



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE  
INSTITUTO METRÓPOLE DIGITAL  
PROGRAMA DE PÓS-GRADUAÇÃO EM BIOINFORMÁTICA

JOÃO VITOR FERREIRA CAVALCANTE

**O Ecossistema Computacional da Metagenômica: Fluxos de Trabalho e  
Reprodutibilidade**

NATAL - RN  
2024

JOÃO VITOR FERREIRA CAVALCANTE

**O Ecossistema Computacional da Metagenômica: Fluxos de Trabalho e  
Reprodutibilidade**

Defesa de Mestrado apresentada ao Programa de  
Pós-Graduação em Bioinformática da Universidade  
Federal do Rio Grande do Norte.

**Área de Concentração:** Bioinformática  
**Linha de Pesquisa:** Biologia de Sistemas  
**Orientador:** Rodrigo Juliani Siqueira Dalmolin

NATAL - RN  
2024

JOÃO VITOR FERREIRA CAVALCANTE

**O Ecossistema Computacional da Metagenômica: Fluxos de Trabalho e  
Reprodutibilidade**

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Bioinformática da Universidade Federal do Rio Grande do Norte

**Área de Concentração:** Bioinformática

**Linha de Pesquisa:** Biologia de Sistemas

**Orientador:** Rodrigo Juliani Siqueira Dalmolin

Natal, 06 de Dezembro de 2024.

**BANCA EXAMINADORA**

---

Prof. Dr. Rodrigo Juliani Siqueira Dalmolin  
Universidade Federal do Rio Grande do Norte  
(Presidente)

---

Prof. Dr. Renan Cipriano Moiolli  
Universidade Federal do Rio Grande do Norte  
(Examinador Interno do Programa)

---

Prof. Dr. Daniel Carlos Ferreira Lanza  
Universidade Federal do Rio Grande do Norte  
(Examinador Interno do Programa)

---

Prof. Dr. Marcel da Câmara Ribeiro-Dantas  
Universidade Potiguar  
(Examinador Externo à Instituição)

Para meus pais, que me mostraram que isso era  
possível.

## **AGRADECIMENTOS**

Agradeço a meus amigos do BioME, colegas na jornada acadêmica, que me forneceram várias discussões, sobre pesquisa ou não, várias descontrações - a tradição do 4 e mate - e várias (necessárias) distrações, além de muitas e muitas perguntas. Mais que tudo isso, vocês me deram o sentimento de fazer parte de uma comunidade. Agradeço a todos vocês, mas em especial a Julia, ao Diego, ao Epitácio, ao Gleison, ao Bruno e a Bianca.

Agradeço ao Rodrigo, por me dar tanta liberdade para executar meu trabalho, pela confiança constante no meu futuro e no caminho que estou trilhando e pelos conselhos precisos quando este caminho deu algumas voltas caóticas.

Agradeço aos amigos do grupo Código Bonito, em especial ao Tiago, ao Kléber e ao Vini. Obrigado por confiar no meu trabalho o suficiente para me recomendar para vagas e eventos, que fizeram uma diferença inestimável na minha trajetória e me proporcionaram com novas experiências, amizades e perspectivas.

Agradeço aos amigos feitos nesses vários eventos, em especial Débora e Pedro.

Agradeço às instituições brasileiras que me possibilitaram fazer pesquisa. Agradeço sobretudo aos inúmeros trabalhadores nessas instituições que as fazem funcionar, apesar das limitações financeiras e estruturais do Brasil.

Agradeço a Beatriz, que sabe que seu impacto é tanto que apenas algumas linhas em um documento acadêmico seria insuficiente ou talvez até desrespeitoso, e que também sabe que já te agradeço tanto em tudo que faço a ponto de ficar repetitivo. Te amo. Obrigado por estar aqui para mim, me mostrando a importância de ser feliz.

Por fim, agradeço em dobro a todos que aqui não foram nomeados, mas que, de uma forma ou de outra, tiveram um impacto tangível na minha vida. Peço apenas que possam perdoar esse terrível lapso.

*"Nature uses only the longest threads to weave her patterns, so that each small piece of her fabric  
reveals the organization of the entire tapestry."  
(Richard P. Feynman)*

## RESUMO

Os últimos anos têm testemunhado avanços significativos no estudo de comunidades microbianas complexas, impulsionados pela evolução das tecnologias de sequenciamento e pela crescente adoção de métodos de sequenciamento total do genoma (whole genome shotgun) em detrimento dos métodos, antes mais tradicionais, baseados em amplicon. Com essa evolução, essas abordagens foram desenvolvidas com estratégias computacionais associadas para lidar com os dados que geram. No entanto, esses métodos computacionais geralmente não foram acompanhados por estratégias de design cuidadosas que priorizam o suporte a longo prazo, com baixa necessidade de manutenção, alta acessibilidade de dados e automação de ponta a ponta.

Neste trabalho, nosso objetivo é, primeiramente, elaborar sobre o cenário computacional em metagenômica e como os métodos atuais podem negligenciar princípios fundamentais de desenvolvimento de software, que os orientariam para uma maior reprodutibilidade, tais como isolamento de dependências, alta parametrização, geração automática de relatórios com figuras interativas que facilitam a exploração de dados e, por fim, documentação descritiva e prática. Em seguida, abordamos as limitações atuais no processamento de dados metagenômicos ao implementar um novo pipeline de análise de dados, o EURYALE, baseado em uma metodologia anterior, o MEDUSA, que selecionou suas ferramentas por meio de rigoroso benchmarking. Esse novo pipeline, adaptável a diferentes cenários e construído com boas práticas de desenvolvimento de software como princípios norteadores, visa avançar o processamento de dados metagenômicos como um todo e, adicionalmente, tornar os dados resultantes desses pipelines de análise acessíveis a um público mais amplo.

**Palavras-chave:** Metagenômica. Fluxos de Trabalho. Reprodutibilidade. Desenvolvimento de Software. Nextflow.

## ABSTRACT

The past few years have seen significant advances in the study of complex microbial communities associated with the evolution of sequencing technologies and increasing adoption of whole genome shotgun sequencing methods over the once more traditional Amplicon-based methods. Through this evolution, these approaches have been built with associated computational strategies to tackle the data they generate. However, these computational methods have not been generally accompanied by thoughtful design strategies that prioritise long term support with low maintainability, high data accessibility and end-to-end automation.

In this work, we aim to first elaborate on the computational landscape in metagenomics, and how its methods can disregard fundamental software development principles that guide them towards improved reproducibility, principles such as dependency isolation, high parameterization, automatic report generation, with interactive figures that facilitate data exploration and, finally, descriptive and practical documentation. Following this, we tackle current limitations in metagenomic data processing by implementing a new data analysis pipeline, EURYALE, based on a previous methodology, MEDUSA, that selected its tools through rigorous benchmarking. This new pipeline, adaptable to different scenarios and built with good software development practices as guiding principles, serves to advance metagenomic data processing as a whole, and, additionally, make the data resulting from these analysis pipelines accessible to a wider audience.

**Keywords:** Metagenomics. Pipeline. Reproducibility. Software Development. Nextflow.



## LISTA DE FIGURAS

Figura 1 – Diagrama que ilustra a estrutura geral de um fluxo de trabalho, compondo ferramentas que realizam diferentes etapas da análise . . . .	12
Figura 2 – Exemplo do cabeçalho de um relatório gerado através da ferramenta MultiQC, implementada como parte integrante do EURYALE. . . . .	29
Figura 3 – Exemplo do cabeçalho de um relatório gerado através da ferramenta Microview, escrita em Python e implementada como parte integrante do EURYALE. . . . .	29
Figura 4 – Diagrama ilustrando como a entrada de download do EURYALE adquire os bancos de dados que alimentam a entrada principal. Com uma análise típica do zero sendo constituída por ambas etapas. . . .	30
Figura 5 – Porção inicial da interface gráfica do EURYALE na Seqera Platform (< <a href="https://cloud.seqera.io/">https://cloud.seqera.io/</a> >). Através desse formulário usuários podem selecionar os parâmetros a serem utilizados para a execução do fluxo de trabalho. . . . .	31

## LISTA DE ABREVIATURAS E SIGLAS

16S	sequenciamento da subunidade ribossomal 16S de genomas bacterianos
ASV	variantes de sequência amplicon - do inglês <i>amplicon sequence variant</i>
DNA	Ácido Desoxirribonucléico
MS	metagenômica <i>shotgun</i>
OTU	unidades taxonômicas operacionais - do inglês <i>operational taxonomic units</i>

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>11</b>
1.1	VISÃO GERAL - METAGENÔMICA . . . . .	11
1.2	DESENVOLVIMENTO DE SOFTWARE CIENTÍFICO . . . . .	12
1.3	O ECOSSISTEMA COMPUTACIONAL EM METAGENÔMICA . . . . .	13
<b>2</b>	<b>OBJETIVOS . . . . .</b>	<b>15</b>
2.1	GERAL . . . . .	15
2.2	ESPECÍFICOS . . . . .	15
	<b>CAPÍTULO 1 . . . . .</b>	<b>16</b>
	<b>CAPÍTULO 2 . . . . .</b>	<b>20</b>
<b>3</b>	<b>DISCUSSÃO . . . . .</b>	<b>28</b>
<b>4</b>	<b>CONCLUSÃO . . . . .</b>	<b>32</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>33</b>
<b>5</b>	<b>ANEXO I . . . . .</b>	<b>36</b>

## 1 INTRODUÇÃO

### 1.1 VISÃO GERAL - METAGENÔMICA

A história da vida microscópica, ou microbiana, no planeta Terra supera a história da vida macroscópica por milhares de anos (Magnabosco et al., 2024). A metagenômica surge como uma abordagem que possibilita descobertas acerca da vida microbiana através do sequenciamento genético. Avanços no que viria eventualmente a se tornar a metagenômica surgem ainda nos anos 90, com o primeiro sequenciamento de genoma completo de um organismo de vida livre, a bactéria *Haemophilus influenza* (Wooley; Godzik; Friedberg, 2010). Esse ponto na história científica marca o primeiro uso bem sucedido do que vem a ser chamado de *whole-genome shotgun*, ou sequenciamento de genoma completo, no qual a amostra possui seu conteúdo genético fragmentado em *reads*, ou leituras, que são então sequenciadas. Essa técnica viria a ser refinada e aplicada para amostras ambientais, seja este ambiente uma amostra de solo florestal ou uma biópsia intestinal. Nessa nova técnica se buscou sequenciar o conteúdo genético que compreenda os diferentes microorganismos presentes em tais amostras, originando assim o que será aqui descrito como metagenômica *shotgun* (MS).

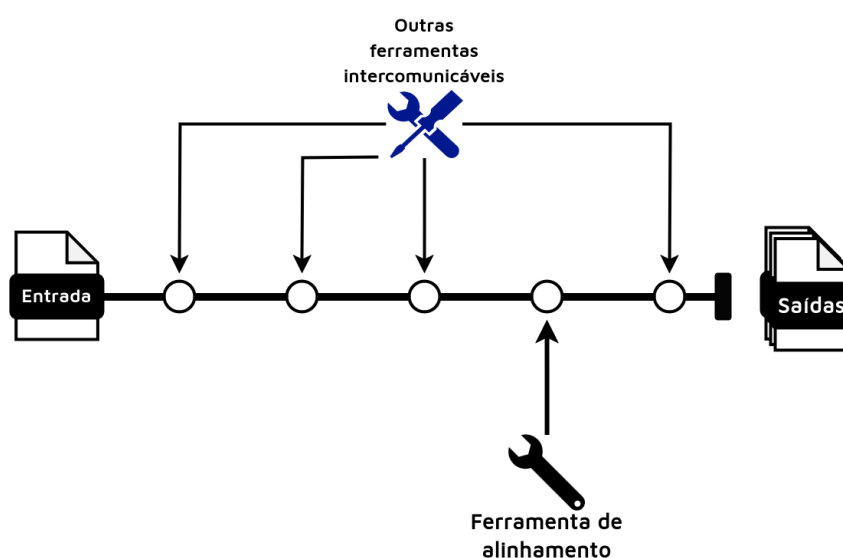
No entanto, o estudo de comunidades microbianas se populariza de fato com uma técnica que não busca capturar o conteúdo genético total de uma amostra, mas apenas uma subregião de seu Ácido Desoxirribonucleico (DNA) ribossomal que possua ao mesmo tempo regiões conservadas, capaz de serem passíveis de anelamento por *primers*, e regiões hipervariáveis, capazes de distinguir um microorganismo de outro. Em bactérias o ribotipo selecionado foi o DNA que codifica a subunidade 16S, que é amplificada e então sequenciada. O sequenciamento da subunidade ribossomal 16S de genomas bacterianos (16S), também denominado metataxonômica (Marchesi; Ravel, 2015), possibilitou uma maneira simples, e pouco computacionalmente intensiva quando comparada à MS (Tremblay; Schreiber; Greer, 2022), para realizar identificação de táxons bacterianos em uma amostra ambiental. Ademais, técnicas computacionais posteriores ultimamente facilitariam a conexão de informação funcional às abundâncias taxonômicas obtidas através desta técnica.

Dessa maneira, possuímos atualmente, duas possíveis abordagens para se estudar comunidades microbianas, a MS e o 16S. Essas abordagens, sobretudo a MS, geram um alto volume de dados, e portanto, há a necessidade do desenvolvimento de ferramentas computacionais que processem esses dados e gerem informação científica que descreva de forma acurada e reproduzível achados acerca do microambiente do estudo (Comin et al., 2021).

## 1.2 DESENVOLVIMENTO DE SOFTWARE CIENTÍFICO

Metodologias computacionais constituem parte indissociável da biologia molecular moderna, com novas ferramentas, abordagens e fluxos de trabalho, isto é, metodologias que agregam diversas ferramentas em sequência (Figura 1), surgindo a cada momento. Nesse contexto, muito tem se discutido acerca da qualidade do software desenvolvido, não apenas do ponto de vista de qualidade científica ou estatística da análise a ser realizada, mas também qualidade a partir de um ponto de vista técnico.

Figura 1 – Diagrama que ilustra a estrutura geral de um fluxo de trabalho, compondo ferramentas que realizam diferentes etapas da análise



Tipicamente software é tratado como um ponto adjacente à uma análise científica, e não seu principal produto. Parcialmente isto se dá devido ao modelo de desenvolvimento de software científico atualmente vigente, que se dá por meio de projetos de doutorado e mestrado, dificultando, dessa maneira, manutenção a longo prazo (Altschul et al., 2013) (Mangul; Martin et al., 2019). Portanto, produzir metodologias que sejam usáveis a longo prazo sem necessitarem de alta manutenção é essencial. Nesse sentido, a utilização de tecnologias que facilitem a instalação de software, como gerenciadores de ambiente, ou tecnologias capazes de encapsular um ambiente computacional, como contêineres, são indispensáveis para a garantia de reprodutibilidade de qualquer método computacional, por garantirem o isolamento das dependências de um *software* indefinidamente (Kadri et al., 2022).

No entanto, garantir a capacidade de instalação do software é um ponto basal para determinar a qualidade de código científico. Outros princípios, aqui denominados como boas práticas de desenvolvimento de software científicos, são igualmente indispensáveis para que uma metodologia seja utilizável a longo prazo, além de capaz de gerar resultados científicos interpretáveis e consistentes. Além da instalabilidade,

alguns outros princípios que garantem a qualidade de um software a longo prazo são uma documentação descritiva, com exemplos práticos de utilização, suporte multi-plataforma (Mangul; Mosqueiro et al., 2019) e, para softwares de processamento de dados, relatórios interativos ou logs acessíveis (Perkel, 2018). A implementação de tais princípios em um *software* científico não apenas aumenta sua usabilidade, mas também aumenta as citações de seus artigos, aumentando, consequentemente, o alcance dessas metodologias (Mangul; Martin et al., 2019).

### 1.3 O ECOSSISTEMA COMPUTACIONAL EM METAGENÔMICA

As metodologias de processamento de dados 16S em grande parte já estão estabelecidas, dada a idade mais avançada da abordagem. Nesse sentido, vemos que o cerne das abordagens trata de atribuir identificadores únicos às sequências 16S obtidas, caracterizando assim táxons distintos. Esses identificadores podem ser atribuídos através de métodos de agrupamento, caracterizando as abordagens baseadas em unidades taxonômicas operacionais - do inglês *operational taxonomic units* (OTU), que agrupam sequências com pelo menos 97% de identidade em grupos biológicos distintos, ou abordagens baseadas em variantes de sequência amplicon - do inglês *amplicon sequence variant* (ASV), que, através de modelos estatísticos, tentam definir variações biológicas reais na sequência - contrastadas com variações devido a erros de sequenciamento, e dessa maneira obter uma resolução taxonômica maior comparada às baseadas em OTU (Chiarello et al., 2022). Nesse contexto, observamos metodologias que buscam, a partir dos OTU ou ASV identificados, inferir abundâncias de vias metabólicas específicas, tirando proveito de bancos de dados de informação curada a respeito desses táxons que possua mapeamento das famílias gênicas de seu genoma a funções biológicas (Douglas et al., 2020). Quanto a fluxos de trabalho para processamento de dados 16S, destaca-se o *nf-core/ampliseq* (Straub et al., 2020), que implementa várias das boas práticas de software citadas anteriormente, como suporte multi-plataforma, documentação descritiva e exemplificada e relatórios automáticos interativos.

Por outro lado, o campo do desenvolvimento de metodologias para dados de MS ainda é bastante fértil, com novas técnicas computacionais desenvolvidas rotineiramente (Liu et al., 2021). De forma geral, podemos agrupar as metodologias em duas grandes categorias: Metodologias livres de montagem, isto é, aquelas que se utilizam apenas da informação contida nas leituras para obter seus resultados, e metodologias baseadas em montagem, que primeiro realizam a montagem de leituras em sequências contíguas (ou *contigs*), que fornecerá então a base para o processamento seguinte (Breitwieser; Lu; Salzberg, 2019). Apesar da capacidade que métodos baseados em montagem tem de descobrir novos organismos e montar genomas inéditos, métodos livres de montagem apresentam certas vantagens, sobretudo quando consideramos

dados com baixa cobertura de sequência, o que pode resultar em montagens pouco precisas (Ayling; Clark; Leggett, 2020).

No que se diz respeito a essas metodologias, vemos que as baseadas em montagem são amplas e cobrem os mais diversos aspectos do processamento de dados de MS, com exemplos como *nf-core/mag* (Krakau et al., 2022) e *metaphor* (Salazar et al., 2023). No entanto, quando observamos métodos livres de montagem, vemos um cenário mais escasso, sobretudo quando consideramos apenas fluxos de trabalho, ou *pipelines*, orquestrados por linguagens de gerenciamento de metodologias científicas, como Nextflow (Di Tommaso et al., 2017) ou Snakemake (Mölder et al., 2021). No contexto de métodos livres de montagem para dados MS, vale ressaltar o fluxo de trabalho MEDUSA (Morais et al., 2022), que apresentou boa sensibilidade e flexibilidade para análises de classificação taxonômica e anotação funcional.

Nesse sentido, há a necessidade de desenvolver uma metodologia para dados de MS que siga boas práticas de desenvolvimento de software científico e que tenha como princípios norteadores a reprodutibilidade, documentação descritiva e prática e interpretabilidade. Este último ponto que se torna especialmente relevante ao considerarmos a complexidade e a alta dimensionalidade desses dados.

## **2 OBJETIVOS**

### **2.1 GERAL**

Obter uma visão geral do ecossistema computacional em metagenômica atual e como ele se associa com princípios de desenvolvimento de software científico, desenvolvendo então uma metodologia para dados de MS que possibilite uma análise metagenômica compreensiva, de forma reprodutível e flexível.

### **2.2 ESPECÍFICOS**

- Avaliar o atual ferramentário computacional para dados de metagenômica e sua adesão a princípios de desenvolvimento de software sustentável.
- Desenvolver uma metodologia robusta, flexível e acessível para análise de dados de MS.



## **CAPÍTULO 1**

### **Artigo: Bridging the Gaps in Meta-Omic Analysis: Workflows and Reproducibility**

Escrito por: João Vitor Ferreira Cavalcante, Iara Dantas de Souza, Diego Arthur de Azevedo Moraes e Rodrigo Juliani Siqueira Dalmolin

*Artigo publicado no periódico OMICS: A Journal of Integrative Biology*

Open camera or QR reader and  
scan code to access this article  
and other resources online.



## Bridging the Gaps in Meta-Omic Analysis: Workflows and Reproducibility

João Vitor Ferreira Cavalcante,<sup>1</sup> Iara Dantas de Souza,<sup>1</sup> Diego Arthur de Azevedo Morais,<sup>1</sup>  
and Rodrigo Juliani Siqueira Dalmolin<sup>1,2</sup>

### Abstract

The past few years have seen significant advances in the study of complex microbial communities associated with the evolution of sequencing technologies and increasing adoption of whole genome shotgun sequencing methods over the once more traditional Amplicon-based methods. Although these advances have broadened the horizon of meta-omic analyses in planetary health, human health, and ecology from simple sample composition studies to comprehensive taxonomic and metabolic profiles, there are still significant challenges in processing these data. First, there is a widespread lack of standardization in data processing, including software choices and the ease of installing and running attendant software. This can lead to several inconsistencies, making comparing results across studies and reproducing original results difficult. We argue that these drawbacks are especially evident in metatranscriptomic analysis, with most analyses relying on *ad hoc* scripts instead of pipelines implemented in workflow managers. Additional challenges rely on integrating meta-omic data, since methods have to consider the biases in the library preparation and sequencing methods and the technical noise that can arise from it. Here, we critically discuss the current limitations in metagenomics and metatranscriptomics methods with a view to catalyze future innovations in the field of Planetary Health, ecology, and allied fields of life sciences. We highlight possible solutions for these constraints to bring about more standardization, with ease of installation, high performance, and reproducibility as guiding principles.

**Keywords:** metagenomics, metatranscriptomics, pipelines, reproducibility, sustainable software, data integration

### Perspective

SIGNIFICANT ADVANCES HAVE BEEN MADE IN microbiology and the study of microbial communities over the past decade. One notable breakthrough is amplicon sequencing, which allows scientists to study the taxonomic composition of an environmental sample. The nascent area of metagenomics was then significantly broadened by the development of whole-metagenome shotgun (WMS) sequencing, which enables the investigation of the full genetic content of samples. This allowed the analysis of functional pathways (Franzosa et al., 2018) as well as the discovery of new microorganisms through metagenome-assembled genomes (Breitwieser et al., 2019). Metagenomics has also contributed to the rise of new fields such as Planetary Health and One

Health that view human and ecosystem health intertwined and interdependent, proving to be an impactful approach for integrative areas of study. On the other hand, although WMS data processing software is now widely adopted in the scientific community, there are still multiple challenges in analyzing these data.

Installability, ease of use, and portability among different systems are central challenges in metagenomics and metatranscriptomics software. These difficulties can be mitigated through the use of different methods to improve software, such as code packaging and container technology, as well as toolset curation and workflow management software.

The intrinsic nature of computational science favors reproducibility in the execution of an analysis, as each step can be programmed or automated. However, this usually does

<sup>1</sup>Bioinformatics Multidisciplinary Environment—IMD, Federal University of Rio Grande do Norte, Natal, Brazil.

<sup>2</sup>Department of Biochemistry—CB, Federal University of Rio Grande do Norte, Natal, Brazil.

not happen in practice. This is due to difficulties in software installation, execution, and documentation (Piccolo and Frampton, 2016). Bioinformatics software can be packaged in a multitude of ways, and even though there is no standard on how to best package and share your software, there are some principles that can be easily followed.

For example, prioritizing using a single software version throughout the analysis can boost reproducibility (Nüst et al., 2020). This can be enforced through the use of containerization software, like Docker or Singularity, and there already are plenty of initiatives seeking to standardize bioinformatics software packaging, like BioConda (<https://bioconda.github.io/>) and BioContainers (<https://biocontainers.pro/>). These technologies may significantly improve the installability and archival stability of software, leading to an increase in citation rates (Mangul et al., 2019). There is an enhancement in the reproducibility of an analysis and a potential increase in the research impact of it by packaging software and distributing their specific versions with technologies such as Conda, Docker, and Singularity.

In our opinion, no evaluation of metagenomics software has directly ascertained the adoption of software packaging and container technology, although there have been ease-of-use evaluations for metagenomics software (Lindgreen et al., 2016). This could point to a low adoption of these practices in the metagenomics community.

Lindgreen et al. (2016) have also pointed out that toolset choice has a significant impact in metagenomic analysis, not only in terms of computational performance but also in terms of accuracy. Therefore, toolset curation has become common among bioinformaticians, spawning many different scientific workflows for metagenomics data analysis, such as nf-core/mag (Krakau et al., 2022) and MEDUSA (Morais et al., 2022). Moreover, many workflow management software have also been created and adopted by the metagenomics community, primarily Snakemake, Nextflow, and Galaxy.

These tools can greatly enhance reproducibility and ease of use by mitigating platform-specific inaccuracies through their support for isolated task execution. Furthermore, they facilitate the integration of disjointed pieces of code, offering a comprehensive view of the entire analysis pipeline (Wratte et al., 2021). They also provide modularity, allowing users to choose different steps for their analysis. Workflow management software provides significant advantages for

metagenomics data processing. However, there is limited adoption of these technologies among bioinformaticians, even among those already involved in toolset curation.

While the limitations discussed here are common to both metagenomics and metatranscriptomics, or, rather, to bioinformatics as a whole, they become more evident with metatranscriptomic methods. Metatranscriptomics helps to provide a clearer understanding of the functional environment in a sample, in a way that is not easily done with metagenomics. For example, metatranscriptomics can show active microbes in a community and which metabolic pathways are most prevalent (Bashiardes et al., 2016), as well as elucidate pathogen–host interactions (Moniruzzaman et al., 2017). Despite its biological potential, there are few performance and accuracy benchmarks for metatranscriptomics pipelines (Shakya et al., 2019).

Additionally, we suggest that many metatranscriptomics pipelines [see Shakya et al. (2019) for an overview] do not adhere to the sustainable software principles mentioned in our present analysis (Table 1). We verified the source code repository and documentation for these pipelines and argue that the application of software packaging, container technology, and workflow management software, practices that enhance the reproducibility and long-term support of these pipelines, are still limited among these tools. We also acknowledge that there are other relevant criteria for their current implementation, such as a focus on web-based analysis, that was not within the scope of this comparison. Furthermore, improvements are evident in recent pipelines, which better tackle the reproducibility issue by implementing container technology such as Docker and Singularity (Taj et al., 2023). Nonetheless, the low adherence to software sustainability practices still shows the necessity for specific and sustainable methods for these types of data.

Methodologies for integrating multiomic data are still few and far between, and sometimes still rely on disconnected scripts without workflow management or containerization. Moreover, benchmarks of omic integration tools are usually restricted to a specific biological question of interest, and, therefore, not generalizable (Subramanian et al., 2020).

Still, there have been notable advances in multiomic data integration, particularly in the Galaxy community, with the development of three integrative meta-omic pipelines that, coupled with a web application, provide an end-to-end

TABLE 1. SELECTED METATRANSCRIPTOMICS PIPELINES AND THEIR COMPARISON IN RELATION TO THE SUSTAINABLE SOFTWARE PRINCIPLES MENTIONED IN THIS PAPER

<i>Pipeline</i>	<i>Software packaging or container technology</i>	<i>Workflow management</i>	<i>Notes</i>
MetaTrans	N/A	N/A	Website could not be accessed. <a href="http://www.metatrans.org">http://www.metatrans.org</a>
COMAN	N/A	N/A	Web server based.
FMAP	No	No	<a href="https://github.com/jiwoongbio/FMAP">https://github.com/jiwoongbio/FMAP</a>
SAMSA2	No	No	<a href="https://github.com/transcript/samsa2">https://github.com/transcript/samsa2</a>
HUMANn2	Yes (Conda)	No	<a href="https://github.com/biobakery/humann">https://github.com/biobakery/humann</a>
SqueezeMeta	Yes (Conda)	No	<a href="https://github.com/jtamames/SqueezeMeta">https://github.com/jtamames/SqueezeMeta</a>
IMP	Yes (Docker)	Yes (Snakemake)	<a href="https://git-r3lab.uni.lu/IMP/IMP">https://git-r3lab.uni.lu/IMP/IMP</a>
MOSCA	Yes (Conda)	Yes (Snakemake)	<a href="https://github.com/iquasere/MOSCA">https://github.com/iquasere/MOSCA</a>

Each of these pipelines was checked for applications of software packaging, container technology, and use of workflow management software. Access date: October 30, 2023.

N/A, not applicable.

analysis of meta-omic data (Schiml et al., 2023). This could motivate similar developments in other workflow orchestration software communities.

As with most software for bioinformatics, software for analyzing metagenomics and metatranscriptomics data need to be developed with clarity, reproducibility, and reusability as guiding principles. This is further supported when we look at the current metatranscriptomics and data integration ecosystems, these that are still nascent approaches to microbiome studies and therefore still show low adherence to these practices. A wider adoption of software sustainability guidelines in metagenomics and metatranscriptomics methods ensures future research and scientists of the high quality of these methods and their ability to stand the test of time, and it also paves the way for meta-omics to be an indispensable approach to various areas of study, such as ecology and Planetary Health.

### Authors' Contributions

J.V.F.C. wrote the original draft and edited it. R.J.S.D. and I.D.d.S. contributed to the discussion of the topic. R.J.S.D., J.V.F.C., and D.A.d.A.M. conceived the original idea. All authors contributed to the drafts and approved the final manuscript.

### Author Disclosure Statement

The authors declare there are no conflicting financial interests.

### Funding Information

This work was supported by the governmental Brazilian agencies CAPES, grant 88887.834652/2023-00, and CNPq, grant 312305/2021-4.

### References

- Bashiardes S, Zilberman-Schapira G, Elinav E. Use of metatranscriptomics in microbiome research. *Bioinform Biol Insights* 2016;10:BBI.S34610; doi: 10.4137/BBI.S34610.
- Breitwieser FP, Lu J, Salzberg SL. A review of methods and databases for metagenomic classification and assembly. *Brief Bioinform* 2019;20(4):1125–1136; doi: 10.1093/bib/bbx120
- Franzosa EA, McIver LJ, Rahnavard G, et al. Species-level functional profiling of metagenomes and metatranscriptomes. *Nat Methods* 2018;15(11):962–968; doi: 10.1038/s41592-018-0176-y
- Krakau S, Straub D, Gourel H, et al. Nf-Core/Mag: A best-practice pipeline for metagenome hybrid assembly and binning. *NAR Genom Bioinform* 2022;4(1):lqac007; doi: 10.1093/nargab/lqac007
- Lindgreen S, Adair KL, Gardner PP. An evaluation of the accuracy and speed of metagenome analysis tools. *Sci Rep* 2016;6(1):19233; doi: 10.1038/srep19233

- Mangul S, Mosqueiro T, Abdill RJ, et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. *PLoS Biol* 2019; 17(6):e3000333; doi: 10.1371/journal.pbio.3000333
- Moniruzzaman M, Wurch LL, Alexander H, et al. Virus-host relationships of marine single-celled eukaryotes resolved from metatranscriptomics. *Nat Commun* 2017;8:16054; doi: 10.1038/ncomms16054
- Morais DAA, Cavalcante JVF, Monteiro SS, et al. MEDUSA: A pipeline for sensitive taxonomic classification and flexible functional annotation of metagenomic shotgun sequences. *Front Genet* 2022;13.
- Nüst D, Sochat V, Marwick B, et al. Ten simple rules for writing dockerfiles for reproducible data science. *PLoS Comput Biol* 2020;16(11):e1008316; doi: 10.1371/journal.pcbi.1008316
- Piccolo SR, Frampton MB. Tools and techniques for computational reproducibility. *GigaScience* 2016;5(1):30; doi: 10.1186/s13742-016-0135-4
- Schiml VC, Delogu F, Kumar P, et al. Integrative meta-omics in galaxy and beyond. *Environ Microbiome* 2023;18(1):56; doi: 10.1186/s40793-023-00514-9.
- Shakya M, Lo C-C, Chain PSG. Advances and challenges in metatranscriptomic analysis. *Front Genet* 2019;10:904.
- Subramanian I, Verma S, Kumar S, et al. Multi-omics data integration, interpretation, and its application. *Bioinform Biol Insights* 2020;14:1177932219899051; doi: 10.1177/1177932219899051
- Taj B, Adeolu M, Xiong X, et al. MetaPro: A scalable and reproducible data processing and analysis pipeline for metatranscriptomic investigation of microbial communities. *Microbiome* 2023;11(1):143; doi: 10.1186/s40168-023-01562-6
- Wratten L, Wilm A, Göke J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nat Methods* 2021;18(10):1161–1168; doi: 10.1038/s41592-021-01254-9

Address correspondence to:

Rodrigo Juliani Siqueira Dalmolin, PhD  
*Bioinformatics Multidisciplinary Environment*  
 Rua do Horto  
 Lagoa Nova  
 Natal 59076-550  
 Brazil

E-mails: rodrigo.dalmolin@imd.ufm.br;  
 dalmolin\_r@yahoo.com.br

### Abbreviation Used

WMS = whole-metagenome shotgun

## **CAPÍTULO 2**

**Artigo: EURYALE: A versatile Nextflow pipeline for taxonomic classification and functional annotation of metagenomics data**

Escrito por: João Vitor Ferreira Cavalcante, Iara Dantas de Souza, Diego Arthur de Azevedo Moraes e Rodrigo Juliani Siqueira Dalmolin

*Artigo submetido ao periódico IEEE Access*

# EURYALE: A versatile Nextflow pipeline for taxonomic classification and functional annotation of metagenomics data

João Vitor F. Cavalcante

*Bioinformatics Multidisciplinary Environment  
Federal University of Rio Grande do Norte  
Natal, Brazil  
0000-0001-7513-7376*

Diego A. A. Morais

*Bioinformatics Multidisciplinary Environment  
Federal University of Rio Grande do Norte  
Natal, Brazil  
0000-0002-7357-3446*

Iara Dantas de Souza

*Bioinformatics Multidisciplinary Environment  
Federal University of Rio Grande do Norte  
Natal, Brazil  
0000-0002-2550-6150*

Rodrigo J. S. Dalmolin

*Department of Biochemistry  
Federal University of Rio Grande do Norte  
Natal, Brazil  
0000-0002-1688-6155*

**Abstract**—EURYALE is a Nextflow pipeline designed for the sensitive taxonomic classification and flexible functional annotation of metagenomic shotgun sequences. It provides a comprehensive solution for preprocessing, assembly, alignment, taxonomic classification, and functional annotation of metagenomic data. EURYALE builds upon the Snakemake-based pipeline MEDUSA. EURYALE inherits the tools present in MEDUSA, selected based on rigorous benchmarks for performance, accuracy, and sensitivity. The new pipeline has been developed with the nf-core pipeline template, which focuses on modularity, allowing a high degree of parameterization. EURYALE provides easier resource management, enforcing strict memory and CPU requirements based on its Nextflow configuration. It has also become more versatile, as it can be executed using Docker and Singularity, which further extends its usability across various platforms. It can also natively take advantage of computational infrastructures such as SLURM and Amazon Web Services. EURYALE inherits the sensitivity in taxonomic classification and flexibility in functional annotation of its predecessor, combined with improved versatility.

**Index Terms**—Metagenomic Analysis; Nextflow; Taxonomic Classification; Functional Annotation; Pipeline

## I. INTRODUCTION

The growth of metagenomics in the last few years as the main approach to study complex microbial communities is apparent, with many different methodologies arising to process whole-metagenome shotgun (WMS) data. Metagenomics consists of an approach to sequence the genetic content of an environmental sample, be that environment a host-associated tissue, like the human gut, be it something like a soil sample.

Metagenomics has typically been restricted, in the past, to amplicon sequencing, an approach nowadays referred to as ‘Metataxonomics’ [1], which allows researchers to investigate the taxonomic composition of a sample by sequencing specific genomic regions, such as the 16S region in bacteria. WMS, on

the other hand, provides not only these specific regions, but a more representative genetic content of a sample, allowing for bacterial assembly, as well as investigations of functional information.

WMS data, due to its size and complexity, presents a series of challenges in regards to its processing. The main challenges that methodologies aim to tackle are: Performance issues and data storage, as metagenomic datasets can be large and unwieldy [2], which can be improved by making metagenomic pipelines usable in cloud environments [3]; Sequence contamination from other sources [4]; And useful and easy-to-parse results, given the inherent complexity of these data [5].

The methods to process WMS data can be broadly divided into two: Assembly-free methods, i.e., those that rely on direct read classification; and assembly-based methods, i.e., methods that first perform read assembly into contigs prior to classification and annotation [6].

Assembly-free methods provide an advantage to assembly-based methods particularly regarding performance, but advantages have also been noted in using these methods with low coverage data, where assembly-based ones can often lead to inaccurate results [7]. Additionally, assembly-free methods have been used to study the biodiversity in marine environments [8] as well as compositional differences in major depressive disorder [9], showing its potential to empower scientific discovery.

Among metagenomics methods, most have organized themselves into pipelines orchestrated by workflow managers, such as Nextflow [10] and Snakemake [11], which increase modularization, parameterization, portability and ease of installation for these softwares [12][13]. Although pipelines covering most of the metagenomic analysis process through assembly-

based strategies, such as nf-core/mag [14], Metaphor [15] and MGnify [16], are widespread, assembly-free methods are under explored and pipelines are few and far between.

One highlight among assembly-free methods is the MEDUSA<sup>1</sup> pipeline [18], which was built based on careful benchmarking of multiple WMS data analysis tools, which were finally brought together in a pipeline orchestrated by Snakemake. Although MEDUSA proved to be a great advance in this field, there are some implementation details that could be improved. First, portability was limited: MEDUSA could only run through a BioConda-based environment [19], which is error-prone and less stable in the long term than a Docker or Singularity image [20]. Secondly, MEDUSA’s DSL of choice, Snakemake, has proven hard to work with and maintain, which is something that others in the field of workflow development have observed [21][22][20], with most preferring Nextflow. Lastly, MEDUSA implemented hard-coded references in its code which were bothersome to change, requiring the user to directly alter the pipeline’s source code.

In this context, we re-implemented MEDUSA’s software in a new, Nextflow-orchestrated pipeline, called EURYALE, which is more portable, providing dedicated Docker and Singularity images, as well as Nextflow’s native support of different HPC schedulers. It is also more parameterizable, removing the need for direct source code modifications. Euryale, as in Greek mythology, is the elder, immortal sister of Medusa, and this name represents the long-term support and stability we aim to provide with this new pipeline. Here we present the decisions we took in developing EURYALE and the results that can be acquired from it.

## II. METHODS

EURYALE was implemented using the nf-core [23] pipeline template, with each software included as part of the pipeline made available through BioConda [19] and BioContainers [24], enabling execution through conda, Docker and Singularity. The nf-core pipeline template and NextFlow enable high parameterization and customizable resource allocation, while also integrating well with High Performance Computing (HPC) schedulers, such as SLURM (<https://slurm.schedmd.com/>), as well as cloud environments, such as Amazon Web Services.

MicroView, included as a reporting tool in EURYALE, is a tool implemented in the Python programming language. It creates static HTML reports containing diversity information for a taxonomic profiling sample resulting from Kraken 2 [25] or Kaiju [26]. In each file, only the taxonomically classified reads (i.e., the reads properly attributed to a known taxon) are considered for diversity calculations. The abundance corresponds to the number of reads attributed to a given taxon. The diversity measures used, i.e. the Shannon index and Bray-Curtis distances, were computed using these read quantifications through the `alpha_diversity` and `beta_diversity` functions provided by SciKit-bio [27].

<sup>1</sup>Distinct and separate from MEDUSA as described by F. H. Karlsson, I. Nookaew, and J. Nielsen [17].

## III. RESULTS AND DISCUSSION

### A. Implementation details

MEDUSA’s tools were fully reimplemented in EURYALE (Fig. 1), including custom solutions implemented in the original MEDUSA like the `annotate` package for functional annotation. EURYALE consists of 5 general steps or “subworkflows”, the first of which comprises general read preprocessing. This step is then followed by host read removal, in the case of samples coming from hosts with known reference genomes, such as human microbiome data. We can then perform an optional - and disabled by default - assembly step, which comes prior to the taxonomic classification and functional annotation. Every step can be skipped, ensuring high customization to EURYALE’s users.

Both MEDUSA and EURYALE have as main inputs the FASTQ files containing reads, as well as an assortment of reference databases, for both the taxonomic classification and the functional annotation. While MEDUSA downloaded the references which were directly defined in its source code, EURYALE relies exclusively on user defined references, allowing for a more fine-grained customization. Additionally, EURYALE provides a separate workflow to download the same set of references as MEDUSA’s, in case full compatibility with previous results is required.

Below we provide an example command for executing the ‘download’ entry for EURYALE, which acquires these references. The different parameters select which references will be downloaded in this execution. Further details are explained in the EURYALE documentation: <https://dalmolingroup.github.io/euryale/>.

```
nextflow run dalmolingroup/euryale \
  --download_functional \
  --download_kaiju \
  --download_host \
  --outdir <output directory> \
  --entry download \
  --profile docker
```

EURYALE provides a new option for taxonomic classification, Kraken 2 [25], which was previously restricted to Kaiju [26]. This decision was taken in view of the Kraken’s team active development and high adoption by the metagenomics community, seen by the recent benchmarks which point to better precision and sensitivity when compared to other softwares[28][29][25].

One departure from MEDUSA was to remove steps that downloaded reference data, since these were prone to failure and required the user to manually alter source code in case of changing the reference databases to be used. Now EURYALE has over 20 parameters that can be added and modified in each execution by simply changing them on the command line, guaranteeing better adaptation to distinct use cases. By taking advantage of the nf-core infrastructure and template for creating pipelines [23], EURYALE also provides a high degree of parameterization for resource allocation, with different



Fig. 1. Diagram showing each step of the EURYALE pipeline, starting with FastQ reads as input. Circles represent steps that are enabled by default, while squares represent steps that are optional or disabled by default.

process labels that specify the number of CPUs and amount of RAM to be used by each process, which are strictly enforced when using cloud environments or HPC schedulers.

Below we provide an example command for executing all of EURYALE’s subworkflows in a Docker environment. The input to the pipeline is all samples, one-per-line, in a comma-separated table with sample name being the first column and the other two being paired FastQ files. The pipeline needs, by default, references for the Kaiju database - or Kraken 2 if you chose that taxonomy profiler - the FASTA reference file for DIAMOND [30], a FASTA reference for the host genome, in case the host read removal subworkflow is enabled, and an ID mapping file between the gene IDs in the reference and the desired functional database.

```
nextflow run dalmolingroup/euryale \
  --input samplesheet.csv \
  --outdir <output directory> \
  --kaiju_db <kaiju database> \
  --reference_fasta <reference FASTA> \
  --host_fasta <host reference FASTA> \
  --id_mapping <ID mapping file> \
  -profile docker
```

### B. Automated Reporting

One important change from MEDUSA to EURYALE is the enhancements regarding automated report creation and information interchange between the pipeline and its end user. In MEDUSA, we started working towards this with per-sample Krona [31] visualizations that provide a simple overview over the proportion of each species’ presence in each sample. Now, in EURYALE, apart from these same reports, we have also implemented two more: MultiQC [32] reports that show general quality control metrics and other measurements throughout various processes and MicroView reports, which have been customly developed for dealing with the taxonomic classification data coming from the results of this pipeline.

MultiQC reports contain a brief overview of nearly every tool included as part of EURYALE, containing visualizations

regarding sequence preprocessing (Fig. 2A), alignment and taxonomic classification (Fig. 2B). Visualizations such as these can both serve as quality control measures as well as indicate which taxa are more abundant in your samples, potentially generating insights about microbial biomarkers.

Apart from this, EURYALE also implements a new tool for reporting: MicroView. This new tool is focused towards calculating common diversity metrics based on the read classification data and providing simple, but insightful, visualizations about these metrics. Currently supported in this version of EURYALE are visualizations for Bray-Curtis beta-diversity, which is used to plot a Principal Coordinate Analysis, the Shannon index and Pielou evenness (Fig 3), making exploratory biodiversity analyses easily available, which could point to differences in biodiversity or species’ evenness among different groups of samples, or the general distribution of these measures, as the figure illustrates.

We see these automated reports as a strong improvement from the previous version, as they increase the pipeline’s usability, giving general directions on how to better explore your data and facilitating the generation of new research questions.

## IV. CONCLUSIONS

EURYALE, a novel metagenomics pipeline, bases itself upon previous well-curated pipelines like MEDUSA, taking its sensitivity, while improving its versatility. The pipeline adds support for container technology through Docker and Singularity; Native support for different HPC schedules as well as cloud environments; And more parameterization, allowing the user to choose different references and select specific pipeline steps to run. These changes have the potential to solve many issues common to WMS data analysis. Furthermore, EURYALE builds upon MEDUSA by adding new automated reports, providing metrics on data quality control, taxonomic classification and diversity, enhancing the methodology’s usability as well as empowering its users to quickly generate new scientific insights. Overall, EURYALE, a novel Nextflow pipeline for WMS data analysis, advances metagenomics data



processing by tackling some of its main issues, particularly regarding computational versatility and interpretability of results.

#### CODE AVAILABILITY STATEMENT

EURYALE is available through a GitHub repository, which can be found at <https://github.com/dalmolingroup/euryale>, with documentation available through <https://dalmolingroup.github.io/euryale/>. MicroView, although executed as part of EURYALE, can be executed stand-alone and has its own source code available in the following repository: <https://github.com/dalmolingroup/microview>.

#### ACKNOWLEDGMENTS

This work was supported by the governmental Brazilian agencies CAPES, grant 88887.834652/2023-00, and CNPq, grant 312305/2021-4. We would also like to thank NPAD / UFRN for computational resources.

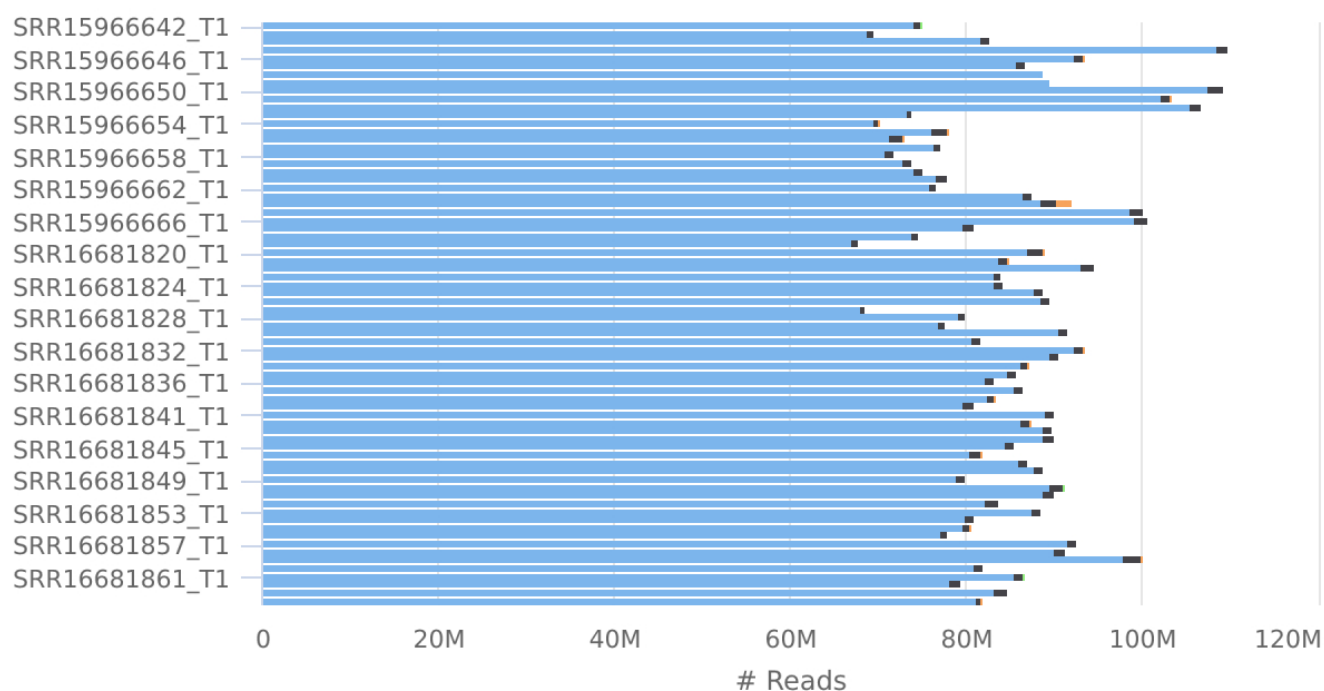
#### REFERENCES

- [1] J. R. Marchesi and J. Ravel, "The vocabulary of microbiome research: A proposal," *Microbiome*, vol. 3, no. 1, p. 31, Jul. 2015, ISSN: 2049-2618. DOI: 10.1186/s40168-015-0094-5. (visited on 06/19/2023).
- [2] S. Hunter, M. Corbett, H. Denise, *et al.*, "EBI metagenomics—a new resource for the analysis and archiving of metagenomic data," *Nucleic Acids Research*, vol. 42, no. D1, pp. D600–D606, Jan. 2014, ISSN: 0305-1048. DOI: 10.1093/nar/gkt961. (visited on 06/01/2024).
- [3] D. D'Agostino, L. Morganti, E. Corni, D. Cesini, and I. Merelli, "Combining Edge and Cloud computing for low-power, cost-effective metagenomics analysis," *Future Generation Computer Systems*, vol. 90, pp. 79–85, Jan. 2019, ISSN: 0167-739X. DOI: 10.1016/j.future.2018.07.036. (visited on 06/02/2024).
- [4] I. Laudadio, V. Fulci, L. Stronati, and C. Carissimi, "Next-Generation Metagenomics: Methodological Challenges and Opportunities," *OMICS: A Journal of Integrative Biology*, vol. 23, no. 7, pp. 327–333, Jul. 2019. DOI: 10.1089/omi.2019.0073. (visited on 06/01/2024).
- [5] P. ten Hoopen, R. D. Finn, L. A. Bongo, *et al.*, "The metagenomic data life-cycle: Standards and best practices," *GigaScience*, vol. 6, no. 8, gix047, Aug. 2017, ISSN: 2047-217X. DOI: 10.1093/gigascience/gix047. (visited on 06/01/2024).
- [6] F. P. Breitwieser, J. Lu, and S. L. Salzberg, "A review of methods and databases for metagenomic classification and assembly," *Briefings in Bioinformatics*, vol. 20, no. 4, pp. 1125–1136, Jul. 2019, ISSN: 1477-4054. DOI: 10.1093/bib/bbx120.
- [7] M. Ayling, M. D. Clark, and R. M. Leggett, "New approaches for metagenome assembly with short reads," *Briefings in Bioinformatics*, vol. 21, no. 2, pp. 584–594, Mar. 2020, ISSN: 1477-4054. DOI: 10.1093/bib/bbz020. (visited on 06/01/2024).
- [8] B. C. F. Santiago, I. D. de Souza, J. V. F. Cavalcante, *et al.*, "Metagenomic Analyses Reveal the Influence of Depth Layers on Marine Biodiversity on Tropical and Subtropical Regions," *Microorganisms*, vol. 11, no. 7, p. 1668, Jul. 2023, ISSN: 2076-2607. DOI: 10.3390/microorganisms11071668. (visited on 06/27/2023).
- [9] J. Mayneris-Perxachs, A. Castells-Nobau, M. Arnoriaga-Rodríguez, *et al.*, "Microbiota alterations in proline metabolism impact depression," *Cell Metabolism*, vol. 34, no. 5, 681–701.e10, May 2022, ISSN: 1550-4131. DOI: 10.1016/j.cmet.2022.04.001. (visited on 06/01/2024).
- [10] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, Apr. 2017, ISSN: 1546-1696. DOI: 10.1038/nbt.3820. (visited on 06/12/2023).
- [11] F. Mölder, K. P. Jablonski, B. Letcher, *et al.*, "Sustainable data analysis with Snakemake," *F1000Research*, vol. 10, p. 33, Apr. 2021, ISSN: 2046-1402. DOI: 10.12688/f1000research.29032.2. (visited on 06/12/2023).
- [12] J. V. F. Cavalcante, I. D. de Souza, D. A. d. A. Morais, and R. J. S. Dalmolin, "Bridging the Gaps in Meta-Omic Analysis: Workflows and Reproducibility," *OMICS: A Journal of Integrative Biology*, Nov. 2023. DOI: 10.1089/omi.2023.0232. (visited on 12/02/2023).
- [13] L. Wratten, A. Wilm, and J. Göke, "Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers," *Nature Methods*, vol. 18, no. 10, pp. 1161–1168, Oct. 2021, ISSN: 1548-7105. DOI: 10.1038/s41592-021-01254-9. (visited on 06/12/2023).
- [14] S. Krakau, D. Straub, H. Gourel, G. Gabernet, and S. Nahnsen, "Nf-core/mag: A best-practice pipeline for metagenome hybrid assembly and binning," *NAR Genomics and Bioinformatics*, vol. 4, no. 1, lqac007, Mar. 2022, ISSN: 2631-9268. DOI: 10.1093/nargab/lqac007. (visited on 04/04/2023).
- [15] V. W. Salazar, B. Shaban, M. d. M. Quiroga, *et al.*, "Metaphor—A workflow for streamlined assembly and binning of metagenomes," *GigaScience*, vol. 12, giad055, Jan. 2023, ISSN: 2047-217X. DOI: 10.1093/gigascience/giad055. (visited on 03/26/2024).
- [16] T. A. Gurbich, A. Almeida, M. Beracochea, *et al.*, "MGnify Genomes: A Resource for Biome-specific Microbial Genome Catalogues," *Journal of Molecular Biology*, Computation Resources for Molecular Biology, vol. 435, no. 14, p. 168016, Jul. 2023, ISSN: 0022-2836. DOI: 10.1016/j.jmb.2023.168016. (visited on 03/23/2024).
- [17] F. H. Karlsson, I. Nookaew, and J. Nielsen, "Metagenomic Data Utilization and Analysis (MEDUSA) and Construction of a Global Gut Microbial Gene Catalogue," *PLOS Computational Biology*, vol. 10, no. 7,

- e1003706, 10 de jul. de 2014, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003706. (visited on 06/01/2024).
- [18] D. A. A. Morais, J. V. F. Cavalcante, S. S. Monteiro, M. A. B. Pasquali, and R. J. S. Dalmolin, "MEDUSA: A Pipeline for Sensitive Taxonomic Classification and Flexible Functional Annotation of Metagenomic Shotgun Sequences," *Frontiers in Genetics*, vol. 13, 2022, ISSN: 1664-8021. (visited on 07/10/2023).
- [19] B. Grüning, R. Dale, A. Sjödin, *et al.*, "Bioconda: Sustainable and comprehensive software distribution for the life sciences," *Nature Methods*, vol. 15, no. 7, pp. 475–476, Jul. 2018, ISSN: 1548-7105. DOI: 10.1038/s41592-018-0046-7. (visited on 06/12/2023).
- [20] S. Grayson, D. Marinov, D. S. Katz, and R. Milewicz, "Automatic Reproduction of Workflows in the Snake-make Workflow Catalog and nf-core Registries," in *Proceedings of the 2023 ACM Conference on Reproducibility and Replicability*, ser. ACM REP '23, New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 74–84, ISBN: 9798400701764. DOI: 10.1145/3589806.3600037. (visited on 07/03/2023).
- [21] M. Jackson, K. Kavoussanakis, and E. W. J. Wallace, "Using prototyping to choose a bioinformatics workflow management system," *PLOS Computational Biology*, vol. 17, no. 2, e1008622, 25 de fev. de 2021, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008622. (visited on 03/23/2024).
- [22] F. M. Celebi, E. McDaniel, and T. Reiter, "Creating reproducible workflows for complex computational pipelines," *Arcadia Science*, Mar. 2023. DOI: 10.57844/arcadia-cc5j-a519. (visited on 04/04/2023).
- [23] P. A. Ewels, A. Peltzer, S. Fillinger, *et al.*, "The nf-core framework for community-curated bioinformatics pipelines," *Nature Biotechnology*, vol. 38, no. 3, pp. 276–278, Mar. 2020, ISSN: 1546-1696. DOI: 10.1038/s41587-020-0439-x. (visited on 06/12/2023).
- [24] F. da Veiga Leprevost, B. A. Grüning, S. Alves Aflitos, *et al.*, "BioContainers: An open-source and community-driven framework for software standardization," *Bioinformatics*, vol. 33, no. 16, pp. 2580–2582, Aug. 2017, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx192. (visited on 07/10/2023).
- [25] D. E. Wood, J. Lu, and B. Langmead, "Improved metagenomic analysis with Kraken 2," *Genome Biology*, vol. 20, no. 1, p. 257, Nov. 2019, ISSN: 1474-760X. DOI: 10.1186/s13059-019-1891-0. (visited on 03/24/2024).
- [26] P. Menzel, K. L. Ng, and A. Krogh, "Fast and sensitive taxonomic classification for metagenomics with Kaiju," *Nature Communications*, vol. 7, no. 1, p. 11257, Apr. 2016, ISSN: 2041-1723. DOI: 10.1038/ncomms11257. (visited on 03/24/2024).
- [27] J. R. Rideout, G. Caporaso, E. Bolyen, *et al.*, *Biocore/scikit-bio: Scikit-bio 0.5.9: Maintenance release*, Zenodo, Aug. 2023. DOI: 10.5281/zenodo.8209901. (visited on 03/26/2024).
- [28] A. R. Odom, T. Faits, E. Castro-Nallar, K. A. Crandall, and W. E. Johnson, "Metagenomic profiling pipelines improve taxonomic classification for 16S amplicon sequencing data," *Scientific Reports*, vol. 13, no. 1, p. 13957, Aug. 2023, ISSN: 2045-2322. DOI: 10.1038/s41598-023-40799-x. (visited on 03/26/2024).
- [29] F. Jurado-Rueda, L. Alonso-Guirado, T. E. Perea-Chamblée, *et al.*, "Benchmarking of microbiome detection tools on RNA-seq synthetic databases according to diverse conditions," *Bioinformatics Advances*, vol. 3, no. 1, vbad014, Jan. 2023, ISSN: 2635-0041. DOI: 10.1093/bioadv/vbad014. (visited on 03/26/2024).
- [30] B. Buchfink, K. Reuter, and H.-G. Drost, "Sensitive protein alignments at tree-of-life scale using DIAMOND," *Nature Methods*, vol. 18, no. 4, pp. 366–368, Apr. 2021, ISSN: 1548-7105. DOI: 10.1038/s41592-021-01101-x. (visited on 03/27/2024).
- [31] B. D. Ondov, N. H. Bergman, and A. M. Phillippy, "Interactive metagenomic visualization in a Web browser," *BMC Bioinformatics*, vol. 12, no. 1, p. 385, Sep. 2011, ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-385. (visited on 03/25/2024).
- [32] P. Ewels, M. Magnusson, S. Lundin, and M. Käller, "MultiQC: Summarize analysis results for multiple tools and samples in a single report," *Bioinformatics*, vol. 32, no. 19, pp. 3047–3048, Oct. 2016, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw354. (visited on 03/12/2024).

A

## Fastp: Filtered Reads



● Passed Filter ● Low Quality ● Too Many N ● Too Short ● Too Long

Created with MultiQC

B

## Kaiju: Top taxa



● Faecalibacterium prausnitzii ● Prevotella copri ● Subdoligranulum sp. APC924/74  
 ● [Eubacterium] rectale ● Bifidobacterium adolescentis ● Other  
 ● Cannot be assigned ● Unclassified

Created with MultiQC

Fig. 2. Example figures included as part of EURYALE's MultiQC report. **A:** Figure for the quality control section, showing the number of reads filtered out in each sample by fastp and their respective thresholds for filtering. **B:** Figure for the taxonomic classification section, showing the percentage of reads assigned to different taxa in each sample by Kaiju. In the HTML report the figures are interactive.

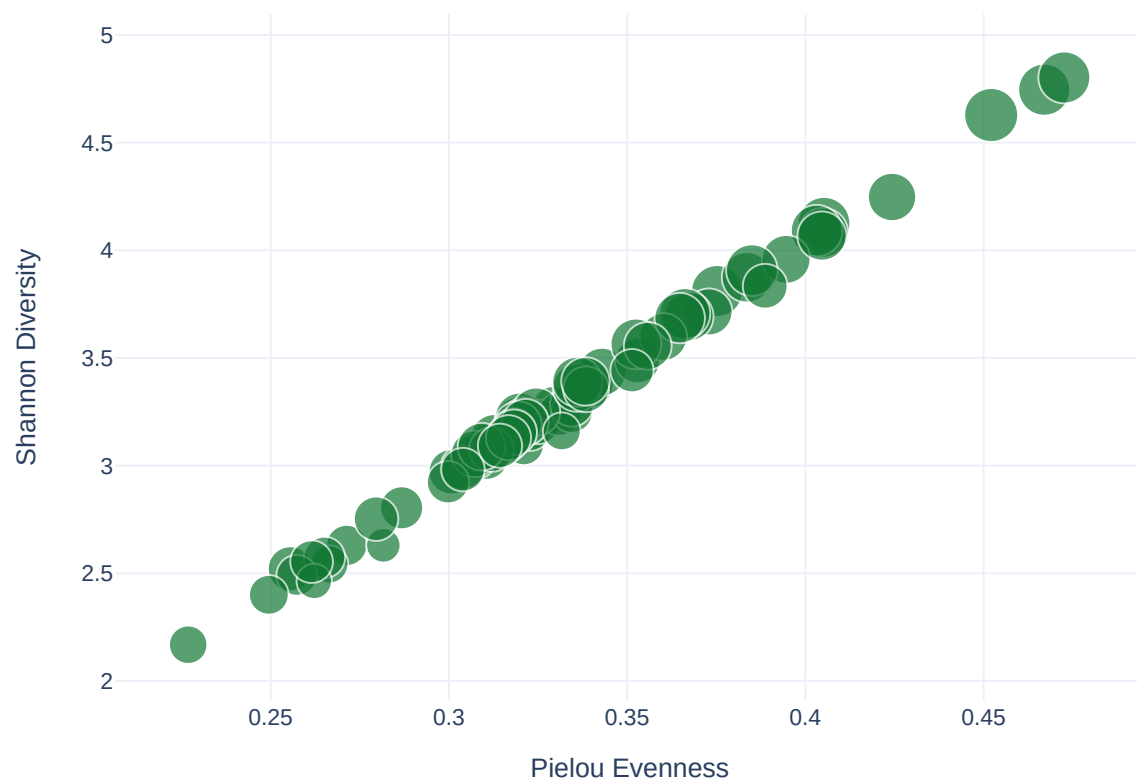


Fig. 3. Figure included as part of the EURYALE's MicroView report. It shows a scatterplot of Pielou Evenness and Shannon's diversity in each sample, represented by a dot. The size of each dot corresponds to the number of distinct species classified in each sample. In the HTML report the figure is interactive.

### 3 DISCUSSÃO

Ao averiguarmos o estado da arte dos fluxos de trabalho computacional presentes na área da metagenômica, pudemos concluir que a adoção de tecnologias de containerização e orquestradores de execução ainda é um tanto limitada. Essas limitações tornam-se mais evidentes sobretudo no contexto de metodologias de metagenômica livres de montagem. Tais metodologias são mais escassas e menos computacionalmente intensas quando comparadas às abordagens baseadas em montagem. Ainda assim, elas necessitam processamento adequado, de forma reprodutível, replicável e automatizável, especialmente por possibilitarem uma análise realizável em infraestruturas de menor porte e por possuírem melhor sensibilidade ao tratar dados com baixa cobertura (Ayling; Clark; Leggett, 2020), ambos fatores comuns quando se tratando da produção e processamento de dados metagenômicos em ambientes com financiamento científico limitado, como no sul global.

No contexto de possíveis metodologias, decidimos então aplicar os princípios postos para a metodologia MEDUSA (Moraes et al., 2022), que apresentou resultados superiores a fluxos de trabalho semelhantes, além de ter obtido suas ferramentas a partir de curadoria manual, com rigorosos processos de benchmarking. Portanto, tomando vantagem da curadoria precedente, decidimos então re-implementar o MEDUSA, utilizando-se agora do gerenciador de pipelines Nextflow e de tecnologias de containerização Docker e Singularity.

Nextflow foi escolhido sobretudo devido à facilidade e rapidez de desenvolvimento, conjunta ao suporte multiplataforma mais abrangente que o orquestrador anteriormente escolhido para o MEDUSA, Snakemake. Ademais, Nextflow já foi selecionado como a melhor opção para desenvolvimento de fluxos de trabalho em comparações anteriores (Jackson; Kavoussanakis; Wallace, 2021) (Celebi; McDaniel; Reiter, 2023). Outro aspecto que levou à decisão por Nextflow foi a existência da comunidade nf-core, que realiza trabalhos de curadoria de fluxos de trabalho em bioinformática e fornecem templates para a criação de pipelines modulares, altamente parametrizáveis e que possibilitem melhor manutenção a longo prazo (Ewels, P. A. et al., 2020).

Apesar de reimplementarmos o MEDUSA por completo como EURYALE, também avançamos em algumas limitações do software original, sobretudo interpretabilidade dos dados e parametrização.

Quanto ao primeiro ponto, notávamos uma clara ausência, no MEDUSA, de visualizações e relatórios que forneçam informações preliminares e análises exploratórias acerca do conjunto de dados processado. No entanto, acessibilidade de dados através de relatórios e figuras interativas é essencial para tornar análises mais compreensíveis e reprodutíveis, sobretudo em um campo com complexidade alta de informação, com muitos arquivos gerados por análise, como a metagenômica.

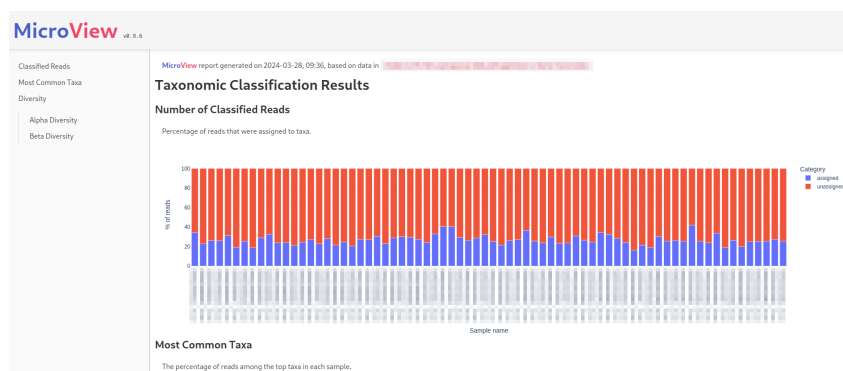
Nesse sentido, primeiro interligamos as diversas ferramentas agora presentes no EURYALE através do MultiQC, um agregador de informação de dados de bioinformática (Ewels, P. et al., 2016) que permitiu disponibilizar um relatório interativo e configurável com informações a respeito das etapas de controle de qualidade, classificação taxonômica e do alinhamento que precede a anotação funcional (Figura 2).

Figura 2 – Exemplo do cabeçalho de um relatório gerado através da ferramenta MultiQC, implementada como parte integrante do EURYALE.



Ademais, através da implementação de uma nova ferramenta, MicroView, também inclusa no EURYALE, disponibilizamos também ao usuário métricas pré-calculadas de diversidade, ilustrando, de forma inicial e exploratória, o quadro geral resultante da classificação taxonômica (Figura 3).

Figura 3 – Exemplo do cabeçalho de um relatório gerado através da ferramenta Microview, escrita em Python e implementada como parte integrante do EURYALE.

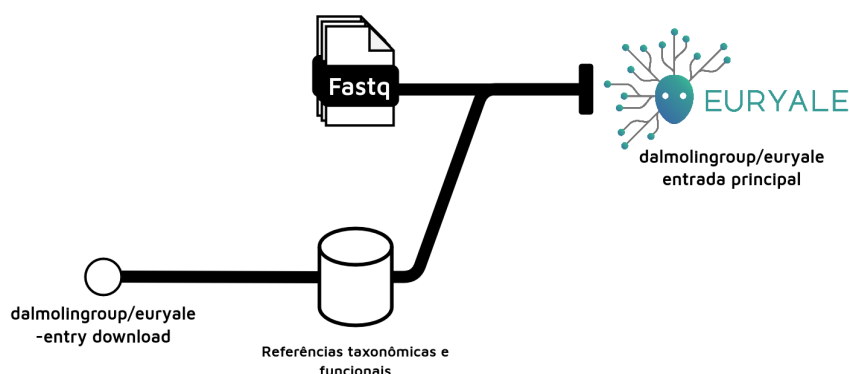


O MicroView também atua como um pacote na linguagem Python de programação, possibilitando sua execução fora do fluxo de trabalho. De forma geral, tais relatórios, com visualizações interativas, aumentam a acessibilidade de dados das análises realizadas com o EURYALE, facilitando não apenas a exploração inicial e a geração de descobertas científicas pelos pesquisadores usuários da metodologia, mas também a exploração posterior dos resultados, promovendo re-análises (Perkel, 2018).

Vale ressaltar também os aprimoramentos em padronização de software e parametrização trazidos da mudança do MEDUSA para o EURYALE. Ao implementarmos a metodologia, optamos por utilizar a infraestrutura da nf-core, uma comunidade que busca padronizar o desenvolvimento de fluxos de trabalho que utilizam Nextflow, com ênfase em modularidade, testes contínuos e documentação (Ewels, P. A. et al., 2020). A comunidade implementa uma suíte de software em Python para a criação e manutenção de fluxos de trabalho (<https://nf-co.re/docs/nf-core-tools/installation>), que foi amplamente utilizada no desenvolvimento do EURYALE. Ao utilizarmos das ferramentas e diretrizes da nf-core, pudemos superar algo que foi a principal crítica de usuários em relação ao MEDUSA: A estaticidade das referências e a pouca parametrização.

Quanto ao primeiro ponto, optamos por uma solução que possibilite a utilização de bancos de dados referência equivalentes aos utilizados pelo MEDUSA, que é o que tomamos como padrão da análise. Nesse sentido, implementamos uma entrada alternativa ao fluxo de trabalho (`-entry download`), que possibilita que os usuários transfiram os bancos de dados referência para sua máquina de análise, alimentando assim o *pipeline* propriamente dito (Figura 4).

Figura 4 – Diagrama ilustrando como a entrada de download do EURYALE adquire os bancos de dados que alimentam a entrada principal. Com uma análise típica do zero sendo constituída por ambas etapas.



Apesar disso, devido à modularidade advinda do *template nf-core*, o EURYALE é modelado para funcionar com quaisquer outro banco de dado compatível com as suas ferramentas integrantes. Dessa maneira, o usuário pode tanto modificar os parâmetros de *download* na entrada específica, quanto modificar diretamente os parâmetros de referência presentes na entrada principal do *pipeline*, caso ele possua referências pré-adquiridas. Esses detalhes funcionais distinguem a usabilidade do EURYALE com a de seu precedente, não requerendo que os usuários alterem o código fonte para utilizar referências diferentes.

Além disso, adicionamos mais parâmetros ao EURYALE, parâmetros estes tanto que alterem as etapas de análise quanto aqueles que alteram os recursos computa-

cionais alocados a estas etapas. A exemplo dessa mudança, podemos ver como a metodologia agora apoia um passo opcional de montagem, semelhante ao descrito no artigo original do MEDUSA, no entanto não aplicado diretamente na sua versão orquestrável original. O novo fluxo de trabalho também oferece mais opções de classificação taxonômica, além de parâmetros que possibilitam desabilitar diferentes passos de análise, como a descontaminação de leituras do hospedeiro, possibilitando que a abordagem se adapte a diferentes contextos de análise. Adicionalmente, outro ponto originado pela utilização do template *nf-core* são os diferentes rótulos de alocação de recursos, que permitem um gerenciamento fácil, da quantidade de recursos que pode ser alocada a cada passo de análise, através de um único arquivo de configuração. Essa adição permite que a metodologia seja executada em diferentes infraestruturas computacionais.

Em última instância, vale ressaltar como a utilização do template *nf-core* também possibilita a implantação do fluxo de trabalho na *Seqera Platform*, com a geração automática de uma interface gráfica, permitindo a usuários executar a metodologia sem necessariamente utilizarem uma interface em linha de comando, dado que, é claro, possuam cadastro na plataforma (Figura 5).

Figura 5 – Porção inicial da interface gráfica do EURYALE na Seqera Platform (<<https://cloud.seqera.io/>>). Através desse formulário usuários podem selecionar os parâmetros a serem utilizados para a execução do fluxo de trabalho.

The screenshot shows the 'dalmolingroup/euryale pipeline parameters' form in the Seqera Platform. The form is divided into several sections:

- Workflow run name:** A text field containing 'curious\_lamport' and a 'Labels' dropdown menu. Below the text field is a note: 'A unique name randomly assigned to this workflow run. Customize this with a name of your choice (optional)'. Below the dropdown is a note: 'A label must contain at least 2 alphanumeric characters.'
- Input/output options:** A section titled 'Define where the pipeline should find input data and save output data.' It contains four fields: 'input', 'outdir', 'email', and 'multiqc\_title'. Each field has a 'Browse' button next to it. Below the 'input' field is a note: 'Path to comma-separated file containing information about the samples in the experiment.' Below the 'outdir' field is a note: 'The output directory where the results will be saved. You have to use absolute paths to storage on Cloud infrastructure.'
- Right sidebar:** A list of steps: 'input', 'outdir', 'email', 'multiqc\_title', 'save\_dbs', 'Skip Steps', 'Decontamination', 'Alignment', 'Taxonomy', 'Functional', 'Assembly', 'Reference genome options', 'Download Entry', and 'Generic options'. At the bottom of the sidebar are three buttons: 'Show hidden params', 'Launch settings', and 'Launch'.



## 4 CONCLUSÃO

Ao averiguar o estado do desenvolvimento de *software* em meta-ômica, pudemos observar uma baixa aderência a boas práticas de desenvolvimento, sobretudo no que tange *pipelines* de análise de dados. Nesse sentido, nos baseamos em uma metodologia anterior para desenvolvermos um fluxo de trabalho, denominado EURYALE, que possibilita uma análise completa de dados de MS, e possui isolamento de dependências, alta parametrização, relatórios automáticos com visualizações interativas e uma documentação expansiva, com exemplos práticos. Nossa iniciativa com o desenvolvimento do EURYALE não só avança ideais de desenvolvimento de *software* científico, como também facilita com que haja um processamento de dados acurado, estável e sensível. No entanto, deve-se ressaltar que, apesar de avançarmos na abordagem fundamental da metagenômica, metodologias de integração de dados meta-ômicos, isto é, metagenômico e metatranscriptômico, ainda são relativamente escassas. Este aspecto se torna ainda mais evidente ao considerarmos fluxos de trabalho automatizados que sigam boas práticas de desenvolvimento de *software*. Portanto, futuras iniciativas devem ser orientadas para que a realização desses futuros métodos siga também os princípios delineados no presente trabalho.

## REFERÊNCIAS

- ALTSCHUL, Stephen et al. The anatomy of successful computational biology software. **Nature Biotechnology**, v. 31, n. 10, p. 894–897, out. 2013. Publisher: Nature Publishing Group. DOI: 10.1038/nbt.2721. Disponível em: <https://www.nature.com/articles/nbt.2721>.
- AYLING, Martin; CLARK, Matthew D; LEGGETT, Richard M. New approaches for metagenome assembly with short reads. **Briefings in Bioinformatics**, v. 21, n. 2, p. 584–594, 23 mar. 2020. DOI: 10.1093/bib/bbz020. Disponível em: <https://doi.org/10.1093/bib/bbz020>.
- BREITWIESER, Florian P.; LU, Jennifer; SALZBERG, Steven L. A review of methods and databases for metagenomic classification and assembly. **Briefings in Bioinformatics**, v. 20, n. 4, p. 1125–1136, 19 jul. 2019. PMID: 29028872 PMCID: PMC6781581. DOI: 10.1093/bib/bbx120.
- CELEBI, Feridun Mert; MCDANIEL, Elizabeth; REITER, Taylor. Creating reproducible workflows for complex computational pipelines. **Arcadia Science**, 7 mar. 2023. Publisher: Arcadia Science. DOI: 10.57844/arcadia-cc5j-a519. Disponível em: <https://research.arcadiascience.com/pub/perspective-reproducible-workflows/release/2>.
- CHIARELLO, Marlène et al. Ranking the biases: The choice of OTUs vs. ASVs in 16S rRNA amplicon data analysis has stronger effects on diversity measures than rarefaction and OTU identity threshold. **PLOS ONE**, v. 17, n. 2, e0264443, inverno 2022. Publisher: Public Library of Science. DOI: 10.1371/journal.pone.0264443. Disponível em: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264443>.
- COMIN, Matteo et al. Comparison of microbiome samples: methods and computational challenges. **Briefings in Bioinformatics**, v. 22, n. 1, p. 88–95, 18 jan. 2021. DOI: 10.1093/bib/bbaa121. Disponível em: <https://academic.oup.com/bib/article/22/1/88/5861761>.
- DI TOMMASO, Paolo et al. Nextflow enables reproducible computational workflows. **Nature Biotechnology**, v. 35, n. 4, p. 316–319, abr. 2017. Number: 4 Publisher: Nature Publishing Group. DOI: 10.1038/nbt.3820. Disponível em: <https://www.nature.com/articles/nbt.3820>.
- DOUGLAS, Gavin M. et al. PICRUSt2 for prediction of metagenome functions. **Nature Biotechnology**, v. 38, n. 6, p. 685–688, jun. 2020. Publisher: Nature Publishing Group. DOI: 10.1038/s41587-020-0548-6. Disponível em: <https://www.nature.com/articles/s41587-020-0548-6>.
- EWELS, Philip et al. MultiQC: summarize analysis results for multiple tools and samples in a single report. **Bioinformatics**, v. 32, n. 19, p. 3047–3048, 1 out. 2016.

DOI: 10.1093/bioinformatics/btw354. Disponível em:  
<https://doi.org/10.1093/bioinformatics/btw354>.

EWELS, Philip A. et al. The nf-core framework for community-curated bioinformatics pipelines. **Nature Biotechnology**, v. 38, n. 3, p. 276–278, mar. 2020. Number: 3 Publisher: Nature Publishing Group. DOI: 10.1038/s41587-020-0439-x. Disponível em: <https://www.nature.com/articles/s41587-020-0439-x>.

JACKSON, Michael; KAVOUSSANAKIS, Kostas; WALLACE, Edward W. J. Using prototyping to choose a bioinformatics workflow management system. **PLOS Computational Biology**, v. 17, n. 2, e1008622, **springN** 2021. Publisher: Public Library of Science. DOI: 10.1371/journal.pcbi.1008622. Disponível em: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008622>.

KADRI, Sabah et al. Containers in Bioinformatics: Applications, Practical Considerations, and Best Practices in Molecular Pathology. **The Journal of Molecular Diagnostics**, v. 24, n. 5, p. 442–454, 1 maio 2022. DOI: 10.1016/j.jmoldx.2022.01.006. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1525157822000381>.

KRAKAU, Sabrina et al. nf-core/mag: a best-practice pipeline for metagenome hybrid assembly and binning. **NAR Genomics and Bioinformatics**, v. 4, n. 1, lqac007, 1 mar. 2022. DOI: 10.1093/nargab/lqac007. Disponível em: <https://doi.org/10.1093/nargab/lqac007>.

LIU, Yong-Xin et al. A practical guide to amplicon and metagenomic analysis of microbiome data. **ProteinCell**, v. 12, n. 5, p. 315–330, 1 maio 2021. DOI: 10.1007/s13238-020-00724-8. Disponível em: <https://doi.org/10.1007/s13238-020-00724-8>.

MAGNABOSCO, Cara et al. Toward a Natural History of Microbial Life. **Annual Review of Earth and Planetary Sciences**, v. 52, Volume 52, 2024, p. 85–108, 23 jul. 2024. Publisher: Annual Reviews. DOI: 10.1146/annurev-earth-031621-070542. Disponível em: <https://www.annualreviews.org/content/journals/10.1146/annurev-earth-031621-070542>.

MANGUL, Serghei; MARTIN, Lana S. et al. Improving the usability and archival stability of bioinformatics software. **Genome Biology**, v. 20, n. 1, p. 47, 27 fev. 2019. DOI: 10.1186/s13059-019-1649-8. Disponível em: <https://doi.org/10.1186/s13059-019-1649-8>.

MANGUL, Serghei; MOSQUEIRO, Thiago et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. **PLOS Biology**, v. 17, n. 6, e3000333, 20 jun. 2019. Publisher: Public Library of Science. DOI: 10.1371/journal.pbio.3000333. Disponível em: <https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.3000333>.

- MARCHESI, Julian R.; RAVEL, Jacques. The vocabulary of microbiome research: a proposal. **Microbiome**, v. 3, n. 1, p. 31, 30 jul. 2015. DOI: 10.1186/s40168-015-0094-5. Disponível em: <https://doi.org/10.1186/s40168-015-0094-5>.
- MÖLDER, Felix et al. Sustainable data analysis with Snakemake. **F1000Research**, v. 10, p. 33, 19 abr. 2021. PMID: 34035898 PMCID: PMC8114187. DOI: 10.12688/f1000research.29032.2. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8114187/>.
- MORAIS, Diego A. A. et al. MEDUSA: A Pipeline for Sensitive Taxonomic Classification and Flexible Functional Annotation of Metagenomic Shotgun Sequences. **Frontiers in Genetics**, v. 13, 2022. Disponível em: <https://www.frontiersin.org/articles/10.3389/fgene.2022.814437>.
- PERKEL, Jeffrey M. Data visualization tools drive interactivity and reproducibility in online publishing. **Nature**, v. 554, n. 7690, p. 133–134, 30 jan. 2018. Bandiera\_abtest: a Cg\_type: Toolbox Publisher: Nature Publishing Group Subject\_term: Authorship, Peer review, Publishing. DOI: 10.1038/d41586-018-01322-9. Disponível em: <https://www.nature.com/articles/d41586-018-01322-9>.
- SALAZAR, Vinícius W et al. Metaphor—A workflow for streamlined assembly and binning of metagenomes. **GigaScience**, v. 12, giad055, 1 jan. 2023. DOI: 10.1093/gigascience/giad055. Disponível em: <https://doi.org/10.1093/gigascience/giad055>.
- STRAUB, Daniel et al. Interpretations of Environmental Microbial Community Studies Are Biased by the Selected 16S rRNA (Gene) Amplicon Sequencing Pipeline. **Frontiers in Microbiology**, v. 11, 23 out. 2020. Publisher: Frontiers. DOI: 10.3389/fmicb.2020.550420. Disponível em: <https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2020.550420/full>.
- TREMBLAY, Julien; SCHREIBER, Lars; GREER, Charles W. High-resolution shotgun metagenomics: the more data, the better? **Briefings in Bioinformatics**, v. 23, n. 6, bbac443, 1 nov. 2022. DOI: 10.1093/bib/bbac443. Disponível em: <https://doi.org/10.1093/bib/bbac443>.
- WOOLEY, John C.; GODZIK, Adam; FRIEDBERG, Iddo. A Primer on Metagenomics. **PLOS Computational Biology**, v. 6, n. 2, e1000667, summerN 2010. Publisher: Public Library of Science. DOI: 10.1371/journal.pcbi.1000667. Disponível em: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000667>.

## **5 ANEXO I**

Segue abaixo a documentação servida no *site* do EURYALE

*Acessada na data 24 de novembro de 2024*



# Table of Contents

- **1 Home**
  - [1.1 Introduction](#)
  - [1.2 Pipeline summary](#)
    - [1.2.1 Pre-processing](#)
    - [1.2.2 Assembly](#)
    - [1.2.3 Taxonomic classification](#)
    - [1.2.4 Functional annotation](#)
  - [1.3 Quick Start](#)
  - [1.4 Databases and references](#)
    - [1.4.1 Alignment](#)
    - [1.4.2 Taxonomic classification](#)
    - [1.4.3 Functional annotation](#)
    - [1.4.4 Host reference](#)
  - [1.5 Documentation](#)
  - [1.6 Credits](#)
  - [1.7 Citations](#)
- **2 dalmolingroup/euryale: Usage**
  - [2.1 Introduction](#)
  - [2.2 Samplesheet input](#)
    - [2.2.1 Multiple runs of the same sample](#)
    - [2.2.2 Full samplesheet](#)
  - [2.3 Running the pipeline](#)
    - [2.3.1 Updating the pipeline](#)
    - [2.3.2 Reproducibility](#)
  - [2.4 Core Nextflow arguments](#)
  - [2.5 Custom configuration](#)
    - [2.5.1 Resource requests](#)
    - [2.5.2 Updating containers \(advanced users\)](#)
  - [2.6 Running in the background](#)
  - [2.7 Nextflow memory requirements](#)
- **3 dalmolingroup/euryale: Output**

- [3.1 Introduction](#)
- [3.2 Pipeline overview](#)
  - [3.2.1 Kaiju](#)
  - [3.2.2 Krona](#)
  - [3.2.3 Diamond](#)
  - [3.2.4 Annotate](#)
  - [3.2.5 MEGAHIT](#)
  - [3.2.6 MultiQC](#)
  - [3.2.7 Pipeline information](#)
- **[4 dalmolingroup/euryale: Citations](#)**
  - [4.1 Pipeline tools](#)
  - [4.2 Software packaging/containerisation tools](#)
- **I Reference**
  - **[5 dalmolingroup/euryale pipeline parameters](#)**
    - [5.1 Input/output options](#)
    - [5.2 Skip Steps](#)
    - [5.3 Decontamination](#)
    - [5.4 Alignment](#)
    - [5.5 Taxonomy](#)
    - [5.6 Functional](#)
    - [5.7 Assembly](#)
    - [5.8 Reference genome options](#)
    - [5.9 Download Entry](#)
    - [5.10 Max job request options](#)
    - [5.11 Generic options](#)

# 1 Home

 nf-core CI passing  docs passing

nextflow DSL2 ≥22.10.1  run with docker  run with singularity



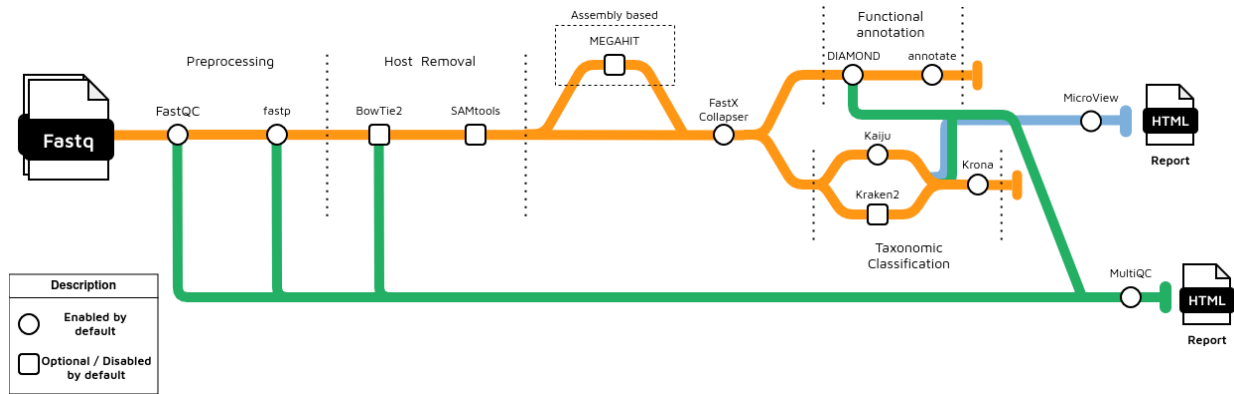
## 1.1 Introduction

**dalmolingroup/euryale** is a pipeline for taxonomic classification and functional annotation of metagenomic reads. Based on [MEDUSA](#).

The pipeline is built using [Nextflow](#), a workflow tool to run tasks across multiple compute infrastructures in a very portable manner. It uses Docker/Singularity containers making installation trivial and results highly reproducible. The [Nextflow DSL2](#) implementation of this pipeline uses one container per process which makes it much easier to maintain and update software dependencies. Where possible, these processes have been submitted to and installed from [nf-core/modules](#) in order to make them available to all nf-core pipelines, and to everyone within the Nextflow community!

## 1.2 Pipeline summary





## 1.2.1 Pre-processing

- Read QC ( [FastQC](#) )
- Read trimming and merging ( [fastp](#) )
- (optionally) Host read removal ( [BowTie2](#) )
- Duplicated sequence removal ( [fastx collapser](#) )
- Present QC and other data ( [MultiQC](#) )

## 1.2.2 Assembly

- (optionally) Read assembly ( [MEGAHIT](#) )

## 1.2.3 Taxonomic classification

- Sequence classification ( [Kaiju](#) )
- Sequence classification ( [Kraken2](#) )
- Visualization ( [Krona](#) )

## 1.2.4 Functional annotation

- Sequence alignment ( [DIAMOND](#) )
- Map alignment matches to functional database ( [annotate](#) )

## 1.3 Quick Start

1. Install [Nextflow](#) ( [>=22.10.1](#) )
2. Install any of [Docker](#) , [Singularity](#) (you can follow [this tutorial](#)), [Podman](#) , [Shifter](#) or [Charliecloud](#) for full pipeline reproducibility (you can use [Conda](#) both to install Nextflow itself and also to manage software within pipelines. Please only use it within pipelines as a last resort; see [docs](#)).
3. Download the pipeline and test it on a minimal dataset with a single command:

```
nextflow run dalmolingroup/euryale -profile test, YOURPROFILE --outdir <OUTDIR>
```

Note that some form of configuration will be needed so that Nextflow knows how to fetch the required software. This is usually done in the form of a config profile ( `YOURPROFILE` in the example command above). You can chain multiple config profiles in a comma-separated string.

- The pipeline comes with config profiles called `docker`, `singularity`, `podman`, `shifter`, `charliecloud` and `conda` which instruct the pipeline to use the named tool for software management. For example, `-profile test,docker`.
- Please check [nf-core/configs](#) to see if a custom config file to run nf-core pipelines already exists for your Institute. If so, you can simply use `-profile <institute>` in your command. This will enable either `docker` or `singularity` and set the appropriate execution settings for your local compute environment.
- If you are using `singularity`, please use the `nf-core download` command to download images first, before running the pipeline. Setting the `NXF_SINGULARITY_CACHEDIR` or `singularity.cacheDir` Nextflow options enables you to store and re-use the images from a central location for future pipeline runs.
- If you are using `conda`, it is highly recommended to use the `NXF_CONDA_CACHEDIR` or `conda.cacheDir` settings to store the environments in a central location for future pipeline runs.
- Start running your own analysis!

```
nextflow run dalmolingroup/euryale \
--input samplesheet.csv \
--outdir <OUTDIR> \
--kaiju_db kaiju_reference \
--reference_fasta diamond_fasta \
--host_fasta host_reference_fasta \
--id_mapping id_mapping_file \
-profile <docker/singularity/podman/shifter/charliecloud/conda/institute>
```

## 1.4 Databases and references

A question that pops up a lot is: Since Euryale requires a lot of reference parameters, where can I find these references?

One option is to execute EURYALE's download entry, which will download the necessary databases for you. This is the recommended way to get started with the pipeline. This uses the same sources as EURYALE's predecessor MEDUSA.

```
nextflow run dalmolingroup/euryale \
  --download_functional \
  --download_kaiju \
  --download_host \
  --outdir <output directory> \
  --entry download \
  --profile <docker/singularity/podman/shifter/charliecloud/conda/institute>
```

Check out the full documentation for a full list of [EURYALE's download parameters](#). In case you download the Kraken2 database ( `--download_kraken` ), make sure to extract it using the following command before using it in the pipeline:

```
tar -xvf kraken2_db.tar.gz
```

Below we provide a short list of places where you can find these databases. But, of course, we're not limited to these references: Euryale should be able to process your own databases, should you want to build them yourself.

### 1.4.1 Alignment

For the alignment you can either provide `--diamond_db` for a pre-built DIAMOND database, or you can provide `--reference_fasta`. For reference fasta, by default Euryale expects something like [NCBI-nr](#), but similarly formatted reference databases should also suffice.

### 1.4.2 Taxonomic classification

At its current version, Euryale doesn't build a reference taxonomic database, but pre-built ones are supported.

- If you're using Kaiju (the default), you can provide a reference database with `--kaiju_db` and provide a .tar.gz file like the ones provided in the [official Kaiju website](#). We have extensively tested Euryale with the 2021 version of the nr database and it should work as expected.
- If you're using Kraken2 (By supplying `--run_kraken2` ), we expect something like the [pre-built .tar.gz databases provided by the Kraken2 developers](#) to be provided to `--kraken2_db`.

### 1.4.3 Functional annotation

We expect an ID mapping reference to be used within annotate. Since we're already expecting by default the NCBI-nr to be used as the alignment reference, [the ID mapping data file provided by Uniprot](#) should work well when provided to `--id_mapping`.

### 1.4.4 Host reference

If you're using metagenomic reads that come from a known host's microbiome, you can also provide the host's genome FASTA to `--host_fasta` parameter in order to enable our decontamination subworkflow. [Ensembl](#) provides easy to download genomes that can be used for this purpose. Alternatively, you can provide a pre-built BowTie2 database directory to the `--bowtie2_db` parameter.

## 1.5 Documentation

The dalmolingroup/euryale documentation is split into the following pages:

- [Usage](#)
  - An overview of how the pipeline works, how to run it and a description of all of the different command-line flags.
- [Output](#)
  - An overview of the different results produced by the pipeline and how to interpret them.

## 1.6 Credits

dalmolingroup/euryale was originally written by João Cavalcante.

We thank the following people for their extensive assistance in the development of this pipeline:

- Diego Morais (for developing the original [MEDUSA](#) pipeline)

## 1.7 Citations

Morais DAA, Cavalcante JVF, Monteiro SS, Pasquali MAB and Dalmolin RJS (2022) MEDUSA: A Pipeline for Sensitive Taxonomic Classification and Flexible Functional Annotation of Metagenomic Shotgun Sequences. *Front. Genet.* 13:814437. doi: 10.3389/fgene.2022.814437

This pipeline uses code and infrastructure developed and maintained by the [nf-core](#) community, reused here under the [MIT license](#).

**The nf-core framework for community-curated bioinformatics pipelines.**

Philip Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm, Maxime Ulysse Garcia, Paolo Di Tommaso & Sven Nahnsen.

*Nat Biotechnol.* 2020 Feb 13. doi: [10.1038/s41587-020-0439-x](https://doi.org/10.1038/s41587-020-0439-x).

## 2 dalmolingroup/euryale: Usage

*Documentation of pipeline parameters is generated automatically from the pipeline schema and can no longer be found in markdown files.*

### 2.1 Introduction

### 2.2 Samplesheet input

You will need to create a samplesheet with information about the samples you would like to analyse before running the pipeline. Use this parameter to specify its location. It has to be a comma-separated file with 3 columns, and a header row as shown in the examples below.

```
--input '[path to samplesheet file]'
```

#### 2.2.1 Multiple runs of the same sample

The `sample` identifiers have to be the same when you have re-sequenced the same sample more than once e.g. to increase sequencing depth. The pipeline will concatenate the raw reads before performing any downstream analysis. Below is an example for the same sample sequenced across 3 lanes:

```
sample,fastq_1,fastq_2
CONTROL_REP1,AEG588A1_S1_L002_R1_001.fastq.gz,AEG588A1_S1_L002_R2_001.fastq.gz
CONTROL_REP1,AEG588A1_S1_L003_R1_001.fastq.gz,AEG588A1_S1_L003_R2_001.fastq.gz
CONTROL_REP1,AEG588A1_S1_L004_R1_001.fastq.gz,AEG588A1_S1_L004_R2_001.fastq.gz
```

#### 2.2.2 Full samplesheet

The pipeline will auto-detect whether a sample is single- or paired-end using the information provided in the samplesheet. The samplesheet can have as many columns as you desire, however, there is a strict requirement for the first 3 columns to match those defined in the table below.

A final samplesheet file consisting of both single- and paired-end data may look something like the one below. This is for 6 samples, where `TREATMENT_REP3` has been sequenced twice.

```
sample,fastq_1,fastq_2
CONTROL_REP1,AEG588A1_S1_L002_R1_001.fastq.gz,AEG588A1_S1_L002_R2_001.fastq.gz
CONTROL_REP2,AEG588A2_S2_L002_R1_001.fastq.gz,AEG588A2_S2_L002_R2_001.fastq.gz
CONTROL_REP3,AEG588A3_S3_L002_R1_001.fastq.gz,AEG588A3_S3_L002_R2_001.fastq.gz
TREATMENT_REP1,AEG588A4_S4_L003_R1_001.fastq.gz,
TREATMENT_REP2,AEG588A5_S5_L003_R1_001.fastq.gz,
TREATMENT_REP3,AEG588A6_S6_L003_R1_001.fastq.gz,
TREATMENT_REP3,AEG588A6_S6_L004_R1_001.fastq.gz,
```

Column	Description
<code>sample</code>	Custom sample name. This entry will be identical for multiple sequencing libraries.
<code>fastq_1</code>	Full path to FastQ file for Illumina short reads 1. File has to be gzipped and have 1
<code>fastq_2</code>	Full path to FastQ file for Illumina short reads 2. File has to be gzipped and have 1

An [example samplesheet](#) has been provided with the pipeline.

## 2.3 Running the pipeline

The typical command for running the pipeline is as follows:

```
nextflow run dalmolingroup/euryale --input samplesheet.csv --outdir <OUTDIR> --genom
```

This will launch the pipeline with the `docker` configuration profile. See below for more information about profiles.

Note that the pipeline will create the following files in your working directory:

```
work                # Directory containing the nextflow working files
<OUTDIR>            # Finished results in specified location (defined with --outdir)
.nextflow_log       # Log file from Nextflow
# Other nextflow hidden files, eg. history of pipeline runs and old logs.
```

### 2.3.1 Updating the pipeline

When you run the above command, Nextflow automatically pulls the pipeline code from GitHub and stores it as a cached version. When running the pipeline after this, it will always use the cached version if available - even if the pipeline has been updated since. To make sure that you're running the latest version of the pipeline, make sure that you regularly update the cached version of the pipeline:

```
nextflow pull dalmolingroup/euryale
```

## 2.3.2 Reproducibility

It is a good idea to specify a pipeline version when running the pipeline on your data. This ensures that a specific version of the pipeline code and software are used when you run your pipeline. If you keep using the same tag, you'll be running the same version of the pipeline, even if there have been changes to the code since.

First, go to the [dalmolingroup/euryale releases page](#) and find the latest pipeline version - numeric only (eg. `1.3.1`). Then specify this when running the pipeline with `-r` (one hyphen) - eg. `-r 1.3.1`. Of course, you can switch to another version by changing the number after the `-r` flag.

This version number will be logged in reports when you run the pipeline, so that you'll know what you used when you look back in the future. For example, at the bottom of the MultiQC reports.

## 2.4 Core Nextflow arguments

**NB:** These options are part of Nextflow and use a *single* hyphen (pipeline parameters use a double-hyphen).

### 2.4.1 `-profile`

Use this parameter to choose a configuration profile. Profiles can give configuration presets for different compute environments.

Several generic profiles are bundled with the pipeline which instruct the pipeline to use software packaged using different methods (Docker, Singularity, Podman, Shifter, Charliecloud, Conda) - see below.

We highly recommend the use of Docker or Singularity containers for full pipeline reproducibility, however when this is not possible, Conda is also supported.

Note that multiple profiles can be loaded, for example: `-profile test,docker` - the order of arguments is important! They are loaded in sequence, so later profiles can overwrite earlier profiles.



If `-profile` is not specified, the pipeline will run locally and expect all software to be installed and available on the `PATH`. This is *not* recommended, since it can lead to different results on different machines dependent on the computer environment.

- `test`
  - A profile with a complete configuration for automated testing
  - Includes links to test data so needs no other parameters
- `docker`
  - A generic configuration profile to be used with [Docker](#)
- `singularity`
  - A generic configuration profile to be used with [Singularity](#)
- `podman`
  - A generic configuration profile to be used with [Podman](#)
- `shifter`
  - A generic configuration profile to be used with [Shifter](#)
- `charliecloud`
  - A generic configuration profile to be used with [Charliecloud](#)
- `conda`
  - A generic configuration profile to be used with [Conda](#). Please only use Conda as a last resort i.e. when it's not possible to run the pipeline with Docker, Singularity, Podman, Shifter or Charliecloud.

### 2.4.2 `-resume`

Specify this when restarting a pipeline. Nextflow will use cached results from any pipeline steps where the inputs are the same, continuing from where it got to previously. For input to be considered the same, not only the names must be identical but the files' contents as well. For more info about this parameter, see [this blog post](#).

You can also supply a run name to resume a specific run: `-resume [run-name]`. Use the `nextflow log` command to show previous run names.

### 2.4.3 `-c`

Specify the path to a specific config file (this is a core Nextflow command). See the [nf-core website documentation](#) for more information.

## 2.5 Custom configuration

### 2.5.1 Resource requests

Whilst the default requirements set within the pipeline will hopefully work for most people and with most input data, you may find that you want to customise the compute resources that the pipeline requests. Each step in the pipeline has a default set of requirements for number of CPUs, memory and time. For most of the steps in the pipeline, if the job exits with any of the error codes specified [here](#) it will automatically be resubmitted with higher requests (2 x original, then 3 x original). If it still fails after the third attempt then the pipeline execution is stopped.

For example, if the nf-core/rnaseq pipeline is failing after multiple re-submissions of the `STAR_ALIGN` process due to an exit code of `137` this would indicate that there is an out of memory issue:

```
[62/149eb0] NOTE: Process `NFCORE_RNASEQ:RNASEQ:ALIGN_STAR:STAR_ALIGN (WT_REP1)` terminated with error
Error executing process > 'NFCORE_RNASEQ:RNASEQ:ALIGN_STAR:STAR_ALIGN (WT_REP1)'

Caused by:
  Process `NFCORE_RNASEQ:RNASEQ:ALIGN_STAR:STAR_ALIGN (WT_REP1)` terminated with a non-zero exit code

Command executed:
  STAR \
    --genomeDir star \
    --readFilesIn WT_REP1_trimmed.fq.gz \
    --runThreadN 2 \
    --outFileNamePrefix WT_REP1. \
    <TRUNCATED>

Command exit status:
  137

Command output:
  (empty)

Command error:
  .command.sh: line 9: 30 Killed      STAR --genomeDir star --readFilesIn WT_REP1_1
Work dir:
  /home/pipelinetest/work/9d/172ca5881234073e8d76f2a19c88fb

Tip: you can replicate the issue by changing to the process work dir and entering the command
```

#### 2.5.1.1 For beginners

A first step to bypass this error, you could try to increase the amount of CPUs, memory, and time for the whole pipeline. Therefore you can try to increase the resource for the parameters `--max_cpus`, `--max_memory`, and `--max_time`. Based on the error above, you have to increase the amount of memory. Therefore you can go to the [parameter documentation of rnaseq](#) and scroll down to the `show hidden parameter` button to get the default value for `--max_memory`. In this case 128GB, you then can try to run your pipeline again with `--max_memory 200GB -resume` to skip all process, that were already calculated. If you can not increase the resource of the complete pipeline, you can try to adapt the resource for a single process as mentioned below.

### 2.5.1.2 Advanced option on process level

To bypass this error you would need to find exactly which resources are set by the `STAR_ALIGN` process. The quickest way is to search for `process STAR_ALIGN` in the [nf-core/rnaseq Github repo](#). We have standardised the structure of Nextflow DSL2 pipelines such that all module files will be present in the `modules/` directory and so, based on the search results, the file we want is `modules/nf-core/star/align/main.nf`. If you click on the link to that file you will notice that there is a `label` directive at the top of the module that is set to `label process_high`. The Nextflow `label` directive allows us to organise workflow processes in separate groups which can be referenced in a configuration file to select and configure subset of processes having similar computing requirements. The default values for the `process_high` label are set in the pipeline's `base.config` which in this case is defined as 72GB. Providing you haven't set any other standard nf-core parameters to cap the [maximum resources](#) used by the pipeline then we can try and bypass the `STAR_ALIGN` process failure by creating a custom config file that sets at least 72GB of memory, in this case increased to 100GB. The custom config below can then be provided to the pipeline via the `-c` parameter as highlighted in previous sections.

```
process {
  withName: 'NFCORE_RNASEQ:RNASEQ:ALIGN_STAR:STAR_ALIGN' {
    memory = 100.GB
  }
}
```

**NB:** We specify the full process name i.e.

`NFCORE_RNASEQ:RNASEQ:ALIGN_STAR:STAR_ALIGN` in the config file because this takes priority over the short name (`STAR_ALIGN`) and allows existing configuration using the full process name to be correctly overridden.

If you get a warning suggesting that the process selector isn't recognised check that the process name has been specified correctly.

## 2.5.2 Updating containers (advanced users)

The [Nextflow DSL2](#) implementation of this pipeline uses one container per process which makes it much easier to maintain and update software dependencies. If for some reason you need to use a different version of a particular tool with the pipeline then you just need to identify the `process` name and override the Nextflow `container` definition for that process using the `withName` declaration. For example, in the [nf-core/viralrecon](#) pipeline a tool called [Pangolin](#) has been used during the COVID-19 pandemic to assign lineages to SARS-CoV-2 genome sequenced samples. Given that the lineage assignments change quite frequently it doesn't make sense to re-release the `nf-core/viralrecon` everytime a new version of Pangolin has been released. However, you can override the default container used by the pipeline by creating a custom config file and passing it as a command-line argument via `-c custom.config`.

1. Check the default version used by the pipeline in the module file for [Pangolin](#)
2. Find the latest version of the Biocontainer available on [Quay.io](#)
3. Create the custom config accordingly:
4. For Docker:

```
nextflow
process {
    withName: PANGOLIN {
        container = 'quay.io/biocontainers/pangolin:3.0.5--pyhdfd78af_0'
    }
}
```

5. For Singularity:

```
nextflow
process {
    withName: PANGOLIN {
        container = 'https://depot.galaxyproject.org/singularity/pangolin:3.0.5--pyhdfd78af_0'
    }
}
```

6. For Conda:

```
nextflow
process {
  withName: PANGOLIN {
    conda = 'bioconda::pangolin=3.0.5'
  }
}
```

**NB:** If you wish to periodically update individual tool-specific results (e.g. Pangolin) generated by the pipeline then you must ensure to keep the `work/` directory otherwise the `-resume` ability of the pipeline will be compromised and it will restart from scratch.

## 2.6 Running in the background

Nextflow handles job submissions and supervises the running jobs. The Nextflow process must run until the pipeline is finished.

The Nextflow `-bg` flag launches Nextflow in the background, detached from your terminal so that the workflow does not stop if you log out of your session. The logs are saved to a file.

Alternatively, you can use `screen` / `tmux` or similar tool to create a detached session which you can log back into at a later time. Some HPC setups also allow you to run nextflow within a cluster job submitted your job scheduler (from where it submits more jobs).

## 2.7 Nextflow memory requirements

In some cases, the Nextflow Java virtual machines can start to request a large amount of memory. We recommend adding the following line to your environment to limit this (typically in `~/.bashrc` or `~/.bash_profile`):

```
NXF_OPTS=' -Xms1g -Xmx4g '
```

# 3 dalmolingroup/euryale: Output

## 3.1 Introduction

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report, which summarises results at the end of the pipeline.

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

## 3.2 Pipeline overview

The pipeline is built using [Nextflow](#) and processes data using the following steps (steps in *italics* don't run by default):

- [Kaiju](#) - Taxonomically classify reads or contigs
- [Krona](#) - Visualize the taxonomic classification for each sample.
- [Diamond](#) - Alignment reads and contigs against a reference database (such as NCBI-nr).
- [Annotate](#) - Functional annotation of alignment matches.
- [MEGAHIT](#) - Assembled contigs.
- [MultiQC](#) - Aggregate report describing results and QC from the whole pipeline
- [Pipeline information](#) - Report metrics generated during the workflow execution

### 3.2.1 Kaiju

#### ► Output files

[Kaiju](#) is a software to perform fast taxonomic classification of metagenomic sequencing reads using a protein reference database.

### 3.2.2 Krona

#### ► Output files

- [Krona](#) is a tool to interactively explore metagenomes and more from a web browser.

### 3.2.3 Diamond

#### ► Output files

- [DIAMOND](#) is an accelerated BLAST compatible local sequence aligner.

### 3.2.4 Annotate

► Output files

- [Annotate](#) is a tool to annotate each query using the best alignment for which a mapping is known.

### 3.2.5 MEGAHIT

► Output files

- [MEGAHIT](#) is an ultra-fast and memory-efficient (meta-)genome assembler

## 3.3 *DIAMOND database*

► Output files

- This output is present if you add the `--save_db` parameter.
- [DIAMOND](#) is an accelerated BLAST compatible local sequence aligner.

### 3.3.1 MultiQC

► Output files

[MultiQC](#) is a visualization tool that generates a single HTML report summarising all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

Results generated by MultiQC collate pipeline QC from supported tools e.g. FastQC. The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see <http://multiqc.info>.

### 3.3.2 Pipeline information

► Output files

[Nextflow](#) provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.

## 4 dalmolingroup/euryale: Citations

### 4.1 [nf-core](#)

Ewels PA, Peltzer A, Fillinger S, Patel H, Alneberg J, Wilm A, Garcia MU, Di Tommaso P, Nahnsen S. The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol.* 2020 Mar;38(3):276-278. doi: 10.1038/s41587-020-0439-x. PubMed PMID: 32055031.

### 4.2 [Nextflow](#)

Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. *Nat Biotechnol.* 2017 Apr 11;35(4):316-319. doi: 10.1038/nbt.3820. PubMed PMID: 28398311.

### 4.3 Pipeline tools

- [FastQC](#)
- [MultiQC](#)

Ewels P, Magnusson M, Lundin S, Källér M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics.* 2016 Oct 1;32(19):3047-8. doi: 10.1093/bioinformatics/btw354. Epub 2016 Jun 16. PubMed PMID: 27312411; PubMed Central PMCID: PMC5039924.

### 4.4 Software packaging/containerisation tools

- [Anaconda](#)

Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web.

- [Bioconda](#)

Grüning B, Dale R, Sjödin A, Chapman BA, Rowe J, Tomkins-Tinch CH, Valieris R, Köster J; Bioconda Team. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods.* 2018 Jul;15(7):475-476. doi: 10.1038/s41592-018-0046-7. PubMed PMID: 29967506.

- [BioContainers](#)



da Veiga Leprevost F, Grüning B, Aflitos SA, Röst HL, Uszkoreit J, Barsnes H, Vaudel M, Moreno P, Gatto L, Weber J, Bai M, Jimenez RC, Sachsenberg T, Pfeuffer J, Alvarez RV, Griss J, Nesvizhskii AI, Perez-Riverol Y. BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics*. 2017 Aug 15;33(16):2580-2582. doi: 10.1093/bioinformatics/btx192. PubMed PMID: 28379341; PubMed Central PMCID: PMC5870671.

- [Docker](#)
- [Singularity](#)

Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. *PLoS One*. 2017 May 11;12(5):e0177459. doi: 10.1371/journal.pone.0177459. eCollection 2017. PubMed PMID: 28494014; PubMed Central PMCID: PMC5426675.

# I. Reference

# 5 dalmolingroup/euryale pipeline parameters

A pipeline for metagenomic taxonomic classification and functional annotation. Based on MEDUSA.

## 5.1 Input/output options

Define where the pipeline should find input data and save output data.

Parameter	Description
<code>input</code>	Path to comma-separated file containing information about the samples in ► Help
<code>outdir</code>	The output directory where the results will be saved. You have to use abso
<code>email</code>	Email address for completion summary. ► Help
<code>multiqc_title</code>	MultiQC report title. Printed as page header, used for filename if not other
<code>save_dbs</code>	Save DIAMOND db to results directory after construction

## 5.2 Skip Steps

Choose to skip pipeline steps

Parameter	Description	Type	Default	Require
<code>skip_classification</code>	Skip taxonomic classification	<code>boolean</code>		
<code>skip_alignment</code>	Skip alignment	<code>boolean</code>		
<code>skip_functional</code>	Skip functional annotation	<code>boolean</code>		
<code>skip_host_removal</code>	Skip host removal	<code>boolean</code>		
<code>skip_microview</code>	Skip MicroView report	<code>boolean</code>		
<code>skip_preprocess</code>	Skip Preprocessing steps	<code>boolean</code>		

## 5.3 Decontamination

Parameter	Description	Type	D
<code>host_fasta</code>	Host FASTA to use for decontamination	<code>string</code>	
<code>bowtie2_db</code>	Pre-built bowtie2 index. Directory where index is located.	<code>string</code>	

## 5.4 Alignment

Parameter	Description	Type	Default	Required
<code>reference_fasta</code>	Path to FASTA genome file.	<code>string</code>		
<code>diamond_db</code>	Path to pre-built DIAMOND db.	<code>string</code>		

## 5.5 Taxonomy

Parameter	Description	Type	Default	Required	Hidden
<code>kaiju_db</code>	Kaiju database	<code>string</code>		True	
<code>kraken2_db</code>	Kraken2 database	<code>string</code>			
<code>run_kaiju</code>	Run Kaiju classifier	<code>boolean</code>	True		
<code>run_kraken2</code>	Run Kraken2 classifier	<code>boolean</code>			

## 5.6 Functional

Parameter	Description	Ty
<code>id_mapping</code>	Path to ID mapping file to be used for the Functional annotation	<code>s</code>
<code>minimum_bitscore</code>	Minimum bitscore of a match to be used for annotation	<code>i</code>
<code>minimum_pident</code>	Minimum identity of a match to be used for annotation	<code>i</code>
<code>minimum_alen</code>	Minimum alignment length of a match to be used for annotation	<code>i</code>
<code>maximum_evalue</code>	Maximum evalue of a match to be used for annotation	<code>n</code>

## 5.7 Assembly

Parameter	Description	Type	Default	Required	Hidden
<code>assembly_based</code>		<code>boolean</code>			

## 5.8 Reference genome options

Reference genome related files and options required for the workflow.

Parameter	Description	Type	Default
<code>genome</code>	Name of iGenomes reference. ► Help	<code>string</code>	
<code>igenomes_base</code>	Directory / URL base for iGenomes references.	<code>string</code>	s3://n
<code>igenomes_ignore</code>	Do not load the iGenomes reference config. ► Help	<code>boolean</code>	
<code>fasta</code>		<code>string</code>	

## 5.9 Download Entry

Parameter	Description	Type
<code>download_functional</code>	Whether to dowload functional references	<code>boolean</code>
<code>download_kaiju</code>	Whether to dowload the Kaiju reference db	<code>boolean</code>
<code>download_kraken</code>	Whether to dowload the Kraken2 reference db	<code>boolean</code>
<code>download_host</code>	Whether to download the host reference genome	<code>boolean</code>
<code>functional_db</code>	Functional reference URL (download entry)	<code>string</code>
<code>functional_dictionary</code>	Functional dictionary URL (download entry)	<code>string</code>
<code>kaiju_db_url</code>	Kaiju reference URL (download entry)	<code>string</code>
<code>kraken2_db_url</code>	Kraken2 reference URL (download entry)	<code>string</code>
<code>host_url</code>	Host FASTA reference URL (download entry)	<code>string</code>

## 5.10 Max job request options

Set the top limit for requested resources for any single job.

Parameter	Description	Type
<code>max_cpus</code>	Maximum number of CPUs that can be requested for any single job. ► Help	integer
<code>max_memory</code>	Maximum amount of memory that can be requested for any single job. ► Help	string
<code>max_time</code>	Maximum amount of time that can be requested for any single job. ► Help	string

## 5.11 Generic options

Less common options for the pipeline, typically set in a config file.

Parameter	Description
<code>help</code>	Display help text.
<code>version</code>	Display version and exit.
<code>publish_dir_mode</code>	Method used to save pipeline results to output directory. ► Help
<code>email_on_fail</code>	Email address for completion summary, only when pipeline fails. ► Help
<code>plaintext_email</code>	Send plain-text email instead of HTML.
<code>max_multiqc_email_size</code>	File size limit when attaching MultiQC reports to summary email.
<code>monochrome_logs</code>	Do not use coloured log outputs.
<code>hook_url</code>	Incoming hook URL for messaging service ► Help
<code>multiqc_config</code>	Custom config file to supply to MultiQC.
<code>multiqc_logo</code>	Custom logo file to supply to MultiQC. File name must also be specified in the config file.
<code>multiqc_methods_description</code>	Custom MultiQC yaml file containing HTML including a method description.
<code>tracedir</code>	Directory to keep pipeline Nextflow logs and reports.
<code>validate_params</code>	Boolean whether to validate parameters against the schema.

Parameter	Description
<code>show_hidden_params</code>	Show all params when using <code>--help</code> ► Help
<code>schema_ignore_params</code>	