest"
        if verify_tar_integrity "$TarFileDest"; then
            echo_success "Run$Run já foi empacotada e está íntegra. Adicionando ao
log e pulando para a próxima Run..."
            echo "Run$Run" >> "$LOG_FILE"
            continue
        else
```

```
            echo_warning "Run$Run possui um arquivo tar existente no destino, mas
ele está corrompido ou incompleto. Recriando o tar.gz..."
            # Remove o tar.gz corrompido no destino
            rm -f "$TarFileDest"
            if [ $? -ne 0 ]; then
                echo_error "Erro ao remover o arquivo tar corrompido: $TarFileDest"
                continue
            fi
        fi
    fi

    # Verificar se o relatório MultiQC já existe
    if [ -f "$MultiqcReport" ]; then
        echo_success "Relatório MultiQC para Run$Run já existe em $MultiqcReport.
Empacotando a Run..."

        # ============================================
        # EXCLUI A PASTA 'work' para otimização de espaço
        # ============================================
        WorkDir="$RunDir/work"
        if [ -d "$WorkDir" ]; then
            echo_info "Pasta 'work' encontrada em $RunDir. Excluindo..."
            rm -rf "$WorkDir"
            if [ $? -eq 0 ]; then
                echo_success "Pasta 'work' excluída com sucesso."
            else
                echo_error "Erro ao excluir a pasta 'work' em $RunDir."
                continue  # Pular para a próxima Run se não for possível excluir
            fi
        else
            echo_info "Pasta 'work' não encontrada em $RunDir."
        fi
        # ============================================

        # Empacotar a Run
        echo_info "Criando arquivo tar temporário em $BASE_DIR para Run$Run..."
        tar -czvf "$TempTarFileDest" -C "$BASE_DIR" "Run$Run"  # Cria um arquivo
tar.gz temporário contendo a pasta da run
        if [ $? -ne 0 ]; then
            echo_error "Erro: Falha ao criar o arquivo tar temporário:
$TempTarFileDest"
            # Remove o arquivo temporário se a criação falhar
            rm -f "$TempTarFileDest"
            continue
        fi

        echo_success "Arquivo tar temporário criado com sucesso: $TempTarFileDest"

        # Verificar a integridade do arquivo tar temporário usando tar -tzf
```

```
        echo_info "Verificando a integridade do arquivo tar temporário: $TempTarFileDest"
        if verify_tar_integrity "$TempTarFileDest"; then
            echo_success "Arquivo tar temporário $TempTarFileDest está válido."
        else
            echo_error "Erro: O arquivo tar temporário $TempTarFileDest está corrompido."
            # Remove o arquivo temporário
            rm -f "$TempTarFileDest"
            continue
        fi

        # Mover o arquivo tar temporário para o destino
        echo_info "Movendo o arquivo tar temporário para o diretório final: $TarFileDest"
        mv "$TempTarFileDest" "$TarFileDest"  # Move o arquivo tar.gz temporário para o diretório de destino
        if [ $? -ne 0 ]; then
            echo_error "Erro ao mover o arquivo tar para o diretório final: $TarFileDest"
            # Remove o arquivo temporário caso o movimento falhe
            rm -f "$TempTarFileDest"
            continue
        fi

        echo_success "Arquivo tar movido com sucesso para $TarFileDest."

        # Excluir a pasta Run após a compressão bem-sucedida
        echo_info "Excluindo a pasta Run$Run..."
        rm -rf "$RunDir"  # Exclui o diretório da run após a compressão bem-sucedida
        if [ $? -eq 0 ]; then
            echo_success "Pasta Run$Run excluída com sucesso."
        else
            echo_error "Erro ao excluir a pasta Run$Run."
        fi

        # Registrar a Run como processada no log (opcional)
        echo "Run$Run" >> "$LOG_FILE"  # Adiciona a run ao log de processados
        echo_success "Run$Run marcada como concluída no log."

        continue
    fi

    # Se o relatório MultiQC não existe, processar com Nextflow
    echo_info "Relatório MultiQC para Run$Run não encontrado. Iniciando processamento com Nextflow."

    # Criar diretório para o Run (se não existir)
```

```
    mkdir -p "$RunDir"

    # Mudar para o diretório do Run
    cd "$RunDir" || { echo_error "Erro: Não foi possível acessar $RunDir."; 
continue; }

    # Executar o comando grep e adicionar o cabeçalho
    echo_info "Gerando sample_sheet.csv para Run$Run..."
    grep -F -f "$RunTxt" "$SAMPLE_SHEET_ORIGINAL" | \
    sed "1i\sample,fastq_1,fastq_2,strandedness" > sample_sheet.csv  # Cria o 
arquivo sample_sheet.csv com o cabeçalho adequado

    # Verificar se sample_sheet.csv foi criado corretamente
    if [ ! -s "sample_sheet.csv" ]; then
        echo_error "Erro: sample_sheet.csv está vazio ou não foi criado 
corretamente para Run$Run."
        cd "$BASE_DIR"
        continue
    fi

    # Criar links simbólicos para os arquivos listados em Run$Run.txt
    echo_info "Criando links simbólicos para arquivos listados em $RunTxt..."
    while IFS= read -r line; do
        SOURCE_FILE="/media/lgbio-nas1/davidsantos/TCC/RNASeq/$line"  # Caminho do 
arquivo de origem
        if [ -f "$SOURCE_FILE" ]; then
            ln -sf "$SOURCE_FILE" .  # Cria um link simbólico para cada arquivo 
listado
        else
            echo_warning "Aviso: O arquivo $SOURCE_FILE não existe. Link simbólico 
não criado."
        fi
    done < "$RunTxt"

    # Criar links simbólicos para os arquivos necessários
    echo_info "Criando links simbólicos para arquivos de requisitos..."
    ln -sf /media/hd9/star_usage/david-sfru/Requisitos/sfru/ .
    ln -sf /media/hd9/star_usage/david-sfru/Requisitos/Sfru.gtf.gz .
    ln -sf /media/hd9/star_usage/david-sfru/Requisitos/Sfru.fna .
    ln -sf /media/hd9/star_usage/david-sfru/Requisitos/rsem .

    # Executar o comando Nextflow com Feedback Visual
    echo_info "Iniciando execução do Nextflow para Run$Run..."
    nextflow run nf-core/rnaseq \
        -c /media/hd12-Renata/lgbio_guest/Nf-Core/Time.config \ # Arquivo de 
configuração para aumentar tempo de execução para etapas longas
        --input sample_sheet.csv \
        --aligner star_rsem \
        --star_index sfru \
```

```
        --rsem_index rsem \
        --fasta Sfru.fna \
        --gtf Sfru.gtf.gz \
        --outdir "$OutputDir" \
        --skip_biotype_qc \
        --remove_ribo_rna \
        --max_cpus 20 \
        -profile singularity \
        --max_time 200h \
        -resume  # Resume o pipeline em caso de erro

    # Após a execução, verificar se o relatório MultiQC foi gerado
    if [ -f "$MultiqcReport" ]; then
        echo_success "Processamento de Run$Run finalizado com sucesso. Relatório
MultiQC encontrado em $MultiqcReport."
    else
        echo_error "Erro: O relatório MultiQC para Run$Run não foi encontrado em
$MultiqcReport."
        cd "$BASE_DIR"
        continue
    fi

    # Excluir a pasta 'work' da Run atual antes de compactar
    echo_info "Excluindo a pasta 'work' de Run$Run..."
    WORK_DIR="$RunDir/work"

    if [ -d "$WORK_DIR" ]; then
        rm -rf "$WORK_DIR"  # Exclui a pasta 'work' para liberar espaço
        if [ $? -eq 0 ]; then
            echo_success "Pasta 'work' excluída com sucesso para Run$Run."
        else
            echo_error "Erro: Falha ao excluir a pasta 'work' para Run$Run."
            cd "$BASE_DIR"
            continue
        fi
    else
        echo_warning "Pasta 'work' não encontrada em $RunDir. Nenhuma exclusão
necessária."
    fi

    # Voltar para o diretório base antes de empacotar
    cd "$BASE_DIR" || { echo_error "Erro: Não foi possível retornar para
$BASE_DIR."; exit 1; }

    # Empacotamento do Run
    echo_info "Iniciando o empacotamento do Run$Run..."

    # Criar arquivo tar.gz temporário no BASE_DIR
    echo_info "Criando arquivo tar temporário em $BASE_DIR para Run$Run..."
```

```
    tar -czvf "$TempTarFileDest" -C "$BASE_DIR" "Run$Run"  # Cria um arquivo tar.gz
contendo a pasta da run
    if [ $? -ne 0 ]; then
        echo_error "Erro: Falha ao criar o arquivo tar temporário:
$TempTarFileDest"
        # Remove o arquivo temporário se a criação falhar
        rm -f "$TempTarFileDest"
        continue
    fi

    echo_success "Arquivo tar temporário criado com sucesso: $TempTarFileDest"

    # Verificar a integridade do arquivo tar temporário usando tar -tzf
    echo_info "Verificando a integridade do arquivo tar temporário:
$TempTarFileDest"
    if verify_tar_integrity "$TempTarFileDest"; then
        echo_success "Arquivo tar temporário $TempTarFileDest está válido."
    else
        echo_error "Erro: O arquivo tar temporário $TempTarFileDest está
corrompido."
        # Remove o arquivo temporário
        rm -f "$TempTarFileDest"
        continue
    fi

    # Mover o arquivo tar temporário para o destino
    echo_info "Movendo o arquivo tar temporário para o diretório final:
$TarFileDest"
    mv "$TempTarFileDest" "$TarFileDest"  # Move o arquivo tar.gz para o diretório
de destino
    if [ $? -ne 0 ];then
        echo_error "Erro ao mover o arquivo tar para o diretório final:
$TarFileDest"
        # Remove o arquivo temporário caso o movimento falhe
        rm -f "$TempTarFileDest"
        continue
    fi

    echo_success "Arquivo tar movido com sucesso para $TarFileDest."

    # Excluir a pasta Run após a compressão bem-sucedida
    echo_info "Excluindo a pasta Run$Run..."
    rm -rf "$RunDir"  # Exclui o diretório da run após a compactação
    if [ $? -eq 0 ]; then
        echo_success "Pasta Run$Run excluída com sucesso."
    else
        echo_error "Erro ao excluir a pasta Run$Run."
    fi
```

```
    # Registrar a Run como processada no log (opcional)
    echo "Run$Run" >> "$LOG_FILE"  # Adiciona a run ao log de processados
    echo_success "Run$Run marcada como concluída no log."

done

echo "----------------------------------------"
echo_success "Todos os Runs foram processados com sucesso."
echo "----------------------------------------"
```

APÊNDICE E - SCRIPT PARA EXECUÇÃO DO WGCNA

```
# Descrição:
# Este script realiza uma análise de expressão gênica utilizando dados de RNA-
Seq. Ele inclui as seguintes etapas:
# 1. Instalação e carregamento das bibliotecas necessárias para a análise.
# 2. Leitura dos dados de expressão gênica e metadados.
# 3. Filtragem e limpeza dos dados, incluindo a remoção de genes com contagens
zero em todas as amostras.
# 4. Normalização dos dados de expressão utilizando o DESeq2.
# 5. Análise de variância dos genes e filtragem com base no quantil de 95%.
# 6. Geração de gráficos, como violin plots para visualizar a distribuição da
expressão gênica por tecido.
# 7. Construção de redes de co-expressão gênica usando WGCNA e identificação de
módulos de co-expressão.
# 8. Geração de dendrogramas e visualizações dos módulos.



library(tidyverse)
library(magrittr)
library(WGCNA)
library(DESeq2)
library(genefilter)

# Carregar os dados de expressão gênica e metadados
data <- readr::read_delim("/media/hd9/star_usage/david-
sfru/R/AnalisesFinal/TPMTecidos.tsv",  delim = "\t")
metadata <- readr::read_tsv("/media/hd9/star_usage/david-
sfru/R/AnalisesFinal/Tecidosids.txt")

# Verificar as primeiras linhas dos metadados e dados de expressão
head(metadata)
data[1:5, 1:10]

# Renomear a primeira coluna para 'GeneId'
names(data)[1] <- "GeneId"

# Obter os nomes das amostras de expressão (removendo a coluna 'GeneId')
expression_samples <- colnames(data)[-1]

# Identificar amostras faltantes nos metadados
missing_samples <- setdiff(expression_samples, metadata$RUN)
print(missing_samples)

# Identificar amostras comuns entre dados de expressão e metadados
common_samples <- intersect(expression_samples, metadata$RUN)
```

```
# Filtrar os dados de expressão para manter apenas as amostras comuns
data_filtered <- data %>% select(GeneId, all_of(common_samples))

# Verificar duplicatas na coluna RUN dos metadados
duplicated_runs <- duplicated(metadata$RUN)
if (any(duplicated_runs)) {
  duplicated_entries <- metadata$RUN[duplicated_runs]
  stop(paste("Metadados contêm RUN duplicados:",
paste(unique(duplicated_entries), collapse = ", ")))
} else {
  print("Nenhuma duplicata encontrada na coluna RUN.")
}

# Filtrar e ordenar os metadados para corresponder à ordem das amostras nos
dados de expressão
metadata_ordered <- metadata %>%
  filter(RUN %in% common_samples) %>%
  arrange(match(RUN, common_samples))

# Verificar se o ordenamento dos metadados está correto
if (!all(metadata_ordered$RUN == common_samples)) {
  stop("Erro ao ordenar metadados. Verifique se há duplicatas ou
inconsistências.")
} else {
  print("Metadados ordenados corretamente.")
}

# Criar o meta_df final, alinhado com os dados de expressão filtrados
meta_df <- data.frame(
  Sample = metadata_ordered$RUN,      # As amostras
  Tissue = metadata_ordered$Tecido    # Os respectivos tecidos
)

# Verificar as primeiras linhas do meta_df
head(meta_df)

# Transformar os dados de expressão para formato longo e mesclar com os
metadados
mdata <- data_filtered %>%
  pivot_longer(cols = -GeneId, names_to = "Sample", values_to = "Expression")
%>%
  left_join(meta_df, by = "Sample") %>%
  mutate(Tissue = as.factor(Tissue))  # Garantir que os tecidos sejam tratados
como fatores

# Ajustar o tamanho da imagem e a resolução
png("plot_output_by_tissue_high_quality.png", width = 35, height = 25, units =
"in", res = 600)
```

```
# Plotar os dados de expressão com agrupamento por tecido
p <- mdata %>%
  ggplot(aes(x = Sample, y = Expression)) +
  geom_violin() +                                    # Gráfico de violino para
mostrar a distribuição dos dados de expressão
  geom_point(alpha = 0.2) +                           # Adicionar pontos
individuais para cada amostra, com transparência
  theme_bw() +
  theme(
    axis.text.x = element_text(angle = 90, size = 10, hjust = 1, vjust = 0.5),
# Ajustar o alinhamento do texto do eixo X
    axis.text.y = element_text(size = 15),
# Ajustar o tamanho do texto do eixo Y
    axis.title.x = element_text(size = 30, margin = margin(t = 15)),
# Aumentar o distanciamento da legenda do eixo X
    axis.title.y = element_text(size = 30, margin = margin(r = 15)),
# Aumentar o distanciamento da legenda do eixo Y
    strip.text = element_text(size = 25),
# Aumentar o tamanho dos títulos dos grupos
    panel.spacing = unit(1, "lines")
# Aumentar o espaçamento entre os gráficos
  ) +
  labs(x = "Samples", y = "RNA Seq Counts") +
  facet_wrap(~ Tissue, scales = "free_x", ncol = 2)  # Facetar o gráfico por
tecido, com escalas independentes para cada eixo X

# Salvar o gráfico
print(p)
dev.off()

# Converter os dados de expressão filtrados em uma matriz para análises
posteriores
de_input_filtered <- as.matrix(data_filtered[, -1])
rownames(de_input_filtered) <- data_filtered$GeneId

# Verificar quantos genes têm todas as contagens iguais a zero
all_zero_genes <- rowSums(de_input_filtered == 0) == ncol(de_input_filtered)

# Remover genes com todas as contagens iguais a zero
de_input_filtered <- de_input_filtered[!all_zero_genes, ]

# Verificar a dimensão após o filtro
dim(de_input_filtered)

# Adicionar um pseudocount de 1 para evitar problemas com contagens iguais a
zero
de_input_filtered_pseudocount <- de_input_filtered + 1

# Criar o DESeqDataSet com a matriz filtrada
```

```
dds <- DESeqDataSetFromMatrix(
  countData = round(de_input_filtered_pseudocount),
  colData = meta_df,
  design = ~ Tissue
)

# Executar o DESeq para realizar a normalização dos dados
dds <- DESeq(dds)

# Aplicar a transformação de variância estabilizada para os dados normalizados
vsd <- varianceStabilizingTransformation(dds)

# Extrair os dados transformados para análise posterior
wpn_vsd <- assay(vsd)

# Verificar a dimensão dos dados transformados
dim(wpn_vsd)

# Calcular a variância por gene nos dados transformados
rv_wpn <- rowVars(wpn_vsd)

# Definir o quantil de 95% para filtrar os genes com alta variância
q95_wpn <- quantile(rv_wpn, 0.95)

# Manter apenas os genes cuja variância é maior que o quantil de 95%
expr_normalized <- wpn_vsd[rv_wpn > q95_wpn, ]

# Verificar a dimensão após o filtro
dim(expr_normalized)

# Definir o nome do arquivo PNG, tamanho da imagem e resolução
png("violin_plot_expression.png", width = 10, height = 6, units = "in", res =
300)

# Mesclar os dados de expressão normalizada com os metadados
expr_normalized_df <- data.frame(expr_normalized) %>%
  mutate(Gene_id = row.names(expr_normalized)) %>%
  pivot_longer(-Gene_id, names_to = "Sample", values_to = "Expression") %>%
  left_join(meta_df, by = c("Sample" = "Sample"))  # Mesclar com os metadados

# Gerar o gráfico violin plot usando o tecido no eixo X
png("violin_plot_by_tissue.png", width = 10, height = 6, units = "in", res =
300)

expr_normalized_df %>%
  ggplot(aes(x = Tissue, y = Expression)) +  # Alterar para que o eixo X mostre
os tecidos
  geom_violin() +
  geom_point(alpha = 0.2) +
```

```
  theme_bw() +
  theme(
    axis.text.x = element_text(angle = 90),  # Girar os nomes dos tecidos para
facilitar a leitura
    plot.title = element_text(face = "bold", margin = margin(b = 10)),  # Título
principal em negrito e afastado
    axis.title.x = element_text(face = "bold", margin = margin(t = 15)),  #
Título do eixo X em negrito e afastado
    axis.title.y = element_text(face = "bold", margin = margin(r = 10))   #
Título do eixo Y em negrito e afastado
  ) +
  ylim(0, NA) +  # Definir o limite inferior do gráfico como zero
  labs(
    title = "Normalized and 95% Quantile Expression by Tissue",
    x = "Tissue",
    y = "Normalized Expression"
  )

dev.off()  # Salvar o gráfico como PNG

# Transpor a matriz de expressão para que as linhas sejam amostras e as colunas
sejam genes
input_mat <- t(expr_normalized)

# Verificar a estrutura da matriz transposta
dim(input_mat)  # Deve retornar [número de amostras] x [número de genes
filtrados]

# Permitir multi-threading (opcional) para acelerar a análise
allowWGCNAThreads(12)

# Escolher uma faixa de valores de soft-threshold para testar
powers <- c(c(1:10), seq(from = 12, to = 20, by = 2))

# Analisar a topologia da rede para escolher o soft threshold
sft <- pickSoftThreshold(
  input_mat,
  powerVector = powers,
  verbose = 5
)

# Plotar os resultados da escolha do soft threshold
png("soft_threshold_plots.png", width = 10, height = 6, units = "in", res = 300)
# Definir layout para dois gráficos lado a lado
par(mfrow = c(1, 2))
cex1 <- 0.9  # Tamanho do texto

# Plotar o gráfico de Scale Free Topology Fit
plot(
```

```
  sft$fitIndices[, 1],
  -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
  xlab = "Soft Threshold (power)",
  ylab = "Scale Free Topology Model Fit, signed R^2",
  main = "Scale independence"
)
text(
  sft$fitIndices[, 1],
  -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
  labels = powers,
  cex = cex1,
  col = "red"
)
abline(h = 0.56, col = "red")  # Linha de referência para o valor R² = 0.90

# Plotar o gráfico de Mean Connectivity
plot(
  sft$fitIndices[, 1],
  sft$fitIndices[, 5],
  xlab = "Soft Threshold (power)",
  ylab = "Mean Connectivity",
  type = "n",
  main = "Mean connectivity"
)
text(
  sft$fitIndices[, 1],
  sft$fitIndices[, 5],
  labels = powers,
  cex = cex1,
  col = "red"
)

dev.off()

# Definir o soft-threshold escolhido (exemplo: power 9)
picked_power <- 9

# Evitar conflitos de namespace com a função 'cor'
temp_cor <- cor       # Salvar a função original 'cor'
cor <- WGCNA::cor     # Usar a função 'cor' do pacote WGCNA

# Construir a rede de co-expressão e identificar módulos
netwk <- blockwiseModules(
  input_mat,
  power = picked_power,
  networkType = "signed",
  deepSplit = 2,
  pamRespectsDendro = FALSE,
  minModuleSize = 30,
```

```
  maxBlockSize = 4000,
  reassignThreshold = 0,
  mergeCutHeight = 0.25,
  saveTOMs = TRUE,
  saveTOMFileBase = "ER",
  numericLabels = TRUE,
  verbose = 3
)

# Restaurar a função 'cor' original
cor <- temp_cor

# Converter as etiquetas dos módulos em cores para o gráfico
mergedColors <- labels2colors(netwk$colors)

# Definir o nome do arquivo PNG, tamanho da imagem e resolução
png("dendrogram_modules.png", width = 15, height = 10, units = "in", res = 300)

# Gerar o dendrograma com as cores dos módulos
plotDendroAndColors(
  netwk$dendrograms[[1]],  # Primeiro dendrograma dos blocos
  mergedColors[netwk$blockGenes[[1]]],  # Cores correspondentes aos módulos dos
genes
  "Module colors",  # Título do gráfico
  dendroLabels = FALSE,  # Não mostrar labels no dendrograma
  hang = 0.03,  # Controlar o alinhamento das folhas do dendrograma
  addGuide = TRUE,  # Adicionar uma linha-guia para facilitar a visualização
  guideHang = 0.05  # Controlar o comprimento da linha-guia
)

# Finalizar o gráfico e salvar o arquivo
dev.off()

# Gerar o dataframe com genes e seus respectivos módulos
module_df <- data.frame(
  gene_id = names(netwk$colors),
  colors = labels2colors(netwk$colors)
)

# Salvar a tabela em um arquivo
write_delim(module_df, file = "gene_modules.txt", delim = "\t")

# Obter os Eigengenes por módulo
MEs0 <- moduleEigengenes(input_mat, mergedColors)$eigengenes

# Reordenar os módulos para agrupar módulos similares
MEs0 <- orderMEs(MEs0)
module_order <- names(MEs0) %>% gsub("ME", "", .)
```

```
# Incluir os tratamentos e tecidos no dataframe de eigengenes
MEs0$treatment <- row.names(MEs0)
MEs0$Tecido <- metadata_ordered$Tecido  # Adicionar coluna de tecidos usando
'Tecido'

# Transformar os dados de eigengenes para o formato longo
mME <- MEs0 %>%
  pivot_longer(-c(treatment, Tecido)) %>%
  mutate(
    name = gsub("ME", "", name),  # Remover o "ME" dos nomes dos módulos
    name = factor(name, levels = module_order)  # Definir a ordem dos módulos
  )

# Criar o gráfico com os nomes dos tecidos no eixo X
plot <- mME %>%
  ggplot(aes(x = Tecido, y = name, fill = value)) +  # Usar Tecido no eixo X
  geom_tile() +
  theme_bw() +
  scale_fill_gradient2(
    low = "blue",
    high = "red",
    mid = "white",
    midpoint = 0,
    limit = c(-1, 1)
  ) +
  theme(
    axis.text.x = element_text(angle = 90, size = 8),  # Girar e ajustar o
tamanho do texto no eixo X
    axis.text.y = element_text(size = 10)  # Ajustar o tamanho do texto no eixo
Y
  ) +
  labs(title = "Module-Trait Relationships", y = "Modules", x = "Tissues", fill
= "corr")

# Salvar o gráfico como PNG
ggsave("module_trait_relationships_by_tissue.png", plot = plot, width = 10,
height = 8, dpi = 300)

# Gerar o dataframe com genes e seus respectivos módulos
module_df <- data.frame(
  gene_id = names(netwk$colors),
  colors = labels2colors(netwk$colors)
)

# Filtrar os dados normalizados para incluir apenas os genes presentes em
module_df
submod <- module_df  # Aqui você pode filtrar para módulos específicos, se
necessário
subexpr <- expr_normalized[submod$gene_id, ]  # Filtrar os genes normalizados
```

```
# Gerar o dataframe com as expressões normalizadas para os genes nos módulos
submod_df <- data.frame(subexpr) %>%
  mutate(gene_id = row.names(.)) %>%
  pivot_longer(-gene_id, names_to = "Sample", values_to = "Expression") %>%
  mutate(module = submod$colors[match(gene_id, submod$gene_id)])

# Verificar se a coluna 'Tissue' existe em 'submod_df' e, se não, adicionar a
partir dos metadados
if (!"Tissue" %in% colnames(submod_df)) {
  submod_df <- submod_df %>%
    left_join(meta_df, by = "Sample")  # Supondo que meta_df contém as colunas
'Sample' e 'Tissue'
}

# Adicionar coluna 'LifeStage' com base no nome do tecido
submod_df <- submod_df %>%
  mutate(
    LifeStage = case_when(
      grepl("Larva", Tissue) ~ "Larva",
      grepl("Pupa", Tissue) ~ "Pupa",
      grepl("Adult", Tissue) ~ "Adult",
      TRUE ~ "Other"
    )
  )

# Ajustar o mapeamento de cores dos módulos para tons mais escuros e corretos
module_color_map <- c(
  "blue" = "#1F78B4",        # Azul mais escuro
  "brown" = "#8B4513",       # Marrom escuro
  "green" = "#228B22",       # Verde escuro
  "grey" = "#696969",        # Cinza escuro
  "red" = "#B22222",         # Vermelho escuro
  "turquoise" = "#00CED1",   # Turquesa vibrante
  "yellow" = "#FFD700"       # Amarelo mais escuro e intenso
)

# Definir a ordem dos estágios de vida e ordenar o fator 'Tissue'
submod_df <- submod_df %>%
  mutate(
    LifeStage = factor(LifeStage, levels = c("Larva", "Pupa", "Adult")),
    Tissue = factor(Tissue, levels = unique(Tissue[order(LifeStage)]))
  )

# Gerar o gráfico de expressão normalizada por módulo e tecido, ordenado por
estágio de vida
png("expression_by_tissue_life_stage_ordered.png", width = 12, height = 8, units
= "in", res = 300)
```

```
submod_df %>%
  ggplot(aes(x = Tissue, y = Expression, group = gene_id)) +
  geom_line(aes(color = module), alpha = 0.2) +  # Linhas representando a
expressão normalizada de cada gene
  theme_bw() +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8,
margin = margin(t = 5)),
    strip.text = element_text(size = 10)
  ) +
  scale_x_discrete(expand = c(0, 0)) +  # Remover espaço extra no eixo X
  scale_color_manual(values = module_color_map) +  # Aplicar o mapeamento de
cores personalizado
  facet_grid(rows = vars(module)) +  # Facetar o gráfico por módulo
  labs(
    x = "Tissue (Ordered by Life Stage)",
    y = "Normalized Expression",
    title = "Normalized Expression by Tissue and Module (Ordered by Life Stage)"
  )

dev.off()
# Gráfico para mediada expressão normalizada

# Calcular a média da expressão normalizada por tecido e módulo
avg_expression_df <- submod_df %>%
  group_by(Tissue, module) %>%
  summarize(MeanExpression = mean(Expression, na.rm = TRUE), .groups = "drop")

# Criar o gráfico png("mean_expression_with_bottom_space.png", width = 14,
height = 10, units = "in", res = 300)

avg_expression_df %>%
  ggplot(aes(x = Tissue, y = MeanExpression, group = module)) +
  geom_line(aes(color = module), size = 0.7) +  # Linhas representando a média
da expressão normalizada
  geom_point(aes(color = module), size = 2) +  # Pontos para destacar valores
médios
  theme_bw() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10, margin =
margin(t = 5)), # Ajustar rotação e tamanho do texto
    axis.title.x = element_text(margin = margin(t = 10)), # Espaçamento para o
eixo X
    axis.title.y = element_text(margin = margin(r = 10)), # Espaçamento para o
eixo Y
    strip.text = element_text(size = 12, face = "bold"), # Títulos das facetas
em negrito e maiores
    plot.margin = margin(t = 30, r = 10, b = 20, l = 10), # Ajustar borda
inferior
    panel.spacing = unit(1.2, "lines") # Aumentar o espaçamento entre facetas
  ) +
  scale_x_discrete(expand = expansion(mult = c(0.05, 0.05))) +  # Espaçamento
extra no eixo X
```

```
  scale_y_continuous(expand = expansion(mult = c(0.1, 0.1))) +  # Espaçamento
extra no eixo Y (inferior e superior)
  scale_color_manual(values = module_color_map) +  # Aplicar o mapeamento de
cores personalizado
  facet_grid(rows = vars(module)) +  # Facetar o gráfico por módulo
  labs(
    x = "Tissue (Ordered by Life Stage)",
    y = "Mean Normalized Expression",
    title = "Mean Normalized Expression by Tissue and Module (Ordered by Life
Stage)"
  )

dev.off()
```